Computer Science and Information Systems
Volume 17, Number 1, January 2020

## Contents

ISSN: 1820-0214 (Print) 2406-1018 (Online)

Computer Science and Information Systems
Vol 17, No 1, January 2020

# Computer Science and Information Systems

Published by ComSIS Consortium

Volume 17, Number 1
January 2020

ComSIS is an international journal published by the ComSIS Consortium

**ComSIS Consortium:**

**University of Belgrade:**
Faculty of Organizational Science, Belgrade, Serbia
Faculty of Mathematics, Belgrade, Serbia
School of Electrical Engineering, Belgrade, Serbia

**Serbian Academy of Science and Art:**
Mathematical Institute, Belgrade, Serbia

**Union University:**
School of Computing, Belgrade, Serbia

**University of Novi Sad:**
Faculty of Sciences, Novi Sad, Serbia
Faculty of Technical Sciences, Novi Sad, Serbia
Faculty of Economics, Subotica, Serbia
Technical Faculty "Mihajlo Pupin", Zrenjanin, Serbia

**University of Montenegro:**
Faculty of Economics, Podgorica, Montenegro

**ComSIS Editorial Office:**
**University of Novi Sad, Faculty of Sciences,**
**Department of Mathematics and Informatics**
Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia
**Phone:** +381 21 458 888; **Fax:** +381 21 6350 458
www.comsis.org; Email: comsis@uns.ac.rs

**Volume 17, Number 1, 2020**
**Novi Sad**

# Computer Science and Information Systems

**ISSN: 1820-0214 (Print) 2406-1018 (Online)**

The ComSIS journal is sponsored by:

Ministry of Education, Science and Technological Development of the Republic of Serbia
http://www.mpn.gov.rs/

# Computer Science and Information Systems

## AIMS AND SCOPE

Computer Science and Information Systems (ComSIS) is an international refereed journal, published in Serbia. The objective of ComSIS is to communicate important research and development results in the areas of computer science, software engineering, and information systems.

We publish original papers of lasting value covering both theoretical foundations of computer science and commercial, industrial, or educational aspects that provide new insights into design and implementation of software and information systems. In addition to wide-scope regular issues, ComSIS also includes special issues covering specific topics in all areas of computer science and information systems.

ComSIS publishes invited and regular papers in English. Papers that pass a strict reviewing procedure are accepted for publishing. ComSIS is published semiannually.

### Indexing Information

ComSIS is covered or selected for coverage in the following:

- Science Citation Index (also known as SciSearch) and Journal Citation Reports / Science Edition by Thomson Reuters, with 2018 two-year impact factor 0.620,
- Computer Science Bibliography, University of Trier (DBLP),
- EMBASE (Elsevier),
- Scopus (Elsevier),
- Summon (Serials Solutions),
- EBSCO bibliographic databases,
- IET bibliographic database Inspec,
- FIZ Karlsruhe bibliographic database io-port,
- Index of Information Systems Journals (Deakin University, Australia),
- Directory of Open Access Journals (DOAJ),
- Google Scholar,
- Journal Bibliometric Report of the Center for Evaluation in Education and Science (CEON/CEES) in cooperation with the National Library of Serbia, for the Serbian Ministry of Education and Science,
- Serbian Citation Index (SCIndeks),
- doiSerbia.

### Information for Contributors

The Editors will be pleased to receive contributions from all parts of the world. An electronic version (MS Word or LaTeX), or three hard-copies of the manuscript written in English, intended for publication and prepared as described in "Manuscript Requirements" (which may be downloaded from http://www.comsis.org), along with a cover letter containing the corresponding author's details should be sent to official journal e-mail.

## Criteria for Acceptance

Criteria for acceptance will be appropriateness to the field of Journal, as described in the Aims and Scope, taking into account the merit of the content and presentation. The number of pages of submitted articles is limited to 20 (using the appropriate Word or LaTeX template).

Manuscripts will be refereed in the manner customary with scientific journals before being accepted for publication.

## Copyright and Use Agreement

# Computer Science and Information Systems

Volume 17, Number 1, January 2020

## CONTENTS

## Papers selected from 8th International Conference on Model and Data Engineering (MEDI 2018)

# Editorial

Mirjana Ivanović[1] and Miloš Radovanović[1]

University of Novi Sad, Faculty of Sciences
Novi Sad, Serbia
{mira,radacha}@dmi.uns.ac.rs

Volume 17 of the Computer Science and Information Systems journal, for the year 2020, is started with this issue consisting of 11 regular articles, as well as 5 articles within the special section dedicated to extended versions of papers published in the proceedings of MEDI 2018, 8th International Conference on Model and Data Engineering, which took place in Marrakesh, Morocco, October 24-26, 2018. We thank the guest editors, Djamal Benslimane, Stephane Jean, Ladjel Bellatreche, and Kazumi Nakamatsu, and acknowledge the hard work and enthusiasm of our authors and reviewers, without whom the current issue would not have been possible.

This issue starts with "Run-time Interpretation of Information System Application Models in Mobile Cloud Environments" by Nikola Tanković and Tihana Galinac Grbac, which proposes an architectural framework for building distributed information system applications in which application models are directly interpreted during execution, shortening evaluation cycles and providing faster feedback to developers. The application model is represented as a graph structure complemented with a procedural action scripting language that can express more complex software behavior. A case study in a mobile cloud environment showed the approach shortened the requirements engineering process and automated the configuration and deployment process by providing more engagement of end-users.

In the second article, "Efficient Virtual Machine Placement Algorithms for Consolidation in Cloud Data Centers," Loiy Alsbatin et al. propose new algorithms for virtual machine (VM) placement for the problem of dynamic VM consolidation in cloud environments. The two proposed algorithms, CPU Priority based Best-Fit Decreasing (CPBFD) and Dynamic CPU Priority based Best-Fit Decreasing (DCPBFD), are compared with the state-of-the-art algorithms for VM placement through simulations with real-world workload traces, showing that the proposed algorithms provide the least service level agreement violations, least VM migrations, and efficient energy consumption.

"Towards a Software-Based Mobility Management for 5G: An Experimental Approach for Flattened Network Architectures," by Jesús Calle-Cancho et al. proposes a novel mobility management solution which takes advantage of software defined networking (SDN). The approach avoids the use of IP-IP tunnels and adds the dynamic flow management capability provided by SDN. Comparison with the network-based distributed mobility management (NB-DMM) approach is provided through an analytical model and experimental testbed, showing that the proposed solution achieves better efficiency in terms of signaling and routing cost.

Stefanos Ougiaroglou et al., in "Instance-Based Classification Using Prototypes Generated from Large Noisy and Streaming Datasets," present a new variation of their algorithm for instance-based classification aimed at streaming data, that maintains prototypes in a convenient and manageable way. This is achieved by removing the weakest prototype when a new prototype is generated. The experimental results reveal that the proposed algorithm is as accurate as its predecessor, but is more efficient and noise tolerant.

"Climate Change Opinions in Online Debate Sites," authored by Adrian Groza et al., focuses on developing technical instrumentation for making sense of a set of online arguments concerning climate change, and aggregating them into usable results for policy making and climate science communication. The objectives are three-fold: (1) aggregate arguments posted for a certain debate topic, (2) consolidate opinions posted under several but related topics, and (3) identify possible linguistic characteristics of the argumentative texts. The methods proposed to fulfill the given objectives may be used in domains different from climate change.

In the article entitled "Architecting Business Process Maps," Geert Poels et al. position the concept of process map within the domain of architecture description. That way, the authors identify and clarify diverging views of this concept as found in the literature and set requirements for describing process maps, producing a meta-model for a process mapping language. The meta-model allows investigating the suitability of enterprise architecture (EA) modeling languages as a basis for defining a domain-specific language for process mapping along with the creation of a better understanding of business process architecture in relation to EA.

"Correctness of the Chord Protocol" by Bojan Marinković et al. studies the Chord protocol – one of the first, the simplest and the most popular distributed protocols that are at the core of technologies such as the Internet of Things (IoT). Up to now, the Chord protocol has been applied without a formal proof of correctness. This article provides the proof of correctness of the Chord protocol using the logic of time and knowledge with the respect to the set of possible executions, called regular runs. A deterministic description of the correctness of the Chord protocol is provided, considering Chord actions that maintain a ring topology while the nodes can freely join or leave.

The article "Distance Transform and Template Matching Based Methods for Localization of Barcodes and QR Codes," by Melinda Katona et al. addresses the problem of the existence of a wide variety of bar/QR code types, sizes, noise levels and blurring by introducing two methods for localization of 1D barcodes based on template matching and distance transformation, and a third method for QR codes, which are able to simultaneously localize several different types of codes. Experimental evaluation shows improvement over previous approaches.

Tina Beranič and Marjan Heričko in their article "Comparison of systematically derived software metrics thresholds for object-oriented programming languages," present the results of an empirical study aimed at comparing systematically obtained threshold values for nine software metrics in four object-oriented programming languages (Java, C++, C#, and Python). The challenges in the threshold derivation domain were addressed within introduced adjustments of the benchmark-based threshold derivation approach. The comparison reveals that threshold values differ between different programming languages.

In "Regression Verification for Automated Evaluation of Students Programs," Milena Vujošević Janičić and Filip Marić propose an approach that provides precise assessment of functional correctness of student programs based on a form of software verification founded on formal static analysis of code called regression verification. Furthermore, the paper describes the open-source, publicly available implementation of the approach, built on top of compiler infrastructure LLVM and the software verification tool LAV. Evaluation of the approach on two real-world corpora of student programs shows that the approach can be used as a precise and reliable supplementary technique in grading of student programs in various computer-science courses, as well as programming competitions.

Finally, “Visualization of path patterns in semantic graphs,” by José Paulo Leal presents a precise definition of a-graphs (a novel kind of graph designed to highlight path patterns using summarization) and of the mapping of semantic graphs into a-graphs. Visualization is obtained with a-graphs diagrams, and a web application to visualize and interact with these diagrams was implemented to validate the proposed approach. Diagrams of well-known semantic graphs are presented to illustrate the use of a-graphs for discovering path patterns in different settings. The validation with large semantic graphs is the basis for a discussion on the insights provided by a-graphs on large semantic graphs: the difference between a-graphs and ontologies, path pattern visualization using a-graphs and the challenges posed by large semantic graphs.

Computer Science and Information Systems 17(1): v–vii https://doi.org/10.2298/CSIS200100vB

# Guest Editorial
# Papers selected from 8th International Conference on Model and Data Engineering (MEDI 2018)

Djamal Benslimane[1], Stéphane Jean[2], Ladjel Bellatreche[2], and Kazumi Nakamatsu[3]

[1] University of Claude Bernard Lyon 1
Lyon, France
djamal.benslimane@univ-lyon1.fr
[2] LIAS/ISAE-ENSMA – Poitiers University
Poitiers, France
(jean,bellatreche)@ensma.fr
[3] University of Hyogo
Hyogo, Japan
nakamatu@pf7.so-net.ne.jp

This special issue aims at shedding the light on some recent and significant advances in the field of model and data engineering. It presents selected papers from the scientific workshops that were held in conjunction with the 8th International Conference on Model and Data Engineering (MEDI 2018), which took place in Marrakesh, Morocco, October 24-26, 2018. MEDI 2018 attracted four workshops on a wide range of topics that fall into the main area of the MEDI 2018 conference:

1. The Model and Data Engineering for Social Good Workshop (MEDI4SG)
2. The international workshop on moDeling, vErification and Testing of dEpendable CriTical systems (DETECT)
3. The International Workshop: Formal Model for Multifaceted Systems (REMEDY)
4. The Second International Workshop on Cybersecurity and Functional Safety in Cyber-Physical Systems (IWCFS)

This special issue was managed as follows: the organizers of the first three workshops proposed their best paper, except DETECT that suggested two papers. This is because this workshop had a good acceptance rate since it attracted 19 submissions and only six papers have been accepted. We also invited one paper from the MEDI 2018 main conference based on its good ranking. The content of each paper has been extended by at least 30%. After the second round of reviews, we finally accepted five papers.
We congratulate the authors who submitted articles to MEDI 2018 workshops.

The five selected papers are summarized as follows:

The first article titled, "A Mobile Crowd Sensing Framework for Suspect Investigation: An Objectivity Analysis and De-Identification Approach", by Hasna Elalaoui Elabdallaoui, Abdelaziz Elfazziki, Fatima Zohra Ennaji, Mohamed Sadgal introduces an approach to develop a crowdsourcing framework allowing a wider collaboration between citizens and their authorities. It mainly allows collecting information on crimes and suspects, computing users credibility and information reliability. A de-identification mechanism is also used to anonymize data users. The proposed framework is generic, can be used in different contexts and is suitable for any type of crimes that can be witnessed by citizen participants. Unsupervised machine learning techniques are used to cluster reported locations on crimes before applying an objectivity analysis. This latter is based on a probabilistic algorithm to identify the most reliable crime locations.

The second article titled, ”Verification and Testing of Safety-Critical Airborne Systems: a Model-based Methodology”, by Mounia Elqortobi, Warda El-Khouly, Amine Rahj, Jamal Bentahar, Rachida Dssouli highlights the importance of formal verification and testing activities in the avionics software development cycle. It then addresses the safety-critical software verification and testing issue and proposes to integrate model-based verification and model-based testing within a single framework. The defined framework starts first by formally modeling the safety-critical airborne system from informal and consistent requirement specifications, and produces an FSM-like model. The obtained model is then refined and encoded in the extended Interpreted Systems Programming Language (ISPL+). Computation Tree Logic (CTL) is also used to extract and express the system requirements in the form of temporal properties. Intended properties are automatically checked and witness-examples or counter-examples are generated to either prove the satisfaction of properties or guide designers to detect and repair errors in the formal system model.

The third article titled, ”Business Process Specification, Verification, and Deployment in a Mono-Cloud, Multi-Edge Context”, by Saoussen Cheikhrouhou, Slim Kallel, Ikbel Guidara, Zakaria Maamar focuses on the satisfaction of time-constrained business processes in the context of cloud and edge-based resources. It presents an approach to formally specify and verify cloud resources allocation to business processes using Time Petri- Nets model, and to fragmenting and deploying free-of-violations time-constrained business processes on mono-cloud and multi-edge context. Different phases are introduced at both design and run time, including specification, (ongoing) placement, transformation, and (ongoing) verification. The ongoing verification phase produces a list of violations that are handled in the ongoing placement by producing corrective actions, mainly on where future data and tasks should be re-placed to better satisfy time constraints.

The fourth article titled, ”A Tool-assisted Method for the Systematic Construction of Critical Embedded Systems using Event-B”, by Pascal André, Christian Attiogbé, Arnaud Lanoix concerns the critical embedded systems and proposes an Event-B based approach to support formal modeling of the full software design process to better address reliability and correctness construction requirements. The design of a companion tool of the proposed method is presented and experimented in the context of the landing gear case study which is a typical cyber-physical system involving the control and interaction of software and physical components.

The fifth article titled ”Game-based learning and Gamification to improve skills in early years’ education”, by Rachid Lamrani and El Hassan Abdelwahed focuses on how to improve children’s skills in their early years education and reduce the drop-out rate of learners through play-based learning and gamification mechanisms. The proposed system principles are aligned with the main orientations of the Montessori approach. Different factors for the assessment of the proposed solution are identified and a variety of serious gaming activities is provided.

We gratefully acknowledge the support of the contributors to this special issue and express our great esteem to the anonymous reviewers for the time and effort they have put in reviewing these papers. For readers of this volume, we hope you will find its content

interesting and will inspire you to look further into the challenges that still lie ahead in our digital society. We also would like to thank Prof. Mirjana Ivanovic, the editor in chief of the COMSIS journal for accepting to run this special issue.

Computer Science and Information Systems 17(1):1–27 https://doi.org/10.2298/CSIS180829021T

# Run-time Interpretation of Information System Application Models in Mobile Cloud Environments

Nikola Tanković[1] and Tihana Galinac Grbac[2]

[1] Faculty of Informatics, Juraj Dobrila University of Pula
nikola.tankovic@unipu.hr

[2] Faculty of Engineering, Juraj Dobrila University of Pula
tihana.galinac@unipu.hr

**Abstract.** Application models are commonly used in the development of information systems. Recent trends have introduced techniques by which models can be directly transformed into execution code and thus become a single source for application design. Inherently, it has been challenging for software developers to become proficient in designing entire systems due to the complex chain of model transformations and the further refinements required to the code generated from the models. We propose an architectural framework for building the distributed information system applications in which the application models are directly interpreted during execution. This approach shortens the evaluation cycles and provides faster feedback to developers. Our framework is based on a holistic application model represented as a graph structure complemented with a procedural action scripting language that can express more complex software behavior.
We present the implementation details of this framework architecture in a mobile cloud environment and evaluate its benefits in eleven projects for different customers in the retail, supply-chain management and merchandising domain involving 300 active application users. Our approach allowed engaging end-users in the software development process in the phase of specifying executable application models. It succeeded in shortening the requirements engineering process and automating the configuration and deployment process. Moreover, it benefited from the automatic synchronization of application updates for all active versions at the customer sites.

**Keywords:** model-driven development, MDD, cloud computing, information system, model interpretation, application graph model

## 1. Introduction

The information systems (IS) development process includes numerous repeating patterns such as constructing database schema, designing user interfaces for data display and manipulation, building communication services [1], etc. These and similar patterns arise during the process of human or semi-automatic translation of the system model artifacts to machine-executable code. Application modeling has become extremely relevant—not just for supporting analysis and design phases—but in serving as the primary source for automatic application generation [2]. Therefore, Model-Driven Development (MDD) advocates the automation of repetitive software development tasks.

In addition, such automation opens new opportunities for end-users to actively participate in the software development process, offloading the development tasks that were exclusive to software engineers [3]. Figure 1 depicts a generic scenario of how this is applied. The software developers can steer their focus on providing generic components and/or model transformation procedures that are actively reused through different application models defined by the end-user developers.

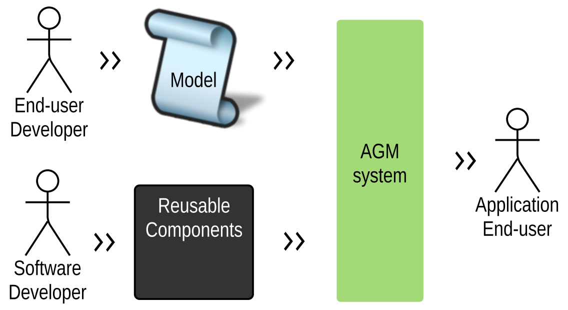


**Fig. 1.** Offloading the software developers by introducing modeling techniques understandable to end-users

Additionally, the cloud computing paradigm facilitates software provisioning by enabling software providers with large-scale infrastructure resources paid on a per usage basis [4]. The cloud services benefit from offloading software and system engineers due to the high level of automation.

This paper proposes an approach to alleviate the role of end-users in the development process of information systems. This goal is achieved by interpreting application models at runtime on a highly automated distributed cloud environment. We strive to gain benefits from: (1) enabling adaptive and rapid-feedback modeling in which changes can be made quickly and applied easily, (2) enabling easier software maintenance and distribution, and (3) runtime application version management with quick transition times between different end-application versions. Our approach is targeted primarily at small-to-medium-sized projects that require fast development cycles and a greater degree of exploration during the requirements-gathering phase.

To achieve model interpretation, we devised a technique that represents models at runtime as *directed graphs*, called the Application Graph Model (AGM). To supplement AGM for building more complex solutions, we provide a complementary action scripting language targeted to advanced users with a software programming skillset. This paper describes the AGM execution architecture and evaluates it with a concrete implementation for building mobile-enabled information systems [5] in the retail, supply chain management, and the merchandising domain.

The rest of this paper is organized as follows: Section 2 provides the rationale behind the model interpretation. Section 3 lays a conceptual foundation for interpreting models in the information system application domain. The implementation details are presented in Section 4 together with a discussion concerning our experiences of AGM usage in practice and an example application model. The limitations of our approach are disseminated in

Section 5. Related work is reviewed in Section 6. Finally, Section 7 concludes the paper and elaborates on the possibilities for further research.

## 2. Background

Two major strategies exist for transforming models to executable applications [6]: (1) the generative approach, in which models undergo a series of transformations that result in executable application code, and (2) the interpretative approach, in which models are exploited through runtime interpretation. Generative solutions yield better end applications performance-wise because runtime interpretation of models comes with additional execution cost. The existing strategies, such as the Model-Driven Architecture initiative (MDA) and the Eclipse Modeling Framework (EMF), focus primarily on the generative approach. Though application performance is superior, the generative approach requires the definition of a series of model transformation steps using different templates for compiling the higher-level model to lower-level programming code. Specifying such transformations can be extremely challenging [7]. Consequently, it involves a broader spectrum of highly specialized engineers. While such involvement is essential for large-scale software products, at the same time, such characteristics hinder large-scale MDD adoption.

Another ongoing research challenge associated with the generative approaches is the manual source-code refinement typically applied after the initial *model to code* transformations. While the code generated from models covers the majority of generic application functionalities, some parts of applications still require the implementation of specific business logic. These highly specific portions of applications are difficult to represent using high-level abstract models [8]. Therefore, specific functionality is often implemented after the initial code has been generated from the models. This post-model-generation code refinement requirement has several inherent drawbacks: (1) it impacts the synchronization between the model and the application source code, (2) it requires specialized software programming skills, and (3) it imposes a burden on model and application change and deployment management, because it requires additional housekeeping for each version of the model, transformation rules, code templates and customized code to maintain their compatibility [8]. To address these drawbacks, some solutions have proposed modeling specific functionalities by providing an abstract action domain-specific language [9,10] that is also transformed to concrete programming code through compilation. This approach works in keeping the models synchronized but raises other difficulties. For example, specifying model transformation rules requires expert knowledge and has a massive impact on the customer-engineer requirements negotiation process. Rapid prototyping becomes almost impossible: the time required for generating code, compiling, installing and restarting existing systems can range from several minutes to several hours [11].

## 3. AGM Solution

As we briefly discussed in a previous research paper [12], AGM reuses the *Object* and *Process* modeling approach [13] standardized in Automation systems and integration—the Object-Process Methodology, ISO/PAS 19450:2015. Object Process Methodology (OPM) is a holistic graphical modeling language applicable to a large variety of domains.

The main advantage of OPM is its ability to capture the dynamic and static aspects of a system within a single model. The OPM approach handles model complexity through refinement techniques [14] and alternative model views rather than splitting the modeling process using several semantically orthogonal modeling languages. The underlying model in OPM is holistic: it captures the complete end-solution within a single model.

Our solution builds upon the OPM research in several ways:

- We emphasize the holistic model idea derived from a reflexive meta-model; that is, a model that can describe itself. The *Meta-Object Facility (MOF)* [15] from the *Model-Driven Architecture (MDA)* initiative is also reflexive, but it is used to derive a set of semantically and not necessarily compatible models.
- We built the AGM meta-model in a manner similar to the OPM meta-model, but it possesses specialized constructs for defining mobile information system applications.

In the following subsections, we present the framework for realizing the runtime model interpretation, followed by an explanation of the AGM meta-model and, finally, the details of how model interpretation is achieved.

## 3.1. AGM Framework Architecture

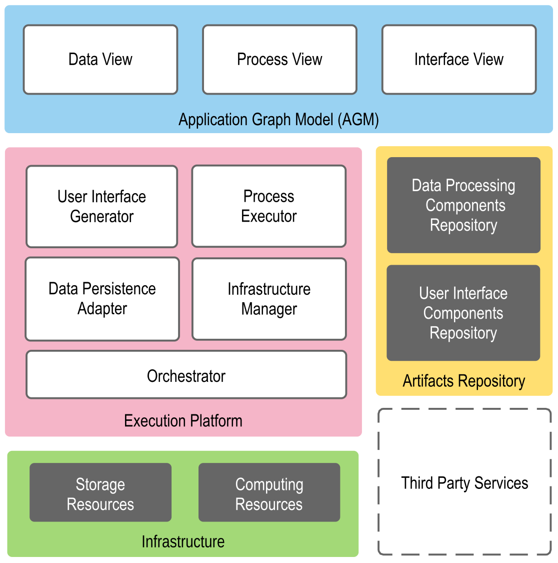


**Fig. 2.** AGM Framework Architecture

The AGM framework architecture for runtime interpretation is presented in Figure 2. The framework consists of an *application graph model (AGM)* interpreted by the *execution platform* that references components available at *artifacts repository*.

The *Application graph model (AGM)* provides a set of notations by which an end-user developer can specify the application models. An information system application model is constructed by using the *Data view*, *Process view* and *Interface view* notations in a holistic model. A *Data view* involves defining runtime invariant data structures, while the *Process view* and *Interface view* are assembled using constructs that reference artifacts stored in an artifact repository. These models respectively represent the graphical user interface (GUI) and the data-flow of the information system.

The *Execution platform* interprets the AGM model. It consists of a *User Interface Generator*, a *Process Executor*, a *Data Persistence Adaptor*, an *Infrastructure Manager* and an *Orchestrator*. The *User Interface Generator* interprets the part of the AGM model that defines GUI elements within the *Interface* view. The *Process Executor* interprets the information system application model and orchestrates the data-flow components available in the *artifacts repository*. The data-flow components are connected with the GUI components to represent information system data. The *Data Persistence Adapter* provides permanent data storage functionality based on structures defined in the information system application within a *Data view*. The *Infrastructure Manager* provisions the infrastructure resources required for the interpretation processes and data processing components. It ensures that non-functional requirements such as application performance are met by providing sufficient resources. Finally, the *Orchestrator* binds all the above components and switches control according to the current system state.

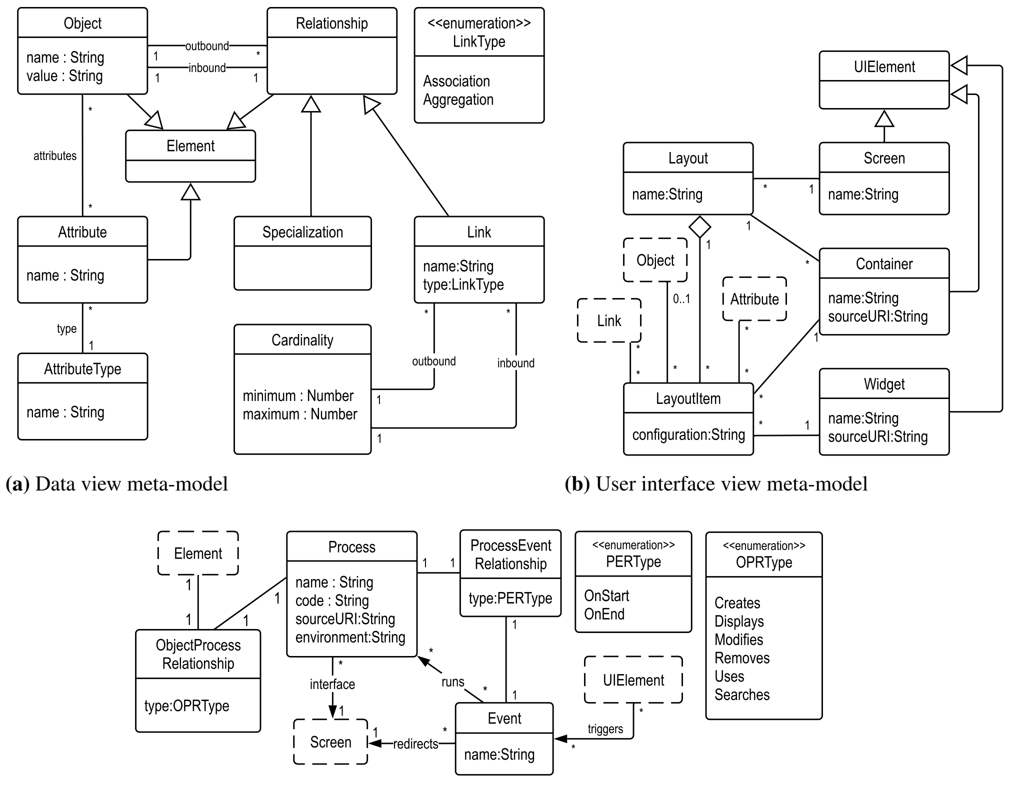
The *Artifacts repository* contains a set of reusable artifacts (generic parts of an information system application). These components are divided into *user interface*-related and *data processing*-related functionality. The *User interface* components represent application-specific information through a set of user interface (UI) elements that enable interaction with end-application users. Note that there could be several different types of UIs depending on the client platform (e.g. smartphone, tablet, or personal computer), and these components can be specialized for specific client platforms. *Data processing* components are used for data management. They provide real-time integration with external systems for persistent data storage.

The *Infrastructure* resources involved in provisioning information system applications can be divided into *computing* resources and *storage* resources. The *computing* resources provision the CPU-intensive parts of information systems such as data processing, while the *storage* resources are specialized for data management.

## 3.2. AGM meta-model

Figure 3 displays an AGM meta-model we developed for modeling information system applications. The AGM meta-model is extensible because of its reflexive nature—it completely defines itself [12,16]. This characteristic enables extending AGM capabilities beyond the three views described in the following subsections. For example, additional views could be defined for other types of interfaces (e.g., special resources and domain-specific devices). The meta-model is defined using UML class model semantics [17].

**Data view** A *Data view* defines the data structures from which different static artifacts can be derived (e.g. the information system database schema, web service interfaces, ...). It is derived both from the *Structure - Classes* specification contained within UML Superstructure model [17] and the OPM meta-model [13].



(a) Data view meta-model

(b) User interface view meta-model

(c) Process view meta-model

**Fig. 3.** AGM Meta-model

Each entity represented within the data structure is represented by an *Object* construct. Each *Object* can have multiple *Attributes* and *Relationships* to other objects. A *Relationship* can be a *Specialization*, representing attribute and link inheritance (as in object-oriented programming) or a *Link* representing connections between objects. The *Cardinality* quantifies the minimal and maximal number of *Objects* connected by *Links*. Unbounded cardinality is achieved by omitting a *maximum* quantity in the *Cardinality* object.

**User interface view** A *User interface view* defines a GUI used to manage the information and conduct business processes defined in a *Process view*. User interface generation combines information stored in the AGM with the current application context (e.g. currently running processes) and composes the graphical user interface using a reusable set of components, referred to as *Widgets* [18]. *Widgets* can be connected either to whole *Object*s or to their *Attribute*s and *Link*s through *LayoutItem*s which serve to select and order the displayed data. For example, *table-like* widgets can display entire *Object* instances, whereas *text-input* widgets display only certain *Attribute*s. A set of *LayoutItem* objects comprises a *Layout* that can be placed directly into an application *Screen* or into a *Container*. A *Container* is a user interface component used to organize information on *Screen*s, e.g. a *tabular* form with many *tabs*. The location of the source code of the *Widget* implementation is contained within *sourceURI* attribute.

**Process view** A *Process view* represents the modeling concepts used to implement dynamic application aspects. A *Process* either models a user activity within the application (which usually corresponds to real-life business process - or activity we wish to electronically document) or a background task that does not require user interaction (e.g. calculations and connection to remote services). A process can also run additional processes, all of which are contained in the application context within a *process stack*.

Every instantiated *Process* node is linked with an *Element* (*Object*, *Attribute* or *Link*) on which it operates through an *ObjectProcessRelationship*. The nature of this relationship is denoted by an *OPRType* enumeration: *Process*es can create, display, use, modify, remove and search for *Elements*. *Process*es can also emit *Event*s, which are represented by *ProcessEventRelationships* that can be a *PERType* (triggered on process start or end). User interface components or *Widget*s can also trigger *Events*, usually as a result of user interactions. Throughout the user interfaces, processes that involve end-application users are represented by the *Screen* constructs from an *Interface view*. These processes collect and display information to/from end-users through a series of user interface forms. In contrast, processes that are not linked with *Screen*s are considered as background processes. *Process* nodes can either directly represent generic components through their *sourceURI* attribute or contain action scripts used for expressing additional functionalities stored in a *Code* attribute. Currently, our proposal assumes that action scripts are written in an internal domain-specific language (DSL) [19], namely JS-DSL, that runs on top of existing procedural code that is also interpreted at runtime.

For the purposes of our evaluation case-study, the mobile cloud IS, or *Processes*, have an additional attribute intended to specify the *environment* in which they should be executed, which can be local (on the client side) or remote (in the cloud). Remote process execution is vital for maintaining a consistent and secure system since the scripts at the front-end are exposed and easy to manipulate.

## 3.3. AGM representation

To describe the methods and algorithms for interpreting the AGM, we first define it using a *directed property graph*. AGM thus becomes an ordered quadruple consisting of edges, vertices and mapping functions denoting their type (class from meta-model):

$$G_{AGM} = (V, E, T_E, T_V)$$

where $V$ is a set of vertices $V = \{x_1, x_2, x_3, \ldots, x_n\}, x_i \in \mathcal{D}$, and $\mathcal{D}$ represents a set of concepts from the modeled domain (e.g. *customer*, *product*, ...); whereas $E \subseteq X \times X$ represents a set of ordered vertex pairs $E = \{(x_i, x_j) | i \neq j, x_i, x_j \in E\}$ that represent the edges (links) from $x_i$ to $x_j$. Mapping functions are defined as $T_E : E \to \mathcal{M}_E$, which represents the node types in the AGM meta-model, and $T_V : V \to \mathcal{M}_V$, denoting the vertex types. According to the AGM meta-model defined in the previous section,

$$\mathcal{M}_E = \{Object, Process, Event, Layout, Screen \ldots\} \tag{1}$$

$$\mathcal{M}_V = \{Association, Aggregation, Inheritance, Uses, Modifies, \ldots\} \tag{2}$$

$Object$ nodes are the primary building blocks for the structural aspects of a system (*Data view*). Node types such as $Process$ and $Event$ represent dynamic (*Process view*) behaviors and node types such as $Layout$ and $Screen$ represent *User interface views*. AGM provides full modeling capability for the structural aspects of a system, including schemes for data storage and user interface definitions. The structural model is complete, meaning that the interpreter can execute them without requiring additional programming code to render database schema [20] or augment the user interfaces. For dynamic system definition, the AGM is supplemented by action scripts contained in $Process$ nodes, meaning that an additional mapping exists from each $Process$ node to executable artifacts or source code, if required.

The biggest advantage of representing models as holistic graphs is that we can use all the well-known concepts from graph theory, including graph traversal, graph matching, and querying for subgraphs. Our graph model interpreters use such capabilities to efficiently and reliably execute end applications.

We will demonstrate runtime traversal in a *Data view*. The Algorithm 1 is used to build a subgraph with the structural model defined for the requested *Object*. The input to the algorithm is the node that represents a certain *Object*, for which a complete list of attributes and links is extracted by traversing the AGM graph. The resulting subgraph can then be used to enable *Object* serialization (e.g., XML or JSON), or to generate SQL-language data manipulation queries to communicate with underlying databases.

```
Data: $G_{AGM} = (X, V)$, and starting node $x_s$ where $T_E(x_s) \in \{Object\}$
Result: $G'_{AGM} = (X', V')$ where $X' \subseteq X$ and $V' \subseteq V$ represent the complete structural
        model of concept $x_s$
begin
    $X' \leftarrow \{x_s\}, V' \leftarrow \emptyset$
    for $x \in X'$ do
        $S_{successors} \leftarrow \{x_j | x_j \in \Gamma^+(x), T_V(x, x_j) = Specialization\}$
        $X' \leftarrow X' \cup S_{successors}$
        for $s \in S_{successors}$ do
            $V' \leftarrow V' \cup (x, s)$
        end
    end
    for $x \in X'$ do
        $S_{successors} \leftarrow \{x_j | x_j \in \Gamma^+(x), T_V(x, x_j) \in$
          $\{Association, Aggregation, Attribute\}\}$
        for $s \in S_{successors}$ do
            $V' \leftarrow V' \cup (x, s)$
        end
    end
end
```

**Algorithm 1:** Traversal of subgraph containing the structural definition of a modeled concept

A similar concept is applied to extract interface compositions and their relationships from $Screen$, $Layout$ and $Widget$ nodes, and to conduct processes defined by $Process$ and $Event$ nodes.

## 3.4. Interpretation principles

Model interpretation consists of three interpreters, a server-side interpreter (SSI), a client-side process interpreter (CSPI), and a client-side user interface interpreter (CSUII) as depicted in Figure 4. The SSI serves as the *Process Executor* and is responsible for executing defined processes on the server side and for creating web service endpoints for bindings with client-side interpreters. On the client side, the CSPI serves both as a *Process Executor* and as an orchestrator for conducting processes locally in the client environment. The CSUII is a *User Interface Generator* used to render user interface (UI) elements. AGM models, when interpreted, are stored as graphs (see Section 3.3) in a *Graph DB*. To minimize the communication between client and server, and to enable the application to work in situations when the client is disconnected from the network, we also implemented a local database (*Local DB*) that contains serialized portions of the AGM model and application data.

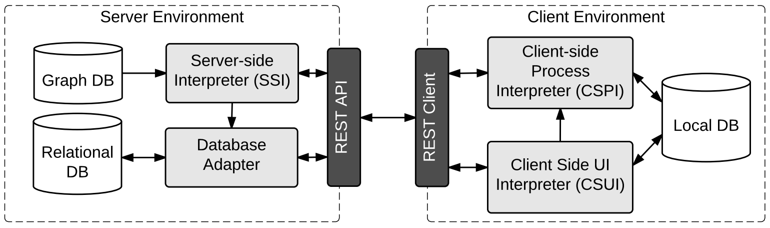

Server Environment
Graph DB
Server-side Interpreter (SSI)
Relational DB
Database Adapter
REST API
REST Client
Client Environment
Client-side Process Interpreter (CSPI)
Local DB
Client Side UI Interpreter (CSUI)

**Fig. 4.** A high-level view of the server-client architecture for interpreting the AGM

The server side includes a *Database Adapter* module to enable access to relational databases for manipulating data in persistent storage according to a defined *Data view* structure in the AGM. This module implements a *Data Persistence Adapter* from the framework presented in Figure 2. Because our approach supports runtime changes, holding relational schema solely within the *Relational DB* hinders instant adaptations to new models. Instead, every model change in a *Data view* yields incremental relational schema updates. The *Database Adapter* analyses model changes and issues updates to relational schema. However, because schema updates are a sensitive process that may result in data losses, it is possible to turn off automatic schema changes and rely on a semi-automatic approach after the initial application release.

Runtime model interpretation makes it possible to link stored data with a specific AGM model version. Multiple AGM model versions can be stored in *Graph DB* due to the continuous evolution of the modeled system. As an information system evolves from version to version, data structure mismatches may be introduced between the older and newer model versions.

The interpretation process transforms an AGM model into an information systems application at runtime. The resulting information system application may include a number of UI forms (we will use the term *application screens*), where users process presented information and decide on their next action. We refer to presentation and interaction between application users and *application screens* as *interpretation cycles*.

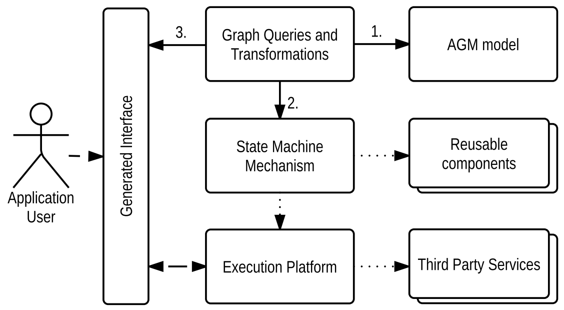


**Fig. 5.** AGM interpretation cycle

Figure 5 shows the steps that the AGM interpreter must take to execute each interpretation cycle. Each cycle begins by querying the AGM model for a definition of the UI and a set of possible user actions (Step 1). This definition is used to construct a state machine (SM) used to implement communication with the user (Step 2). Finally, the UI is generated for the application user (Step 3). Figure 6 shows a simplified version of the states and transitions for an SM. The SM is initialized to a start state that triggers UI rendering (*application screen*); then, it enters a *wait-for-user* state. For each $Process$ node and connected $Event$, a transition and a new state is added to SM and can be executed.

The algorithm 2 shows how a state machine is generated for each interpretation cycle. To provide user interaction for each *application screen*, the CSUII traverses AGM model from the *Screen* nodes in a *User interface view* to all connected *Process* and *Event* nodes that serve as input for defining state machine states and transitions. Three default states are always present: (1) an $s_{RI}$ state in which the user interface is rendered, (2) a $s_{WU}$ state that represents user *think time*, and (3) $s_{NC}$, which is a final state that occurs when the CSPI makes a switch to the next interpretation cycle. Additional states are defined for each process obtained while traversing the AGM. Transitions correspond to incident $Event$ nodes that trigger those processes.

The UI is constructed according to a *User interface view* in the AGM. It consists of multiple *Screen* nodes that define the appearance of each end-application UI component. Each *Layout* specifies a mapping between object attributes and widgets. It is important for widgets to be developed using *generic programming* approaches, allowing them to serve as templates that display different information based on linked $Attribute$s from $Object$s. We provide different *Widget* types according to the cardinality between an $Object$ and its $Attribute$ or $Link$.

```
Data: $s_{RI}$ - render interface state, $s_{WU}$ - wait for user state, $s_{NC}$ - next cycle state, and
      $P_e = \{(p, e)\}$ - set of process-event pairs from AGM for current cycle,
      $p \in P, e \in E$
Result: generated state machine $(\Sigma, S, s_0, \delta, F)$ where $\Sigma$ is set of transition events, $S$ is
        set of possible states, $s_0$ is a start state, and $\delta$ is a set of transitions $\delta : S \times \Sigma \rightarrow S$
begin
    $s_0 \leftarrow s_{RI}, F \leftarrow \{s_{NC}\}$ $S \leftarrow \{s_{RI}, s_{NC}, s_{WU}\}, \Sigma \leftarrow \emptyset$ $\delta \leftarrow \emptyset$
    for $(p, e) \in P_e$ do
        $S \leftarrow S \cup \{s_p\}$
        $\Sigma \leftarrow \Sigma \cup \{v_e\}$
        $\Sigma \leftarrow \Sigma \cup \{v_{finish}\}$
        if isInterfaceTriggered(e) then
            $\delta \leftarrow \delta \cup \{\delta(s_{WU}, v_e) \rightarrow s_p\}$
        end
        else if isDataTriggered(e) then
            $\delta \leftarrow \delta \cup \{\delta(s_{RI}, v_e) \rightarrow s_p\}$
        end
        if redirects(p) then
            $\delta \leftarrow \delta \cup \{\delta(s_p, v_{finish}) \rightarrow s_{NC}\}$
        end
        else if updatesInterface(p) then
            $\delta \leftarrow \delta \cup \{\delta(s_p, v_{finish}) \rightarrow s_{RI}\}$
        end
        else
            $\delta \leftarrow \delta \cup \{\delta(s_p, v_{finish}) \rightarrow s_{UW}\}$
        end
    end
end
```

**Algorithm 2:** Constructing a cycle state machine from a set of processes and events linked to each application screen

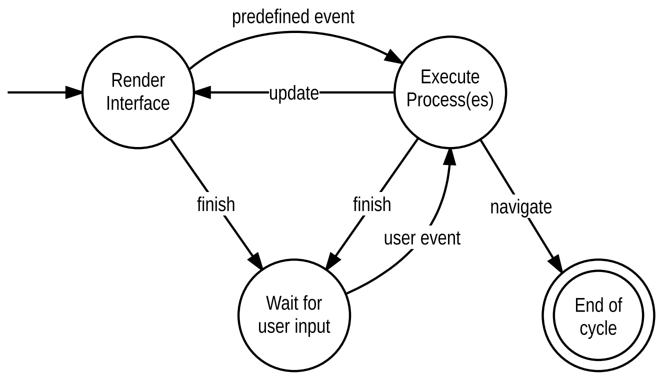


**Fig. 6.** A state machine is constructed for each *interpretation cycle*

## 4. Implementation and Evaluation

In this section, we describe the implementation of the presented framework for executing business applications in a mobile cloud context. IS applications contain numerous repeated patterns, which we have identified over the past ten years of professional software development[3] in the retail, supply-chain and merchandising domains. To automate the software development processes, we abstracted these patterns and now provide them as *generic components* through the AGM. Figure 7 shows an overview of our implementation. The *AGM interpreter* is divided into a *Mobile interpreter* and a *Cloud server interpreter*. The *Cloud server interpreter* additionally manages infrastructure resources using an *Infrastructure API* provided by the cloud provider. The main challenge in this type of implementation was the distributed nature of mobile cloud applications because it requires keeping the AGM models and application data synchronized in a distributed mobile execution environment.

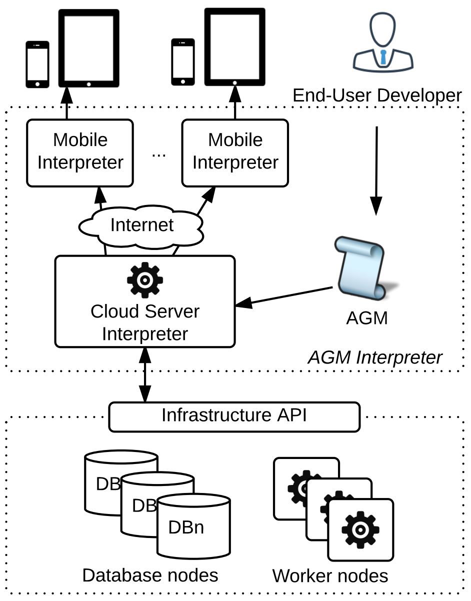


**Fig. 7.** AGM concept in the mobile cloud computing environment

Because AGM is a directed graph structure, we sought a semantically similar solution to manage it efficiently and reliably. We decided to use a graph database called Neo4j [21], which supports efficient queries because of its specialized graph-structure storage scheme. Neo4j provides a *property graph* model for storing data. This model allows each node and vertex to be associated with its own *key-value* data-store that can hold additional information. This capability was used to represent the mappings $T_E$ and $T_V$ from $G_{AGM}$

[3] One of the authors was associated with a Croatian software company *Superius d.o.o.*, dedicated to building mobile cloud information systems

as well as other additional values associated with certain model nodes (e.g., *Object* nodes contain names and values, and *Process* nodes can contain action script source code (see Figure 3).

Currently, mobile interpreters have been implemented for Android and iOS platforms using a hybrid development approach that combines native platforms with web application components [22]. The server-side interpreter is built as a Java application running on (but not limited to) *Apache Tomcat* application servers.

## 4.1. Data persistence

To manage stored data, we implemented $DatabaseAdapters$ for three relational databases: PostgreSQL, Oracle, and MSSQL. A generative tool is used to construct database schema from AGM models, allowing us to keep the database schema synchronized with the AGM models.

In the current implementation, the scalability and elasticity limitations of the relational databases are a drawback, since these relational databases do not typically provide elastic capabilities [23]. Our execution engine does not currently control database elasticity, hence the database components need to be provisioned according to planned workloads. Obviously, this is not something that end-user developer can achieve, and thus this steps requires specialized infrastructure personnel for on-premises usage of AGM. On the other hand, if one wishes to use public cloud providers, one can use a managed relational database such as Amazon RDS [4] or DigitalOcean PostgreSQL [5]. Achieving cloud-native automatically elastic persistence for AGM is a great future challenge, where we could also explore NoSQL solutions like document databases (e.g. MongoDB [6]).

## 4.2. Action Scripting Language Implementation

*Process* nodes from AGM are enriched with an action scripting language called *JS-DSL*. JS-DSL is an internal domain-specific language (DSL) developed on top of *JavaScript*. Figure 9 represents a block of JS-DSL code from one of our applications that sums up the total charges in an invoicing process. JS-DSL provides special constructs for accessing and manipulating user-level data. Figure 8 displays how JS-DSL can be used to customize user interfaces.

To create JS-DSL we followed the guidelines for creating internal DSLs proposed in [24], [25] and [19]. JS-DSL currently provides the following capabilities:

a) It can make runtime changes to UI widgets (e.g., emphasizing certain information using color, controlling widget behavior, and navigating through the application).
b) It can access data stored in background services (e.g., data persisted on a smartphone client or from remote servers),
c) It can apply a set of simple mathematical operators to data (e.g. sum, average, min, and max).
d) It can invoke remote third-party services.

[4] Amazon RDS, available at `https://aws.amazon.com/rds/`
[5] DigitalOcean cloud provider, available at `https://www.digitalocean.com/`
[6] MongoDB, available at `https://www.mongodb.com/`

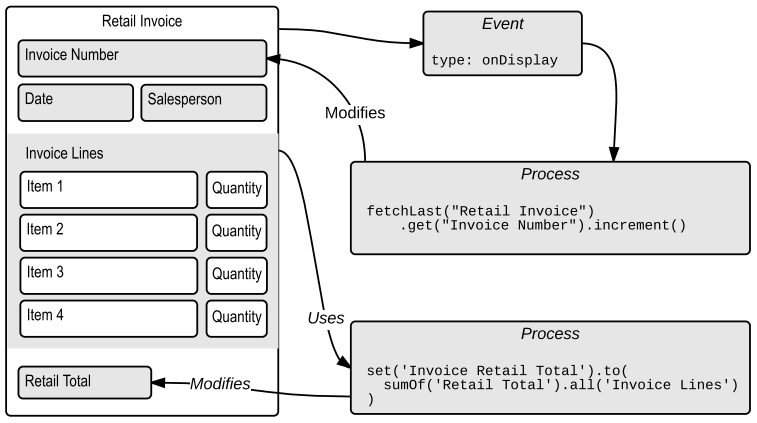


**Fig. 8.** Example of *Process* nodes combined with JS-DSL in specifying custom application behavior

```
set('Invoice Retail Total').to(
  sumOf('Retail Total').all('Invoice Line')
)
set('Lines Count').to(
  all('Invoice Line').count()
)
```

**Fig. 9.** A sample of JS-DSL source code for a *Process* node.

When desired actions are not available within JS-DSL, designers can rely on classical JavaScript code, which exploits the benefits of adopting a DSL embedded in the underlying JavaScript language. Note that such extensions require a more advanced user skill set.

## 4.3. Defining AGM models

To construct application models and load them to an execution platform we provide an additional DSL, called AGM-DSL. AGM-DSL is a one-to-one textual representation of an AGM model. Each model view has associated AGM-DSL commands (e.g., a *Data view* is associated with a *DEF* command). A BNF specification for a *DEF* command is presented in Figure 10, and Figure 11 shows an example *DEF* command used in an application from the retail domain.

## 4.4. Reusable components

Client-side user interface interpreter (CSUII) interprets the AGM model and composes the user-interface using the reusable generic components - widgets. Widgets are designed to be the gatekeepers of complexity towards the end-user developers. They are engineered

$$\begin{aligned}
\langle \text{define statement}\rangle &\models \texttt{DEF}\ \textit{object}\ \langle\text{inheritance}\rangle\ \langle\text{newline}\rangle\ \langle\text{attribute list}\rangle\\
\langle\text{attribute list}\rangle &\models \langle\text{attribute}\rangle\ \langle\text{newline}\rangle\ \langle\text{attribute list}\rangle \ \mid\ \langle\text{attribute}\rangle\\
\langle\text{Inheritance}\rangle &\models\ :\ \textit{inheritedObject}\\
\langle\text{attribute}\rangle &\models \langle\text{tab}\rangle\textit{attribute name}\ :\ \textit{attribute type}\ \langle\text{card}\rangle\ \langle\text{nl}\rangle\\
\langle\text{card}\rangle &\models \langle\text{quantity}\rangle..\langle\text{quantity}\rangle \ \mid\ \langle\text{quantity}\rangle\\
\langle\text{quantity}\rangle &\models \textit{numeric value}\ \mid\ \star\\
\langle\text{nl}\rangle &\models \backslash n\\
\langle\text{tab}\rangle &\models \backslash t
\end{aligned}$$

**Fig. 10.** BNF specification for DEF command in AGM DSL

```
DEF Product :Resource
   Name: Name 1..1
   Unit Of Measure: Unit Of Measure 0..1
   Wholesale Price: Number 0..1
   VAT: Number 0..1
   Retail Price: Number 0..1
   Stock: Number 0..1
   Stock Date: Date 0..1
   Code: Number 0..1
   Weight: Number 0..1
   Tax: Number 0..1
   Package Weight: Number 0..1
   Pallet Weight: Number 0..1
   Barcode: Number 0..1
   Package Quantity: Quantity 0..1
```

**Fig. 11.** Excerpt from the AGM-DSL defining a product in a retail domain

with classical software engineering methods by professional teams to conform to the predefined component specification. AGM achieves extensibility through the development of new widget components paired with their meta-models - connection points to the rest of the AGM model. Some widgets like `InputTextWidget` have a single connection point (e.g. the attribute of the object that needs to be provided by the user), while some widget can have multiple connection points (e.g. `LabelWidget` can represent multiple objects' attributes).

Widget component-model interface is derived from the *port-based interface* component-model [26]. The deviation from port-based interfaces is the introduced connection between components and the meta-data. This enables turning generic components into specific representations based on the context. The widget interface is displayed in Figure 12. Each widget implementation is complemented with an AGM node inherited from a $Widget$ node. The meta-data connection within $I_M$ are referencing the *data-view* elements (Fig. 3a) refined through *LayoutItem* nodes. The data that flows through data-

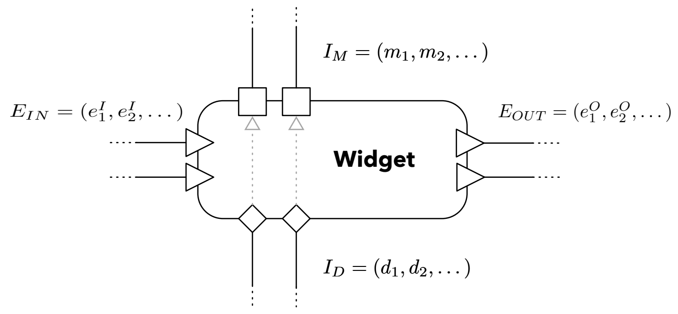

$I_M = (m_1, m_2, \dots)$
$E_{IN} = (e_1^I, e_2^I, \dots)$
$E_{OUT} = (e_1^O, e_2^O, \dots)$
Widget
$I_D = (d_1, d_2, \dots)$

**Fig. 12.** Interface for UI components - Widgets

interface $I_D$ conforms to *data-view* elements from $I_M$. The set of input and output events ($E_{IN}$, $E_{OUT}$) is also linked to *Event* nodes from the *process-view* model (Fig. 3c).

Currently, widget components are implemented in JavaScript, HTML, and CSS and executed strictly on the client-side, at the user-interface level. Since our mobile application engine is implemented as a hybrid web application using PhoneGap [27], all widgets are both working on Android, as well as iOS smart-phones. By using PhoneGap, we can provide an embedded web browser (e.g. *Android WebView*) as an integral part of the client application engine such that JavaScript interpreter is always available and enabled.

## 4.5. Performance considerations

Implementing interpreters with solid execution performance is challenging; it took us four developer-years to develop the proposed framework and obtain a satisfying user experience from a performance perspective (e.g. application load-time, GUI rendering-time, and communication-time). We encountered issues with the limited computing power of smartphones, resulting in slow UI rendering and slow execution of defined *Process* nodes. We compared AGM-modeled applications with previous generation non-modeled applications and noticed severe performance degradation. We tracked the main performance bottlenecks in the preparation process for executing the code contained in *Process* nodes and added *caching* mechanisms that stored the state machine specifications for each interpretation cycle as well as generated user interface code. After introducing such mechanisms, the interpreted model exploits performance benefits previously available only to generative approaches; the *cached* data represents *compiled* fragments of AGM models. *Caching* introduced drastic performance improvements to the initial prototype and reduced the performance penalty to only 15–20% compared to classically developed non-modeled applications. Our approach to the AGM model interpretation can be considered as a form of *just-in-time* (JIT) compilation.

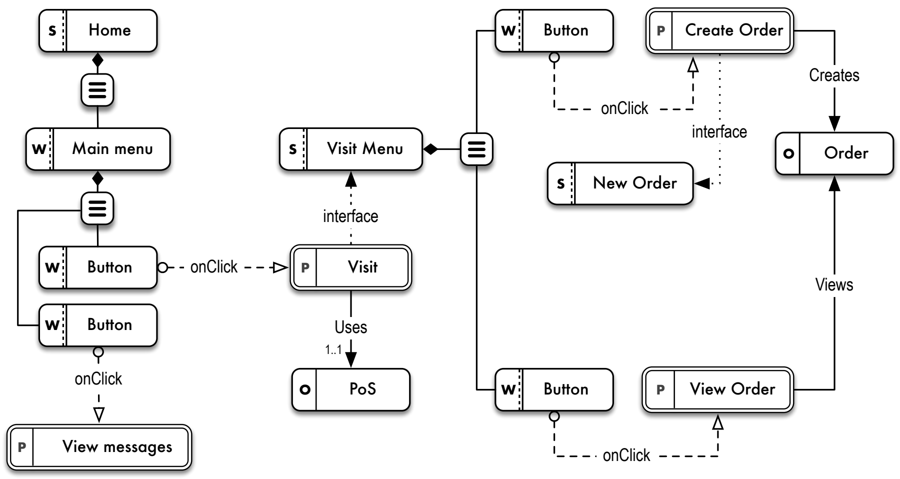
## 4.6. Reference application

In order to better illustrate the applicability of the AGM framework, we will disseminate an AGM model example and the resulting mobile application derived from it. The application presented is a subset of the typical application in the domain of supply chain management. The application is used by the field operatives from the distribution company in the supply chain process that are visiting end distributors (noted *PoS* - Point of Sales) and collecting orders for products that should be distributed. Orders that are collected are sent through the Internet and stored in the central database where the integration modules are used to transfer those orders into existing systems. The integration components are currently not provided by the AGM system due to the vast differences between different ERP vendors.

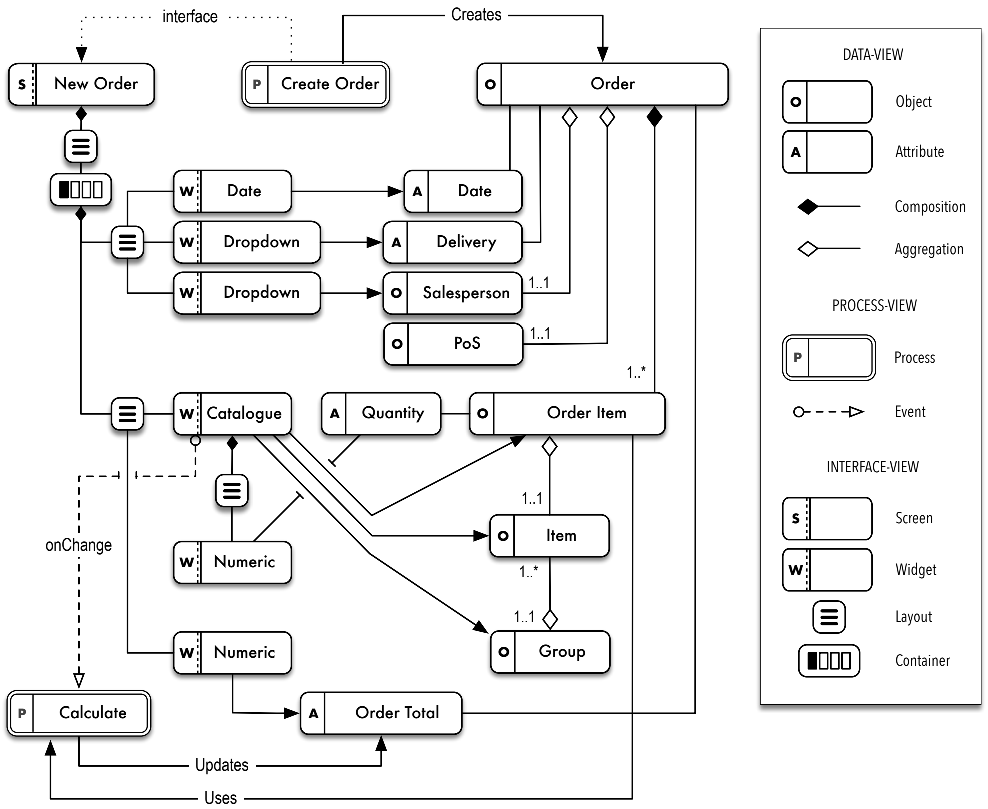
Figure 13a displays the main part of the AGM application model from the *process-view* and *interface-view* perspective. We can observe that the application consists of four main processes: (a) *Visit* process - where the users select a Point of Sales and can create a new Order, (b) *Create Order* process for the actual user-input of a new order, (c) *View Order* process for displaying previous orders, and (d) *Messages* for internal communication between organization members (this process is not further disseminated for brevity). Figure 13b displays the details of the *Create Order* process including the elements of the user interface and their links to the objects and attributes of the order object. We can observe that the *PoS* object required no widget since it is automatically extracted from the context of the *Visit* process which requires *PoS* to be defined.

Figure 14 displays the resulting user interface screens:

(a) a *Home screen* which is rendered from the specification of the *Home* Screen element that contains the *Menu* widget,
(b) *PoS selection screen* which is not explicitly modeled but inferred from the fact that the *Visit* process that is selected requires a *PoS* object to be selected. The user interface for selecting a PoS instance is not specified so it is automatically derived.
(c) *Visit screen* which is specified with *Visit Menu* screen from the model,
(d) and (e) - *Order insertion* screens which are specified with the *Create Order* nodes and their connections with different *user-interface* and *data-view* nodes.

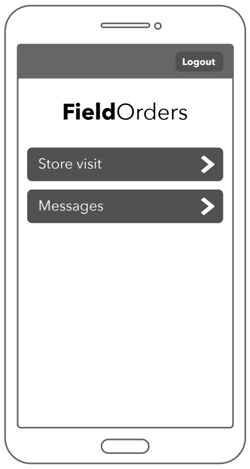


**(a)** The process and interface for navigating through application

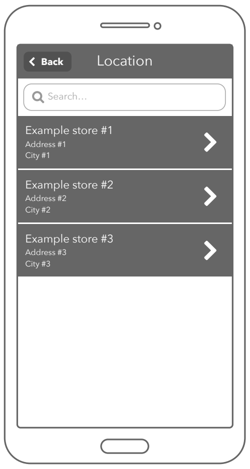


**(b)** The process for creating order

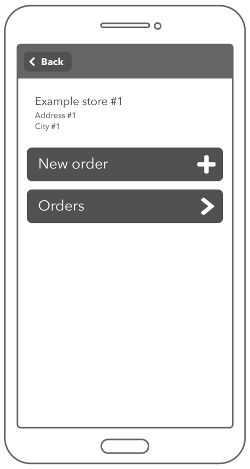
**Fig. 13.** The AGM source model for the example application



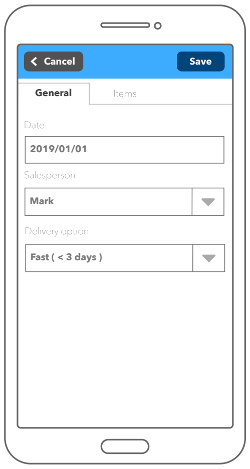
**(a)** Application home screen



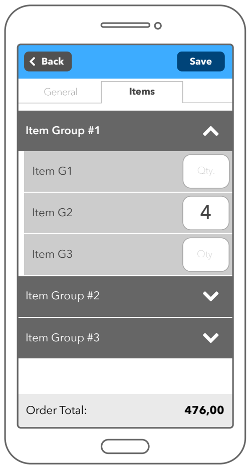
**(b)** Selecting a store to visit



**(c)** The menu with activities when a store is selected



**(d)** Inserting new order



**(e)** Selecting the items for the new order

**Fig. 14.** Example mobile cloud application in the supply chain domain.

## 4.7. Evaluation

We applied the AGM approach in implementing eleven projects for Southern European customers in the retail, supply chain management, and merchandising domains. The projects included building mobile information systems integrated with existing customer information systems. Combined, the projects involved over 300 end-application users with Android-based smartphones. In the retail domain, application functionality included collecting product orders and inspecting current stock levels in retail shops. The merchandising domain applications included conducting various surveys at points of sale to gain input on product quality, shelf placement, exposure metrics, and retail prices from competitive products. The sizes of the projects varied from 80 to 170 modeled entities, and from 8 to 27 modeled business processes. The largest project stored approximately 2–3 million transactions each month.

**Table 1.** A list of software development process improvements introduced by AGM

| | Before AGM | After introducing AGM |
|---|---|---|
| Requirements Definition | UML semantics were hard for our customers to understand | Visual application feedback on a graphical user-interface level |
| Implementation | Applications were implemented according to defined UML models. Software evolution meant additional effort for keeping the UML models in sync | The model itself is the implementation; implementation process leaves on implementing the specific JS-DSL action scripts which are integral to model itself |
| Verification and Validation | Each functionality point required a set of unit tests across all architecture layers (storing and representing information) | Unit tests are executed on a per predefined component level. New tests are required only when introducing new user-interface or data-processing components. |
| Distribution | Each new functionality point required the repackaging and redistribution of whole system. | New functionality points are introduced with new application model versions which are synchronized automatically upon application load. Repackaging and redistribution was required only when new modeling artifacts are introduced. |

By using AGM, our application development team achieved noticeable time-savings. Although we have not yet reached a point where our customers have been able to develop their own information system applications, we have enabled our software analysts to define nearly complete end-applications. The only point at which software engineers were required was when using the JS-DSL to fine-tune the models.

The main benefit was accrued from the quicker development cycle, which enabled visual exploration and negotiation while gathering requirements with our customers. Model interpretation enabled rapid visual feedback of the end system; thus, it allowed quicker convergence to the end-stage requirements. A more detailed set of improvements grouped by software development process phases is given in Table 1.

AGM also made the installation and distribution of end-applications application users easier because the users all shared the same mobile interpreter. After installation on the users' smartphones, the client interpreters loaded their AGM models from the associated cloud environment. This facilitated the *configuration management* process in the sense that we were able to reduce the number of client application versions and releases.

Software evolution also became easier because the interpreters always pull the latest model changes at run-time. The previous classically-built client applications binaries were over 10 MB in size which made remote transport and installation on each smartphone difficult because some users did not have solid, stable cellular network connections. This is a common issue in mobile cloud computing [28]. Introducing the AGM solution required synchronizing only new model versions, which was considerably faster. The AGM mobile interpreter occupies slightly more than 1 MB and is redistributed to users only upon new AGM meta-model element releases or when new reusable components are added. In comparison, the generative approach typically generates source code compiled the same or similar to classical methods, meaning that every model update requires a complete re-installation.

The downside of the AGM approach is in the large investment required to implement the distributed interpreter. A single software defect in the interpreter is usually manifested among all end-applications and solving such problems involves redistributing the interpreter to all end-application users. The greatest challenge was implementing an interpreter that was both fast and energy efficient enough for use in the mobile cloud domain. It was essential for end-application users to be able to use the application throughout an 8–10-hour business day without having to recharge their smart-phones or tablets. As mentioned earlier, just-in-time interpretation and caching are essential in achieving that goal.

## 5. Limitations

Our research demonstrates an end-user developer friendly framework for building information system applications in the cloud. There are some limitations that need to be considered both in terms of the system itself, and the way we have conducted the evaluation. We present these separately together with our current belief and future plans on how these limitations could be circumvented.

### 5.1. AGM limitations

There are two drawbacks to the current implementation of the AGM. The first drawback is using the textual AGM-DSL language in defining the data, process and interface structure. The users are required to learn the syntax of the language which is achieved by using existing examples of the language constructs and the way these are mapped to resulting applications. We are working on a visual representation of the language which follows the AGM meta-model. Visually, the language will resemble the visual representation used in Figure 13. This will also enable even faster visual feedback where the resulting application can be rendered side-by-side to the model itself.

The second challenge for end-users is using the action language embedded in Javascript which requires a basic knowledge of the JavaScript itself. This is a serious limitation and a significant learning step for an end-user developer. Although this language only uses a small subset of vanilla JavaScript in the form of basic control flow statements and data-structures (e.g. we require no regular expressions, higher-order functions, Document Object Model, modules&packages, callbacks, or closures), end-users have a significant problem to understand the basic concepts of programming. This design choice results in the fact that these scripts were needed to be additionally composed by software developers. While we can still report significant time savings in the development of the end applications, the development process cannot be 100% offloaded to end-user developers. We plan to build a visual representation language in order to specify the behavior needed. A good example we are considering is the *Blueprints* language[7] which should be adapted so that it fits our AGM meta-model.

### 5.2. Research methods limitations

There are also some threats to the validity of this research that also should be considered [29].

**Construct validity** There are many forms of mobile applications that can be built. Our approach currently targets the data-collection applications which complement the existing information systems. Currently, the AGM framework has limited support for the data transformation, analysis, and reporting of the higher-level data aggregations.

**Internal and external validity** Our study did not perform controlled experiments on the degree of usability in designing the applications compared to the classical methods. We plan to conduct these experiments once we complete the framework with visual modeling and action script programming parts.

**Conclusion validity** Based on that controlled experiments were not conducted, a more general conclusion on the applicability of our results to the general case of mobile software development cannot be reported with significant confidence. A full scale controlled experiment on a convincing number of different application domains is required.

## 6. Related Work

Information systems modeling has been well researched within the generative MDD field. Table 2 lists some of the well-established general-usage MDD tools, the majority of which follow a generative MDD approach. For building information systems, Milicev proposed an approach using an executable UML profile called *Object-Oriented Information Systems* (OOIS) [2]. After the models are compiled, they can be used in a special runtime Java-based environment. *SOLOist* is a tool based on OOIS that uses Java code for application customization.

Unlike OOIS, which uses Java, Popovic et al. [9] developed a DSL to specify application customization code at a *platform independent model* (PIM) level. Similar to AGM, PIM is targeted at information systems, and its approach is also generative but uses a pre-generated application interface and database. Dimitrieski et al. [30] also took a generative approach in their Multi-Paradigm Information System Modeling Tool (MIST) for building information systems through the simultaneous use of three different approaches.

[7] Blueprints language is used in the *Unreal Engine 4* engine, which is available at `https://www.unrealengine.com/en-US/`

The selected approach can thus depend on the problem domain and on the knowledge and personal preferences of an IS designer. MIST translates models to a relational data model or a class model.

MIDAS is a model-driven generative methodology proposed by Cáceres et al. [31] for developing web-based information systems. MIDAS is a specific application of Model-Driven Architecture (MDA) for Web platforms that uses XML and object-relational methodology. Currently, however, MIDAS provides only structural modeling of information systems.

Boyd and McBrien [32] also used graph structures for model representation. They proposed a *hypergraph data model (HDM)* structure for data model representation. HDM concentrates only on the data model of application and alleviates the need for model-to-model transformations used in previous generative approaches. AGM also uses a directed property graph, but unlike HDM, it is interpreted at runtime.

Many studies have emphasized end-user involvement in application development. Cappiello et al. [33] developed a UI-centric model that enables end-user developers to create *mashup* applications by applying WYSIWYG (what-you-see-is-what-you-get) specifications of data integration and service orchestration. They argued that user interfaces function as the medium most easily understandable by end-users. Vera [34] suggested that MDD methodology can be simplified by using a set of user interface components configured to define system behavior. Francese et al. [35] proposed an approach for model-driven development of portable applications based on a finite-state machine for specifying GUIs, transitions, and data-flow. Rivero et al. [36] proposed an MDD approach to capture requirements from end-users faster by using user interface prototypes that end-users completely understand. Garzotto [37] also promoted end-user development by proposing an approach that combined Model-Driven and End-user Development paradigms in modeling web applications in *cultural heritage* and *cultural tourism* domains.

There are few tools intended to perform model interpretation. Mendix, a commercial MDD tool, exploits runtime model interpretation for modeling web applications [38]; however, we are unaware of the internal details of the Mendix interpreter's operation because it is a closed-source commercial product.

The idea of directly executing UML models was introduced by Riehle et al. [11], who proposed a UML virtual machine; the biggest issue with this approach was that UML is too abstract to specify the behavioral aspects of applications. Following the work of Shlaer and Mellor [39], the OMG issued two important standards: an executable subset of the UML language called Foundational UML (fUML) [40] and the Action Language for Foundational UML (Alf) [41]. These standards enabled designers to create UML models with detailed behavioral specifications that could be effectively transformed into executable programs. This capability enables graphical specification of UML models supplemented by textual semantically related Alf code. Because fUML and Alf are novel specifications, few tools support them yet, especially tools targeted toward the IS domain. There has been some research proposing Alf transformation [10], but to best of our knowledge, no tools for interpretation of these standards yet exist, especially in the domain of modeling mobile cloud applications. However, because these standards are aligned with our proposal, we are exploring ways to integrate fUML and Alf when specifying AGM models.

**Table 2.** Some of available MDD Tools

| Product | Url |
|---|---|
| *Generative approach* | |
| WebRatio | `http://www.webratio.com` |
| WebML | `http://www.webml.org` |
| EMF | `http://www.eclipse.org/modeling/emf` |
| AndroMDA | `http://www.andromda.org` |
| IBM Rational Rhapsody | `http:ibm.com/software/awdtools/rhapsody` |
| OpenMDX | `http://www.openmdx.org` |
| MetaEdit+ | `http://www.metacase.com` |
| Cloudfier | `https://cloudfier.com/` |
| SOLOist | `http://www.soloist4uml.com/` |
| *Interpretative approach* | |
| Mendix | `http://www.mendix.com` |
| *Hybrid approach* | |
| System Vision | `http://www.mentor.com/products/sm` |
| OOA Tool | `http://ooatool.com/OOATool.html` |

## 7. Concluding Remarks

In this paper, we proposed an approach that enables faster development of information system applications. The developed models are then interpreted directly at runtime.

We presented an architectural framework for an Application Graph Model (AGM), which is used to model IS applications using generic components and an action scripting language contained directly within the model. Through model interpretation, we enabled run-time adaptations of modeled systems, resulting in faster prototyping and rapid software delivery.

Implementing the AGM framework in concrete industrial projects resulted in several improvements. We enabled software analysts and developers to cooperate in implementing information systems, which drastically improved requirements negotiation and reduced the team size to a single analyst and engineer. However, we did measure a 15–20% performance penalty, which is especially noticeable in smartphone execution environments. This performance penalty is due to the overhead associated with querying the model for interpretation and run-time interface generation.

We are also working on building a graphical modeling environment for end-user developers [42] that will increase their productivity and reduce errors. Spreadsheet-like software has amply demonstrated that the *what-you-see-is-what-you-get* concept is highly appealing; having a runtime interpretive model is the foundation for a similar solution when designing IS systems. Our future work will also include efforts to implement a graph analysis algorithm that could be used to propose optimally efficient cloud deployment strategies [43] based on operational data inspections. Using this approach, a cloud executor could save costs by dynamically reassigning computation tasks among heterogeneous cloud resources according to workload demands.

**Acknowledgments.** This work has been supported in part by Croatian Science Foundation's funding of the project UIP-2014-09-7945.

## References

1. Li, J., Rong, W., Yin, C., Xiong, Z.: Goal-oriented dependency analysis for service identification. Computer Science & Information Systems **16**(2) (2019)
2. Milicev, D.: Model-Driven Development with Executable UML. John Wiley & Sons (2009)
3. Harel, D., Marron, A.: The quest for runware: On compositional, executable and intuitive models. Software and Systems Modeling **11**(4) (2012) 599–608
4. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications **1**(1) (apr 2010) 7–18
5. Samad, J., Loke, S.W., Reed, K.: Mobile Cloud Computing. Cloud Services, Networking, and Management (2015) 153–190
6. Tankovic, N., Vukotic, D., Zagar, M.: Rethinking Model Driven Development: analysis and opportunities. In Luzar-Stiffler, V., Jarec, I., Bekic, Z., eds.: Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference on, SRCE (2012) 505–510
7. Hailpern, B., Tarr, P.: Model-driven development: The good, the bad, and the ugly. IBM Systems Journal **45**(3) (2006) 451–461
8. Stahl, T., Völter, M., Bettin, J., Haase, A., Helsen, S.: Model-Driven Software Development: Technology, Engineering, Management. (2006)
9. Popovic, A., Lukovic, I., Dimitrieski, V., Djukic, V.: A DSL for modeling application-specific functionalities of business applications. Computer Languages, Systems & Structures **43** (2015) 69–95
10. Ciccozzi, F., Cicchetti, A., Sjodin, M.: Towards Translational Execution of Action Language for Foundational UML. 2013 39th Euromicro Conference on Software Engineering and Advanced Applications (SEPTEMBER) (2013) 153–160
11. Riehle, D., Fraleigh, S., Bucka-Lassen, D., Omorogbe, N.: The architecture of a UML virtual machine. Environment **36**(11) (2001) 327–341
12. Tanković, N., Vukotić, D., Žagar, M.: Executable graph model for building data-centric applications. Proceedings of the International Conference on Information Technology Interfaces, ITI (2011) 577–582
13. Dori, D.: Object-Process Methodology: A Holistic Systems Paradigm; with CD-ROM. Volume 1. Springer Science & Business Media (2002)
14. Ma, Q., Kelsen, P., Glodt, C.: A generic model decomposition technique and its application to the Eclipse modeling framework. Software & Systems Modeling (2013) 1–32
15. OMG: Meta Object Facility$^{TM}$ (MOF$^{TM}$) Version 2.5 Specification (2015)
16. Reinhartz-Berger, I., Dori, D.: A Reflective Meta-Model of Object-Process Methodology: The System Modeling Building Blocks. Business Systems Analysis with Ontologies (2005) 130–173
17. OMG: OMG Unified Modeling Language (OMG UML) Superstructure (2010)
18. Nicolaescu, P., Klamma, R.: A Methodology and Tool Support for Widget-Based Web Application Development. In Cimiano, P., Frasincar, F., Houben, G.J., Schwabe, D., eds.: Engineering the Web in the Big Data Era. Volume 9114 of Lecture Notes in Computer Science. Springer, Cham (2015) 515–532
19. Fowler, M.: Domain-Specific Languages. Addison-Wesley Professional (2010)
20. Brdjanin, D., Banjac, D., Banjac, G., Maric, S.: Automated two-phase business model-driven synthesis of conceptual database models. Computer Science & Information Systems **16**(2) (2019)
21. Miller, J.: Graph Database Applications and Concepts with Neo4j. Proceedings of the 2013 Southern Association for Information Systems (2013) 141–147
22. Charland, A., Leroux, B.: Mobile application development. Communications of the ACM **54**(5) (may 2011) 49

23. Elmore, A.J., Das, S., Agrawal, D., El Abbadi, A.: Towards an elastic and autonomic multi-tenant database. In: Proc. of NetDB Workshop. (2011)
24. Freeman, S., Pryce, N.: Evolving an embedded domain-specific language in Java. Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (2006) 855–865
25. Kossakowski, G., Amin, N., Rompf, T., Odersky, M.: JavaScript as an embedded DSL. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7313 LNCS** (2012) 409–434
26. Crnkovic, I., Sentilles, S., Vulgarakis, a., Chaudron, M.R.V.: A Classification Framework for Software Component Models. IEEE Transactions on Software Engineering **37**(5) (2011) 593–615
27. Wargo, J.M.: PhoneGap essentials: Building cross-platform mobile apps. Addison-Wesley (2012)
28. Dinh, H.T., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Communications and Mobile Computing **13**(18) (dec 2013) 1587–1611
29. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting experiments in software engineering. Guide to Advanced Empirical Software Engineering (2008) 201–228
30. Dimitrieski, V., Čeliković, M., Aleksić, S., Ristić, S., Alargt, A., Luković, I.: Concepts and evaluation of the extended entity-relationship approach to database design in a multi-paradigm information system modeling tool. Computer Languages, Systems & Structures **44** (2015) 299–318
31. Cáceres, P., Marcos, E., Vela, B., Juan, R.: A MDA-Based Approach for Web Information System Development. Methodology
32. Boyd, M., McBrien, P.: Comparing and Transforming Between Data Models via an Intermediate Hypergraph Data Model. Journal on Data Semantics IV **4** (2005) 69–109
33. Cappiello, C., Matera, M., Picozzi, M.: A UI-Centric Approach for the End-User Development of Multidevice Mashups. ACM Transactions on the Web **9**(3) (2015) 1–40
34. Vera, P.M.: Component Based Model Driven Development:. International Journal of Information Technologies and Systems Approach **8**(2) (jun 2015) 80–100
35. Francese, R., Risi, M., Scanniello, G., Tortora, G.: Model-Driven Development for Multi-platform Mobile Applications. In Abrahamsson, P., Corral, L., Oivo, M., Russo, B., eds.: Product-Focused Software Process Improvement. Volume 9459 of Lecture Notes in Computer Science. Springer International Publishing, Cham (2015) 61–67
36. Rivero, J.M., Luna, E.R., Grigera, J., Rossi, G.: Improving user involvement through a model-driven requirements approach. In: 2013 3rd International Workshop on Model-Driven Requirements Engineering (MoDRE), IEEE (jul 2013) 20–29
37. Garzotto, F.: Enterprise Frameworks for Data Intensive Web Applications: An End-User Development, Model Based Approach. Journal of Web Engineering **10**(January) (2011) 87–108
38. Henkel, M., Stirna, J.: Pondering on the key functionality of model driven development tools: The case of mendix. Perspectives in Business Informatics Research **BIR 2010,** (2010) 146–160
39. Shlaer, S., Mellor, S.J.: The Shlaer-Mellor Method. (1996) 1–13
40. OMG: Semantics of a Foundational Subset for Executable UML Models (FUML) Version 1.1 Specification (2013)
41. OMG: Action Language For Foundational UML (ALF) 1.0.1 Specification (2013)
42. Tankovic, N., Galinac Grbac, T., Zagar, M.: Experiences from building a EUD business portal. In: 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE (may 2014) 551–556
43. Tanković, N., Galinac Grbac, T., Truong, H.l., Dustdar, S.: Transforming Vertical Web Applications Into Elastic Cloud Applications. In: International Conference on Cloud Engineering (IC2E 2015), IEEE (mar 2015) 135–144

**Nikola Tanković** is a postdoctoral researcher at the Juraj Dobrila University of Pula. His main research interests are directed to model-driven development of information systems, quality optimisation of distributed systems, and generally black-box model optimisation using soft computing and simulation. He is involved in several industry projects in developing predictive machine-learning models and web services in cloud.

**Tihana Galinac Grbac** is a full professor of computer science and the head Software Engineering and Information Processing Laboratory (SEIP Lab) at the Juraj Dobrila University of Pula. Her main research interests are related to large scale and complex software systems that are evolutionary developed. In a broader sense, she is also interested in a variety of large scale complex systems including smart cities, telecommunication networks and others. She is actively involved as the leader, management committee member and researcher in a number of research projects funded by European Union, Croatian government or industry partners. The results of her work are continously published in international scientific journals and conferences.

*Received: October 11, 2018; Accepted: September 17, 2019.*

Computer Science and Information Systems 17(1): 29–50 https://doi.org/10.2298/CSIS181122036A

# Efficient Virtual Machine Placement Algorithms for Consolidation in Cloud Data Centers

Loiy Alsbatin[1,2], Gürcü Öz[1], and Ali Hakan Ulusoy[3]

[1] Department of Computer Engineering, Faculty of Engineering, Eastern Mediterranean University,
Famagusta, North Cyprus via Mersin 10 Turkey,
loiy.alsbatin@gmail.com, gurcu.oz@emu.edu.tr
[2] Department of Computer Science, Collage of Computing and Information Technology, Shaqra University,
Riyadh, Saudi Arabia
[3] Department of Information Technology, School of Computing and Technology, Eastern Mediterranean University,
Famagusta, North Cyprus via Mersin 10 Turkey,
alihakan.ulusoy@emu.edu.tr

**Abstract.** Dynamic Virtual Machine (VM) consolidation is a successful approach to improve the energy efficiency and the resource utilization in cloud environments. Consequently, optimizing the online energy-performance tradeoff directly influences quality of service. In this study, algorithms named as CPU Priority based Best-Fit Decreasing (CPBFD) and Dynamic CPU Priority based Best-Fit Decreasing (DCPBFD) are proposed for VM placement. A number of VM placement algorithms are implemented and compared with the proposed algorithms. The algorithms are evaluated through simulations with real-world workload traces and it is shown that the proposed algorithms outperform the known algorithms. The simulation results clearly show that CPBFD and DCPBFD provide the least service level agreement violations, least VM migrations, and efficient energy consumption.

**Keywords:** Cloud computing, energy consumption, dynamic consolidation, virtualization

## 1. Introduction

Dynamic Virtual Machine (VM) consolidation effectively improves the energy efficiency and resource utilization in data centers. Reallocating VMs from an overloaded Physical Machine (PM) maximizes the utilization and energy efficiency with providing a high Quality of Service (QoS). The goal of consolidation of VMs ensures an efficient utilization that can be achieved through the use of VM migrations across different PMs. Power consumption of USA data centers has increased by 62.5% from 2005 to 2013 and expected to increase by 150% in 2020 [1]. Most of the energy consumption of data centers is consumed by computing resources. Accordingly, resource management is important to ensure that the applications efficiently utilize the available computing resources. Switching the idle nodes to sleep mode to eliminate the idle power consumption can achieve a reduction in energy consumption.

One efficient way to improve the utilization of cloud data center resources is the dynamic consolidation of VMs [2-11]. The dynamic consolidation reallocates VMs periodically using migration to reduce the number of active PMs required to handle requests. The objective of this approach is mainly to minimize energy consumption and maximize of QoS provided by the system.

It is complex to solve dynamic VM consolidation problem analytically as a whole [3, 4]. In general, the problem can be decomposed into tasks as following [8]:

1. PM underload detection: This is the phase when a PM is considered as being underloaded, so all VMs running on an underloaded PM should be migrated to other PMs and the underloaded PM should be switched to the sleep mode (to reduce the number of active PMs).
2. PM overload detection: This is the phase when a PM is considered as being overloaded, so some VMs running on an overloaded PM should be migrated to another active PM (to avoid violation QoS requirements).
3. VM selection: This is the phase to select VMs to be migrated from the overloaded PM.
4. VM placement: This is the phase to place selected VMs for migration on another active PM.

In this study, we mainly focus on VM placement problem. Algorithms named as CPU Priority based Best-Fit Decreasing (CPBFD) and Dynamic CPU Priority based Best-Fit Decreasing (DCPBFD) are proposed for VM placement. We implemented a number of placement algorithms to compare with the proposed algorithms using real workload traces.

The rest of the paper is organized as follows. The related work and the system model are discussed in Sections 2 and 3, respectively. The metrics used to show the performance of the algorithms are described in Section 4. The proposed VM placement algorithms are presented in Section 5. In Section 6, the experimental setup, evaluations and results are discussed. Finally, we conclude the results and discuss the future work in Section 7.

## 2. Related Work

The two main types of energy efficient resource management algorithms in the cloud data centers are constraint energy consumption algorithm [12, 13] and energy efficiency (energy consumption and Service Level Agreement (SLA) violation) algorithm [7, 8, 14, 15]. The constraint energy consumption algorithm aims to minimize the energy consumption, but this type of algorithm does not consider SLA violation at all or focuses a little on it. Therefore, this type of algorithm does not meet the requirements of users. For example, two heuristic algorithms are proposed by Lee and Zomaya [12] to reduce the energy consumption, but the algorithms do not consider SLA violation. Similarly, Kang and Ranka [13] proposed an energy-saving algorithm, but it also does not consider SLA violation. The main goal of the energy efficiency algorithm is to reduce the energy consumption and SLA violation in data centers. Several VM placement algorithms are proposed in [7, 8, 14, 15]. These algorithms reduce SLA violation and save energy consumption, but SLA violation remains at a high level. In our previous study [16], we proposed dynamic VM consolidation based on a PM overload detection algorithm and a combination of PM overload detection algorithm and VM quiescing to minimize number of VM migrations according to QoS requirements. The goal of the model is to improve utilization of resources, SLA, and energy efficiency in cloud data centers. However, this model does not focus on energy consumption.

In the past few years, many approaches to the dynamic consolidation of VMs have been proposed [2-11]. Comparative studies of various existing consolidation of VM algorithms using real-world workload traces were presented. Some of VM consolidation algorithms based on different heuristics on the legitimate PM were analyzed in [17]. A scheduling algorithm to assign VMs to PMs in a data center was proposed in [18]. The goal was to improve energy efficiency by taking into consideration the conflicts between the costs of VM migration and CPU and disk utilizations. Four models named as the migration model, the energy model, the application model, and the target system model were presented to identify the conflicts.

An adaptive threshold-based algorithm was proposed by Deng et al. in [19]. The overload threshold of CPU utilization and the average utilization of active PMs were used for PM underload detection algorithm, and minimum average utilization difference of the data center was used for VM placement algorithm. Several dynamic VM consolidation algorithms were proposed by Khoshkholghi et al. in [20] to improve the utilization, energy consumption and SLA violations based on the CPU, RAM and bandwidth. They used an iterative weighted linear regression method for PM overload detection and a vector magnitude squared of resources for PM underload detection. They also proposed SLA and power-aware VM selection algorithm and VM placement algorithm. PM overload and underload detection algorithms and VM placement algorithm based on dynamic thresholds and probable future load were proposed by Shaw et al. in [21]. They used simple exponential smoothing technique to predict CPU utilization and calculate dynamic upper and lower utilization thresholds. A VM consolidation algorithm with utilization prediction of multiple resource types based on the local history of PMs was proposed by Nguyen et al. in [22] to improve the energy efficiency of cloud data centers. Two adaptive energy-aware algorithms for minimizing SLA violation and maximizing energy efficiency in cloud data centers were proposed by Zhou et al. in [23]. CPU, memory resources and application types were considered during the deployment of VMs.

Managing resource allocation to improve response time using control loops at the server and cluster levels were applied in [24]. The server migrated a VM if the server's resource capacity was not enough to meet SLA of application. An adaptive heuristics energy-aware algorithm that used an upper threshold of CPU utilization for PM overload detection and dynamic VM selection algorithms was proposed in [25]. A greedy consolidation algorithm based on VM placement algorithm was proposed in [26] to improve the network usage and performance of applications in the data centers. The greedy consolidation algorithm reduced the number of migrations and speed up the placement decisions. In [27], two algorithms which could be used together for live migration of multiple VMs were proposed. The proposed VM migration depended on three factors that were the cost of migration, the expected distribution of workload and the state of PM after migration. The algorithms distributed the workload efficiently in the system. In spite of that, the research did not discuss how to meet SLA. In [3], the problem of dynamic VM placement was solved by a heuristic bin packing algorithm. However, SLA cannot be met because of unforeseeable workloads and instability.

A dynamic consolidation of VMs for web applications was implemented in [10]. In this study, the response time was used to define SLA. Weighted linear regression was applied to get the future workload and improve the distribution of workload. VM consolidation algorithms under QoS expectations were evaluated using the CloudSim toolkit showing high improvement of cost savings and energy efficiency using dynamic workload scenarios [7, 8]. They proposed maximum correlation, random selection and Minimum Migration Time (MMT) policies for VM selection from the overloaded PM and utilized interquartile range, median absolute deviation, robust local regression and Local Regression (LR) algorithms for PM overload detection. Simple Method (SM) was used to find underloaded PM which was with the least resource utilization. For VM placement, they proposed Power-Aware Best-Fit Decreasing (PABFD) algorithm, which based on sorting VMs by CPU utilization in decreasing order and placing a VM in PM that will have the minimum expected increasing in power consumption. The results showed that the combination of LR for PM overload detection and MMT for VM selection had better performance in the number of VM migrations, energy consumption, and SLA violations.

In this research, LR method was used for PM overload detection, SM policy was used for PM underload detection, MMT policy was used for VM selection, and for comparison purposes with proposed CPBFD and DCPBFD VM placement algorithms we used PABFD, First-Fit Decreasing (FFD) [28-30] and Best-Fit Decreasing (BFD) [29, 30] algorithms which are well-known algorithms for bin-packing problem. VM placement algorithm based on FFD sorts VMs by CPU utilization in decreasing order and places a VM in the first PM that will fit it [30]. VM placement algorithm based on BFD sorts VMs by CPU utilization in decreasing order and places a VM in PM that will have the maximum CPU utilization after allocating VM [30].

## 3. System Model

We use the system model presented in [8] to evaluate VM placement algorithms by using the CloudSim [31] toolkit. The system consists of *X* heterogeneous PMs in a large-scale data center. Characteristics of each PM are defined by CPU performance denoted by Random Access Memory (RAM), Millions Instructions Per Second (MIPS), and network bandwidth. The storage of servers for VM live migration is network attached storage. Multiple independent users request for supplying *Y* VMs characterized by requirements denoted by RAM, MIPS and network bandwidth.

As shown in Fig. 1, the system includes global and local managers. The local manager on each PM monitors CPU utilization of PM using VM Monitor (VMM). VMM decides when and which VMs should be migrated to other PMs. The global manager which acts as the controller in the system collects information of the utilization of PMs from the local managers and decides VM placement, and VMMs migrate VMs and change the power mode of PMs.

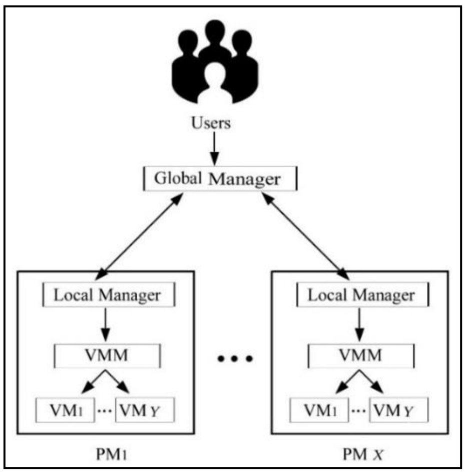


**Fig. 1.** The system model [8]

## 4. Metrices

### 4.1. Power Model

In the data centers, power consumption of PMs is usually defined by CPU, cooling systems, power supplies, memory and disk storage [32]. The power consumption of PMs can be defined using linear relationship of CPU utilization even if dynamic voltage and frequency scaling is used [7, 33, 34]. Due to the fact that the limited number of the frequency and voltage states of a CPU and other system components, such as network interfaces and memory, the voltage and frequency scaling are not used. Since analytical models of power consumption is a complex research problem for modern multi-core CPUs [8], real power consumption benchmark results provided by the SPECpower [35] are used.

The work in [7, 36] shows that a PM when it is idle uses approximately 70% of its maximum energy consumption. As presented in [7, 36], the power consumption of PM can be defined as

$$P(u) = 0.7 \times P_{max} + 0.3 \times P_{max} \times u \qquad (1)$$

where $P_{max}$ is the maximum power of a fully utilized PM and is set to 250 W as presented in [33, 34]. $u$ is CPU utilization. Since CPU utilization changes over time, it is presented as a function of time as $u(t)$. As presented in [7], energy consumption can be obtained as

$$\mathrm{E} = \int_t P(u(t))dt. \qquad (2)$$

Since the energy consumption of a PM is determined by CPU utilization, we take CPU utilization into consideration in the proposed VM placement algorithms to reduce the energy consumption in the system [6].

## 4.2. Cost of Live Migration of VMs

Live migration of VMs transfers VMs between PMs without suspension. However, the large number of live VM migrations may drop the performance of applications. So, the number of VM migrations should be reduced. The behavior of applications causes performance degradation. We use cost model for VM migration presented in [6] to avoid performance degradation. The authors stated in [6] that CPU utilization can be increased by 10% for each VM migration. So, each VM migration can cause SLA violations. Therefore, VM migrations should be reduced, and VM with minimum memory should be selected.

## 4.3. SLA Violation Metrics

In cloud computing environments, it is highly important to meet QoS requirements. QoS is usually defined in the form of SLA that is defined through some characteristics such as maximum response time or minimum throughput of the system [8]. To evaluate QoS requirements, SLA metric is defined as a workload independent metric for any loads in Infrastructure as a Service (IaaS). Two metrics are used to measure SLA violations: The fraction of time when CPU utilization of PM has been 100%, SLA violation Time per Active Host/PM (SLATAH) as shown in (3); and Performance Degradation due to Migrations (PDM) as shown in (4) [8],

$$\mathrm{SLATAH} = \sum_{i=1}^{X} \frac{T_{s_i}}{T_{a_i}} \tag{3}$$

$$\mathrm{PDM} = \frac{1}{Y} \sum_{j=1}^{Y} \frac{C_{d_j}}{C_{r_j}} \tag{4}$$

where $X$ represents the number of PMs, $T_{s_i}$ is the time when the utilization of $i$-th PM is 100% which leads to an SLA violation, $T_{a_i}$ is the time when i-th PM is active, $Y$ represents the number of VMs, $C_{d_j}$ is the estimated performance degradation caused by $j$-th VM migrations, $C_{r_j}$ is the total CPU capacity requested by j-th VM. $C_{d_j}$ is estimated to be 10% of CPU utilization in MIPS during the j-th VM migrations [8].

The level of SLA violations is independently characterized by both SLATAH and PDM metrics. So, we use a metric presented in [8] that includes performance degradation caused by both overloaded PM and VM migrations, denoted as SLA Violation (SLAV) that is calculated as

$$\mathrm{SLAV} = \mathrm{SLATAH} \times \mathrm{PDM}. \tag{5}$$

VM consolidation objective is to reduce both energy consumption (E) and SLAV. ESV metric presented in [8] that equals the product of energy consumption and SLA violations is used as

$$\mathrm{ESV} = \mathrm{E} \times \mathrm{SLAV}. \tag{6}$$

SLAV and energy consumption are the most important metrics that should be minimized to improve efficiency of resources [7, 8]. SLA violation has a negative relation with the energy consumption in the cloud data center [37]. Therefore, ESV is used for performance evaluation of all algorithms to show the trade-off between energy consumption and SLA violation [7, 8, 37].

## 5. VM Placement

We propose novel VM placement algorithms based on giving priority of PM with highest CPU utilization between two sided limits, then giving priority to PM with CPU utilization outside the two-sided limits and nearest to the two-sided limits. Second priority is given to PMs with CPU utilization outside the two-sided limits, since selecting PMs with low load or high load leads to less active PMs and less energy consumption than waking up PMs from sleep mode. We modified well-known BFD algorithm [29] to be suitable for VM placement by limiting the upper CPU utilization threshold [38-42] and lower CPU utilization threshold and implementing the abovementioned priority. We denote proposed algorithm that used static upper and lower CPU utilization thresholds as CPBFD, and proposed algorithm that used dynamic upper and lower CPU utilization thresholds as DCPBFD. Not setting an upper limit for the CPU utilization of allocated PMs may cause frequent overloading of allocated PMs, which leads to performance degradation and increases the number of VM migrations. We propose to limit the upper threshold of CPU utilization of allocated PM to avoid performance degradation caused by VM migrations to PM with high load and to minimize the number of VM migrations. Furthermore, not setting a lower limit for CPU utilization of allocated PMs may cause allocating underloaded PMs, which leads to more active PMs and more energy consumption. We propose to limit the lower threshold of CPU utilization of allocated PM to improve energy consumption.

### 5.1. CPBFD Algorithm

CPU resource utilization model of PM of CPBFD algorithm is shown in Fig. 2. In CPBFD, all VMs are sorted in the decreasing order of their current CPU utilizations and allocate each VM to a PM with maximum CPU utilization less than upper CPU utilization threshold *a* and more than lower CPU utilization threshold *b*. If there are no PMs with CPU utilization between upper and lower threshold, a PM with the nearest CPU utilization to upper or lower thresholds will be allocated. The priority may be given to PM with CPU utilization nearest to upper threshold or lower threshold by adjusting parameter *c*. PM with CPU utilization that nearest to *c* and outside of *a* to *b* range will be selected.

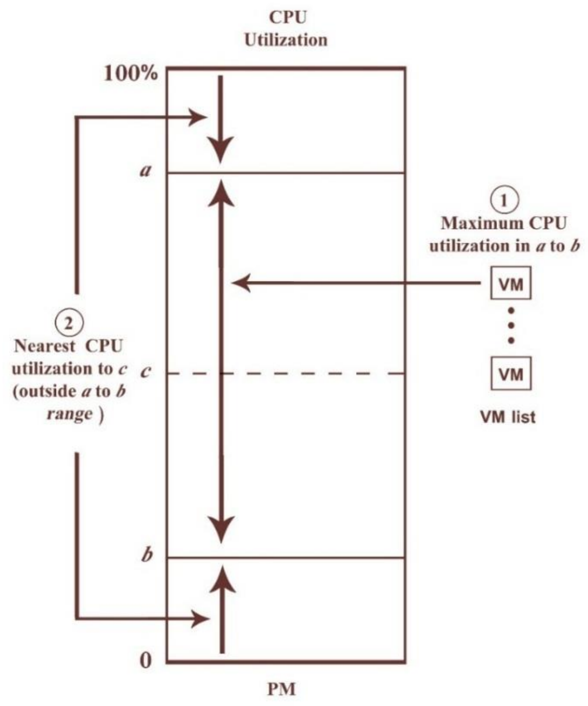

CPU Utilization
100%
a
c
b
0
PM
1
Maximum CPU utilization in a to b
VM
VM
VM list
2
Nearest CPU utilization to c (outside a to b range )

**Fig. 2.** CPU utilization model of PM of CPBFD and DCPBFD algorithm

There is no specific optimal upper CPU threshold value among the researchers [38-42]. Selecting high upper CPU threshold may significantly drop the performance of VMs running on a PM, while selecting low upper CPU threshold value makes consolidation inefficient to reduce energy consumption. Furthermore, there is no specific optimal lower CPU threshold value. So, selecting the suitable value for upper and lower CPU threshold is important. The upper and lower threshold of CPU utilization modified according to parameter *a* and *b*, respectively. CPBFD VM placement algorithm is shown in Algorithm 1.

Algorithm 1: CPBFD VM Placement Algorithm

```
Input: pmList, vmList, $a$, $b$, $c$
Output: allocatedPm
 1; vmList.sortDecreasingUtilization()
 2: for each vm in vmList do
 3:   maxCpu = min
 4:   minCpu = max
 5:   allocatedPm = null
 6:   for each pm in pmList do
 7:     if pm is excluded pm then
 8:       continue
 9:     end if
10:     if pm has enough resources for vm then
```

```
11:       if pm.Cpu ≠ 0 and pm is over utilized after allocation vm then
12:          continue
13:       end if
14:       if pm.Cpu ≤ a and pm.Cpu ≥ b then
15:          if pm.Cpu > maxCpu then
16:             allocatedPm = pm
17:             maxCpu = pm.Cpu
18:          end if
19:       else if Abs(pm.Cpu - c) < minCpu then
20:          allocatedPm2 = pm
21:          minCpu = Abs(pm.Cpu - c)
22:       end if
23:     end if
24:   end for each
25:   if allocatedPm = null then
26:     allocatedPm = allocatedPm2
27:   end if
28:   if allocatedPm ≠ null then
29:     allocation.add(vm,allocatedPm)
30:   end if
31: end for each
32: return allocatedPm
```

## 5.2. DCPBFD Algorithm

CPU resource utilization model of PM of DCPBFD algorithm is the same as that used in CPBFD shown in Fig. 2. The only difference between DCPBFD and CPBFD is that upper and lower CPU utilization thresholds in DCPBFD are dynamic, but in CPBFD they are static as discussed in Section 5.1. In DCPBFD, all VMs are sorted in the decreasing order of their current CPU utilizations and allocate each VM to a PM with maximum CPU utilization less than dynamic upper CPU utilization threshold *a* and more than dynamic lower CPU utilization threshold *b*. If there are no PMs with CPU utilization between upper and lower threshold, a PM with the nearest CPU utilization to upper or lower thresholds will be allocated. The priority may be given to PM with CPU utilization nearest to upper threshold or lower threshold by adjusting parameter *c*. PM with CPU utilization that is nearest to *c* and outside of *a* to *b* range will be selected.

There is no specific optimal upper and lower CPU utilization threshold values among the researchers. In DCPBFD, the proposed optimal value of dynamic upper and lower CPU utilization threshold *a* and *b* are based on fixed highest and lowest CPU utilization values, Median Absolute Deviation (MAD) of historical values of PM CPU utilization, and median of historical values of PM CPU utilization divided by number of VMs.

The fixed highest CPU utilization is used for upper CPU threshold as highest CPU threshold, and fixed lowest CPU utilization is used for lower CPU threshold as lowest CPU threshold. MAD is defined as the median of the absolute deviations from the median of historical values of PM CPU utilization. MAD gives an idea about CPU utilization variability, which is important to calculate suitable upper and lower CPU threshold. Median of historical values of PM CPU utilization divided by number of

VMs gives an idea about the utilization of VMs that will be allocated to PM, which is also important to calculate suitable upper and lower CPU thresholds. $a$ and $b$ are calculated as shown in (7) and (8), respectively.

$$a = HighestCpu \ - s \times (CpuMAD + CpuMedian/Number\ of\ VMs) \tag{7}$$

$$b = LowestCpu + s \times (CpuMAD + CpuMedian/Number\ of\ VMs) \tag{8}$$

where $s$ is a parameter that allows the adjustment of the dynamic upper and lower CPU utilization limits, the lower $s$, the wider two sided limits and the less the energy consumption, but the higher the level of SLA violations caused by the consolidation.

DCPBFD VM placement algorithm is shown in Algorithm 2.

Algorithm 2: DCPBFD VM Placement Algorithm

```
   Input: pmList, vmList, highestCpu, lowestCpu, c, s
   Output: allocatedPm
 1: vmList.sortDecreasingUtilization()
 2: for each vm in vmList do
 3:   maxCpu = min
 4:   minCpu = max
 5:   allocatedPm = null
 6:   for each pm in pmList do
 7:     if pm is excluded pm then
 8:       continue
 9:     end if
10:     if pm has enough resources for vm then
11:       if pm.Cpu ≠ 0 and pm is over utilized after allocation vm then
12:           continue
13:       end if
14:       if (pm.CPUHistory.length ≥ 12) then
15:           a = highestCpu – s × (pm.CPUHistoryMAD + pm.CPUHistoryMedian / #ofvm)
16:           b = lowestCpu + s × (pm.CPUHistoryMAD + pm.CPUHistoryMedian / #ofvm)
17:       else
18:         a = highestCpu - 0.05
19:         b = lowestCpu + 0.05
20:       end if
21:       if pm.Cpu ≤ a and pm.Cpu ≥ b then
22:         if pm.Cpu > maxCpu then
23:           allocatedPm = pm
24:           maxCpu = pm.Cpu
25:         end if
26:       else if Abs(pm.Cpu - c) < minCpu then
27:         allocatedPm2 = pm
28:         minCpu = Abs(pm.Cpu - c)
29:       end if
30:     end if
31:   end for each
32:   if allocatedPm = null then
33:       allocatedPm = allocatedPm2
```

```
34:   end if
35:   if allocatedPm ≠ null then
36:     allocation.add(vm,allocatedPm)
37:   end if
38: end for each
39: return allocatedPm
```

Calculation of MAD starts when at least 12 historical values of CPU utilization of PM are obtained. 12 is used as a safe value to calculate MAD [8]. We suggest having the initial values of *a* and *b* before obtaining the first MAD value. The initial value of *a* is adjusted to (highest CPU threshold – 5%), and the initial value of *b* is adjusted to (lowest CPU threshold + 5%), which are supposed to be appropriate to avoid that *a* reaches the highest CPU threshold and *b* reaches the lowest CPU threshold.

## 6. Performance Evaluation

### 6.1. Experiment Setup

A cloud computing user accesses infinite computing resources. Large scale and repeatable experiments which are necessary to analysis and compare the algorithms is very difficult on a real-world infrastructure [8]. We use simulations for ensuring the repeatability of experiments. We use the CloudSim toolkit [14, 31] as a simulation platform that allows the energy consumption modeling on cloud computing environments.

As presented in [8], we simulate a data center containing 800 heterogeneous PMs. PM types are HP ProLiant ML110 Generation 4 and HP ProLiant ML110 Generation 5. CPU frequencies of the servers are mapped onto MIPS rating: 1,860 MIPS for each core of HP ProLiant ML110 G4 server, and 2,660 MIPS for each core of HP ProLiant ML110 G5 server. Each server is modeled to have 1 GB/s network bandwidth. VM characteristics correspond to Amazon EC2 instance types including Extra Large Instance (3.75 GB, 2,000 MIPS); High-CPU Medium Instance (0.85 GB, 2,500 MIPS); Micro Instance (613 MB, 500 MIPS); and Small Instance (1.7 GB, 1,000 MIPS). Initialization of VMs allocation is done according to the resource requirements of VM types. However, VM's useless resources during the lifetime according to the workload create opportunities for dynamic consolidation.

### 6.2. Workload Data

To make the evaluation of simulation applicable, we use real-world workload traces provided as a monitoring infrastructure for PlanetLab [43], which is a part of the CoMon project. We use CPU utilization traces presented in [8] from more than a thousand VMs running on PMs located in more than 500 places around the world. Utilization is collected every 5 minutes. Random 10 days from the collected workload traces are used in the simulations. The workload characteristics for each day are presented in Table 1. In the simulations, each VM is randomly assigned a workload trace from one of VMs from the corresponding day. VM consolidation is not limited by the memory bounds to avoid the constraint on the consolidation.

**Table 1.** Workload data characteristics [8]

| Workload | Number of VMs | Mean of CPU utilization | Standard deviation of CPU utilization | Median of CPU utilization |
|---|---|---|---|---|
| 1 | 1052 | 12.31% | 17.09% | 6% |
| 2 | 898 | 11.44% | 16.83% | 5% |
| 3 | 1061 | 10.70% | 15.57% | 4% |
| 4 | 1516 | 9.26% | 12.78% | 5% |
| 5 | 1078 | 10.56% | 14.14% | 6% |
| 6 | 1463 | 12.39% | 16.55% | 6% |
| 7 | 1358 | 11.12% | 15.09% | 6% |
| 8 | 1233 | 11.56% | 15.07% | 6% |
| 9 | 1054 | 11.54% | 15.15% | 6% |
| 10 | 1033 | 10.43% | 15.21% | 4% |

## 6.3. Simulations Results

The algorithms are evaluated using the CloudSim and the workload traces presented in Section 6.2. We compare proposed CPBFD algorithm to FFD, BFD and PABFD. We simulate all combinations of SM underload detection algorithm, LR overloading detection algorithm, MMT VM selection policy and four VM placement algorithms (FFD, BFD, PABFD and proposed CPBFD). Algorithms that are used in dynamic VM consolidation problem are shown in Fig. 3.

For the proposed CPBFD algorithm, two-sided limits (*a* and *b*) of CPU utilization is varied from wider to narrower based as 90% to 10%, 80% to 20%, 70% to 30% and 60% to 40%. Moreover, *c* parameter is varied to give more priority for low CPU utilization, equal priority to low or high CPU utilization, and more priority for high CPU utilization as 0.45, 0.50, and 0.55, respectively. According to these variations, combinations of *a*, *b* and *c* parameters are varied as (0.9, 0.1, 0.45), (0.9, 0.1, 0.5), (0.9, 0.1, 0.55), (0.8, 0.2, 0.45), (0.8, 0.2, 0.5), (0.8, 0.2, 0.55), (0.7, 0.3, 0.45) (0.7, 0.3, 0.5), (0.7, 0.3, 0.55), (0.6, 0.4, 0.45) (0.6, 0.4, 0.5), and (0.6, 0.4, 0.55). The purposes of these variations are to get better upper and lower CPU utilization threshold and better priority for low or high CPU utilization for the proposed CPBFD algorithm and to use these better parameters in proposed DCPBFD algorithm.

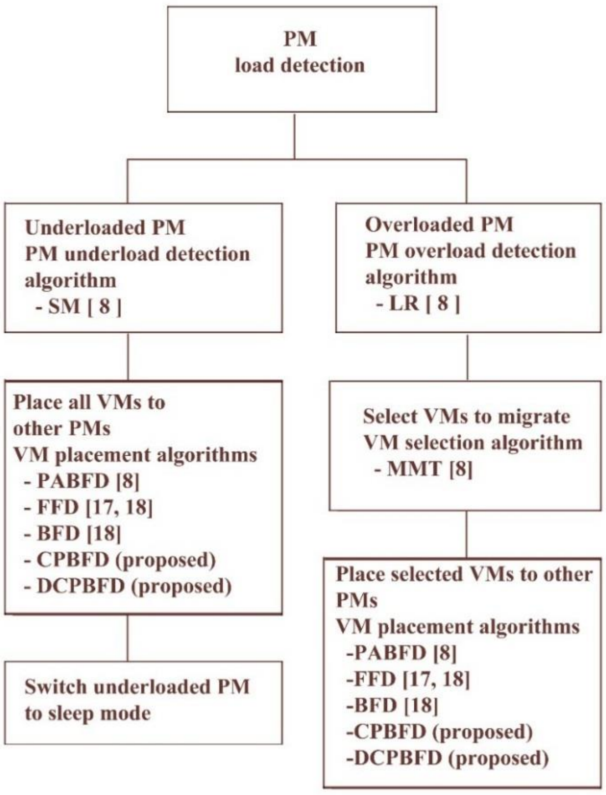


**Fig. 3.** Dynamic VM consolidation problem and algorithms in cloud data centers.

Figs. 4 to 9 show the average results with 95% confidence intervals of ESV metric, energy consumption, SLAV metric, number of migrations, PDM metric, and SLATAH metric for all algorithms combination in 10 workload cases, respectively. Results in Figs. 4 to 9 show that CPBFD leads to better results regarding all parameters compared to FFD. CPBFD leads to better of ESV metric, SLAV metric, number of migrations, PDM metric, and SLATAH metric compared to BFD and PABFD. Moreover, BFD leads to better results regarding ESV metric, SLAV metric, number of migrations, PDM metric compared to FFD and PABFD. On the other hand, PABFD only leads to the least energy consumption compared to all algorithms. Fig. 4 shows that CPBFD has better ESV on average approximately 61.1%, 54.2% and 17.5% than FFD, PABFD, and BFD, respectively. Fig. 5 shows that CPBFD has less energy consumption on average approximately 3.8% than FFD and more energy consumption on average approximately 0.75%, and 3% than BFD, and PABFD, respectively. Fig. 6 shows that CPBFD has less SLAV on average approximately 54.2%, 56.7% and 19% than FFD, PABFD, and BFD, respectively. Fig. 7 shows that CPBFD has fewer number of migration on average approximately 22.1%, 27.4%, and 8.3% than FFD, PABFD, and BFD, respectively. Fig. 8 shows that CPBFD has less PDM on average approximately 35.9%, 52.5% and 12.7% than FFD, PABFD, and BFD, respectively. Fig. 9 shows that CPBFD has less SLATAH on average approximately 19.3%, 8% and 13.1% than FFD, PABFD, and BFD, respectively.

CPBFD_70,30,55 has better ESV and SLAV on average approximately 7.7% and 8.7% than CPBFD with other parameters. CPBFD_80,20,55 has better efficient energy consumption on average approximately 0.8% than CPBFD with other parameters. CPBFD with ($a$ = 70, $b$ = 30) of CPU has better ESV on average approximately 14.5%, 3% and 4.3% than CPBFD with ($a$ = 90, $b$ = 10), CPBFD with ($a$ = 80, $b$ = 20), and CPBFD with ($a$ = 60, $b$ = 40), respectively. CPBFD with ($a$ = 70, $b$ = 30) of CPU has better SLAV on average approximately 15.9%, 3.9% and 3.9% than CPBFD with ($a$ = 90, $b$ = 10), CPBFD with ($a$ = 80, $b$ = 20), and CPBFD with ($a$ = 60, $b$ = 40), respectively. CPBFD with ($a$ = 80, $b$ = 20) has almost the same energy consumed by CPBFD with ($a$ = 90, $b$ = 10), and better efficient energy consumption on average approximately 0.7%, and 1.5% than CPBFD with ($a$ = 70, $b$ = 30), and CPBFD with ($a$ = 60, $b$ = 40).

CPBFD_80,20,55 has almost the same energy consumed by BFD, and more energy consumption on average approximately 2.3% than PABFD, but CPBFD_80,20,55 has better ESV and SLAV on average approximately 62.2% and 63.9% than PABFD. This means even if we consider that ESV metric is modified to the product of energy consumption powered by 20 and SLAV (modified ESV = $E^{20} \times$ SLAV), CPBFD_80,20,55 will still better than PABFD in regard of modified ESV.

The energy consumption changes slightly compared to SLA violation. Therefore, the impact of SLA violation on ESV metric is greater than energy consumption. However, the suitable value of $a$, $b$ and $c$ should be selected to make a tradeoff between meeting QoS and improving energy efficiency. From the simulation results, we observe that CPBFD with ($a$ = 70, $b$ = 30) provides best ESV metric, SLA violations and number of migrations. In addition, CPBFD with ($a$ = 80, $b$ = 20) provides best efficient energy consumption. Moreover, CPBFD when $c$ parameter equals 0.55 leads to a little better ESV metric, SLA violations and energy consumption. Furthermore, we observe that selecting moderate two-sided limits of CPU utilization between ($a$ = 70, $b$ = 30) and ($a$ = 80, $b$ = 20) for CPBFD is better than selecting too wide ($a$ = 90, $b$ = 10) or too narrow ($a$ = 60, $b$ = 40) sided limits of CPU utilization.

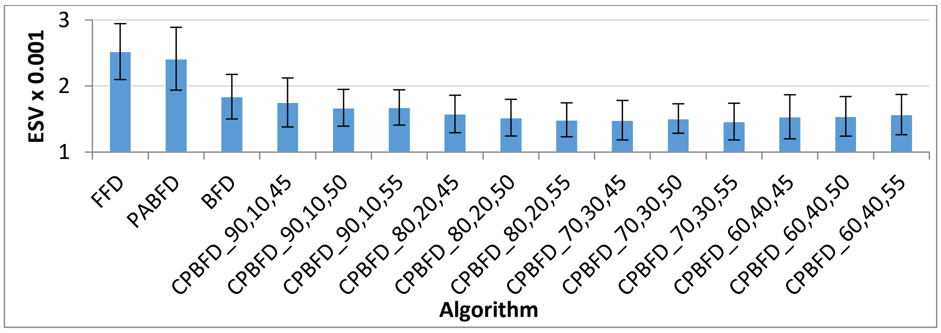


**Fig. 4.** ESV metric of VM placement algorithms

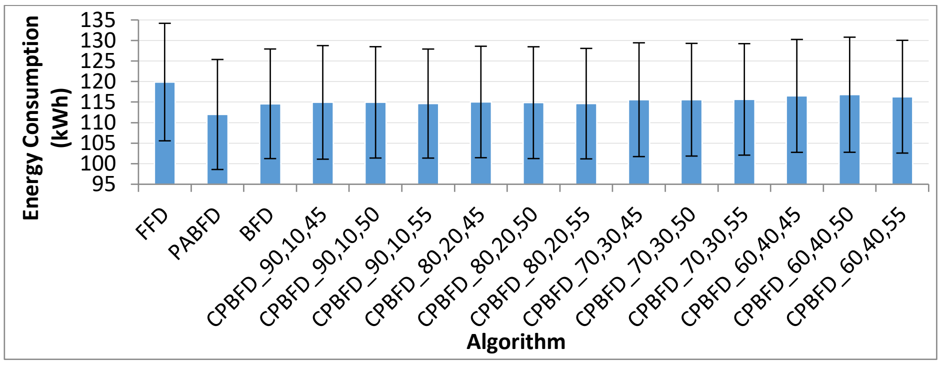


**Fig. 5.** Energy consumption of VM placement algorithms

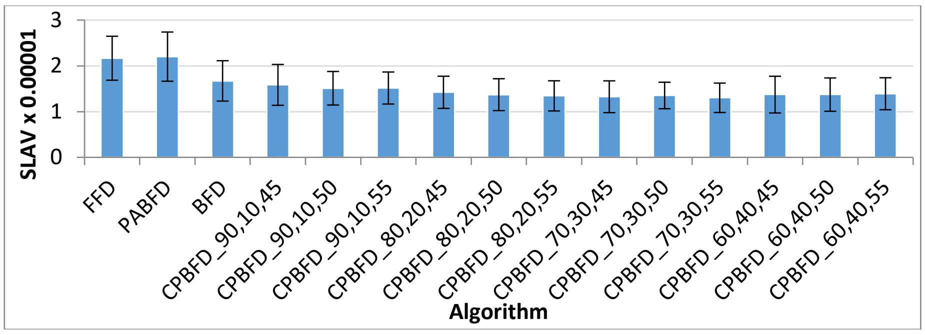


**Fig. 6.** SLAV metric of VM placement algorithms

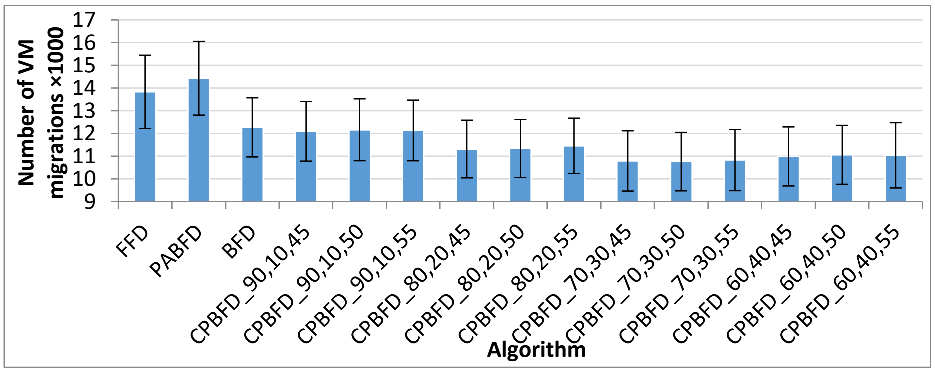


**Fig. 7.** Number of VM migrations of VM placement algorithms

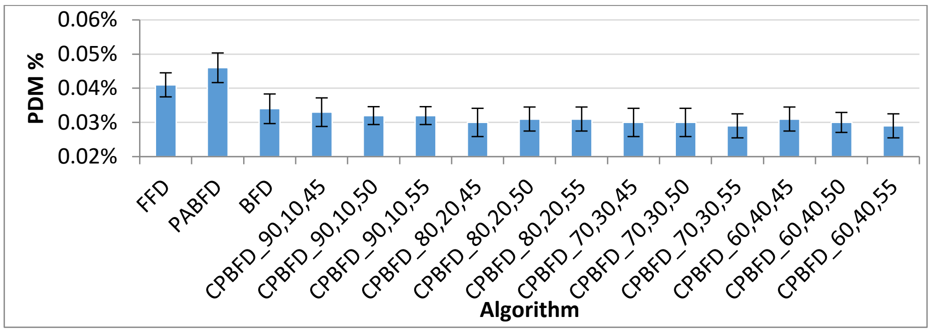


**Fig. 8.** PDM metric of VM placement algorithms

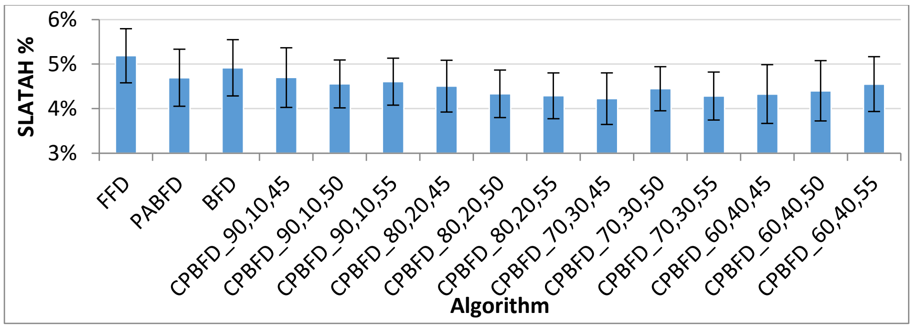


**Fig. 9.** SLATAH metric of VM placement algorithms

For the proposed DCPBFD algorithm, the fixed highest CPU utilization used for higher CPU utilization limit (*a*) is set to 80%, and the fixed lowest CPU utilization used for lower CPU utilization limit (*b*) is set to 20%, which are suitable moderate values according to the results obtained from CPBFD algorithm. *s* parameter of two-sided limits (*a* and *b*) of CPU utilization is varied as 1, 0.75, and 0.5. Moreover, *c* parameter is set to 0.55 to give more priority for high CPU utilization, which gives the best result according to the results obtained from CPBFD algorithm. According to these variations, combinations of the fixed highest CPU utilization, the fixed lowest CPU utilization, *c* and *s* parameters are varied as (0.8, 0.2, 0.55, 1), (0.8, 0.2, 0.55, 0.75), and (0.8, 0.2, 0.55, 0.5).

Figs. 10 to 15 show the average results with 95% confidence intervals of ESV metric, energy consumption, SLAV metric, number of migrations, PDM metric, and SLATAH metric for DCPBFD algorithm with all combination of parameters, and CPBFD_80,20,55, CPBFD_75,25,55, and CPBFD_70,30,55 that have the best results compared to CPBFD with other parameters. Results in Figs. 10 to 15 show that DCPBFD_80,20,55,75 leads to better results compared to DCPBFD with other parameters, and CPBFD_75,25,55 leads to better results compared to CPBFD with other parameters. Results in Figs. 10, 12, 13, 14, and 15 show that DCPBFD_80,20,55,75 leads to better of ESV metric, SLAV metric, number of migrations, PDM metric, and SLATAH metric compared to DCPBFD and CPBFD with other parameters. Moreover, results in Fig. 11 show that CPBFD_80,20,55 provides a little better efficient energy consumption compared to DCPBFD and CPBFD with other parameters. We observe that selecting moderate two-sided limits of CPU utilization for DCPBFD by selecting moderate highest CPU utilization and lowest CPU utilization and adjusting *s* parameter to moderate value is better than selecting too wide or too narrow sided limits of CPU utilization.

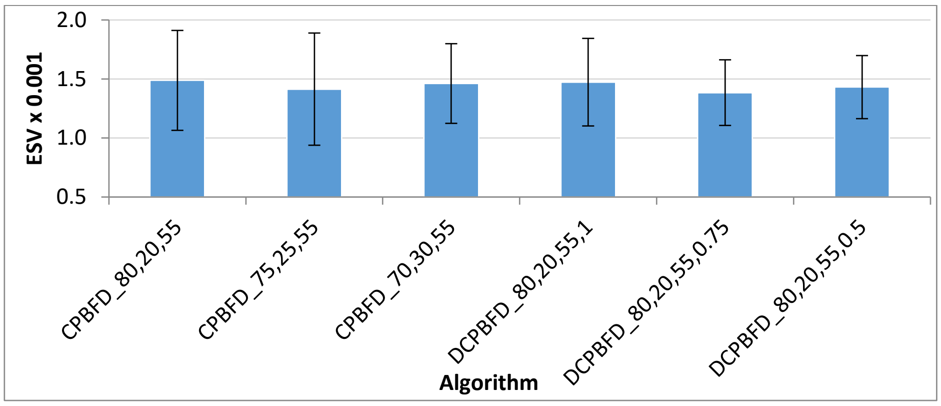


**Fig. 10.** ESV metric of CPBFD and DCPBFD algorithms

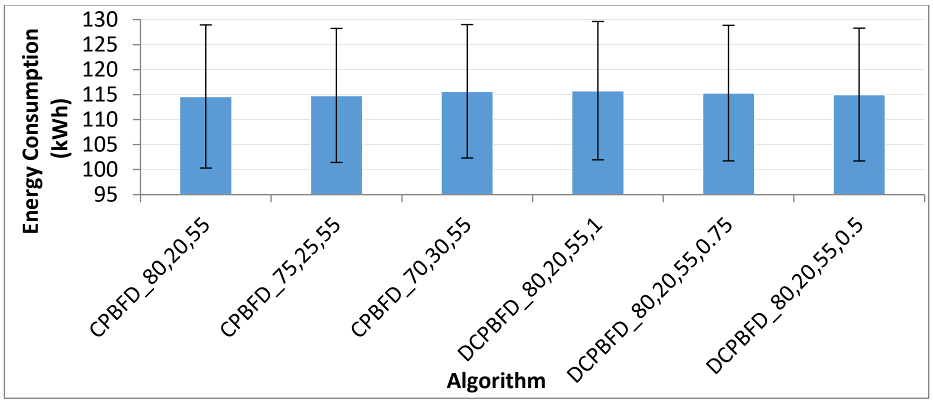


**Fig. 11.** Energy consumption of CPBFD and DCPBFD algorithms

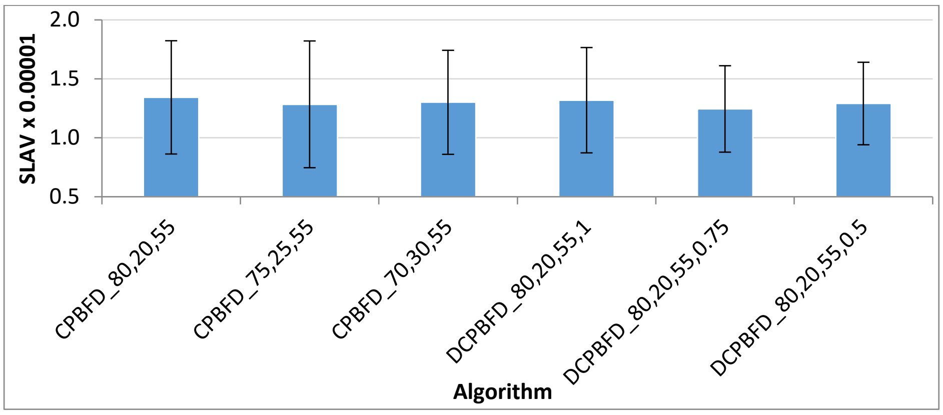


**Fig. 12.** SLAV metric of CPBFD and DCPBFD algorithms

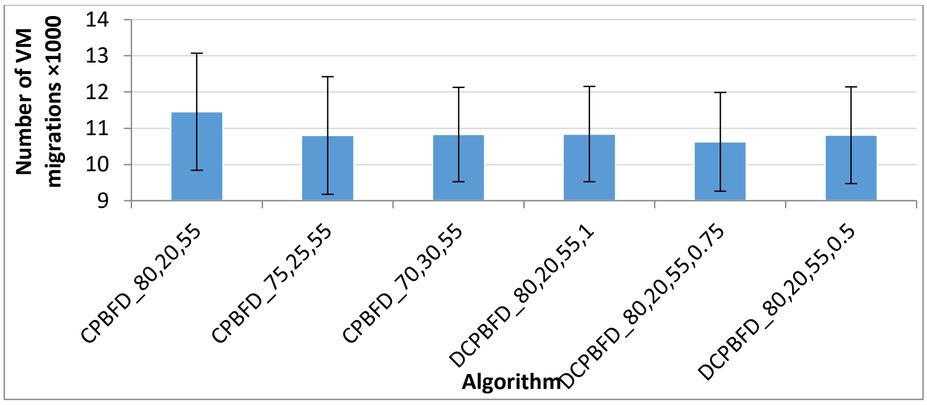


**Fig. 13.** Number of VM migrations of CPBFD and DCPBFD algorithms

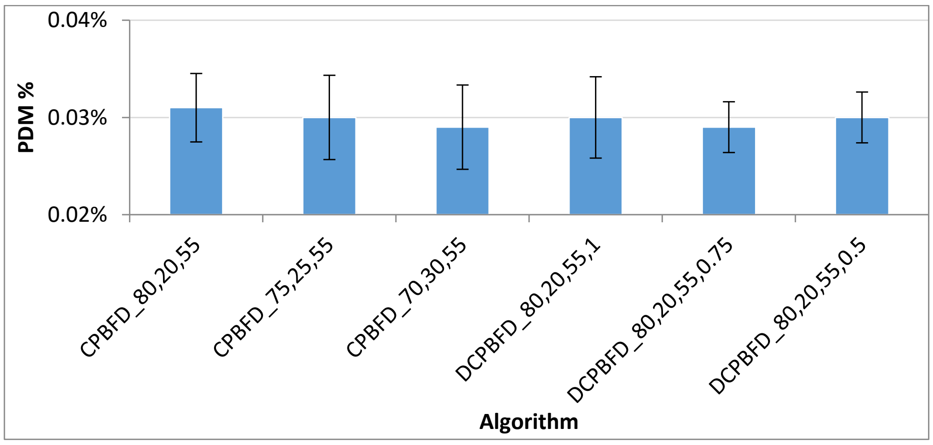


**Fig. 14.** PDM metric of CPBFD and DCPBFD algorithms.

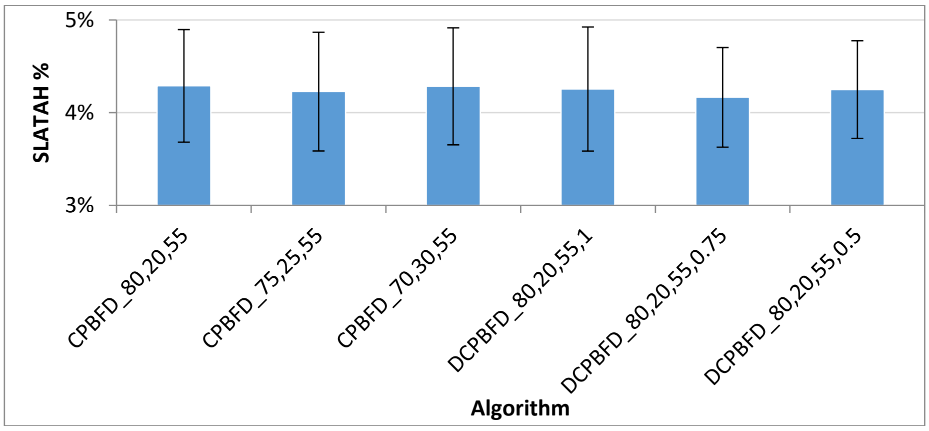


**Fig. 15.** SLATAH metric of CPBFD and DCPBFD algorithms.

## 7. Conclusion and Future Work

The goal of the proposed CPBFD and DCPBFD VM placement algorithms is to improve energy efficiency, and SLA in cloud data centers. A number of VM placement algorithms are implemented to compare with the proposed algorithm. We evaluate the algorithms through simulations with real-world workload traces. CPBFD and DCPBFD algorithms produce better results by avoiding VM migrations to PM with high load that may cause SLA violations or low load, which lead to more active PMs and more energy consumption. The simulation results show that CPBFD and DCPBFD with moderate two-sided limits of CPU utilization provide the least ESV, least SLA violations, least VM migrations, and efficient energy consumption. However, PABFD algorithm leads to a little better energy consumption than CPBFD and DCPBFD algorithms.

As a future work, we plan to extend our research by using a software framework for dynamic and energy efficient consolidation of VMs applied in existing cloud deployments and in research on dynamic consolidation of VMs to optimize the resource utilization and energy efficiency.

## References

1. Weber, W. D., Fan, X., Barroso, L. A.: Powering the data center. U.S. Patent No. 8,595,515. Washington, DC: U.S. Patent and Trademark Office. (2013)
2. Beloglazov, A., Buyya, R.: Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No.7, 1366-1379. (2013)
3. Verma, A., Ahuja, P., Neogi, A.: pMapper: power and migration cost aware application placement in virtualized systems. In Proceedings of ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer Berlin Heidelberg, 243-264. (2008)

4. Jung, G., Hiltunen, M. A., Joshi, K. R., Schlichting, R. D., Pu C.: Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In Proceedings of IEEE 30th International Conference on Distributed Computing Systems, 62-73. (2010)
5. Gmach, D., Rolia, J., Cherkasova, L., Kemper, A.: Resource pool management: Reactive versus proactive or lets be friends. Computer Networks, Vol. 53, No. 17, 2905-2922. (2009)
6. Fu, X., Zhou, C.: Virtual machine selection and placement for dynamic consolidation in Cloud computing environment. Frontiers of Computer Science, Vol. 9, No. 2, 322-330. (2015)
7. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems, Vol. 28, No. 5, 755-768. (2012)
8. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive Heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice and Experience, Vol. 24, No. 13, 1397-1420. (2012)
9. Han, G., Que, W., Jia, G., Shu, L.: An efficient virtual machine consolidation scheme for multimedia cloud computing. Sensors, Vol. 16, No. 2, 246-246. (2016)
10. Guenter, B., Jain, N., Williams, C.: Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In Proceedings of 30'st Annual IEEE Intl. Conf. on computer communications (INFOCOM), 1332-1340. (2011)
11. Alsbatin, L., Oz, G., Ulusoy, A. H.: An overview of energy-efficient cloud data centres. In Proceedings of the International Conference of computer and applications (ICCA2017), Dubai, United Arab Emirates, 211- 214. (2017)
12. Lee, Y. C., Zomaya, A. Y.: Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 8, 1374–1381. (2011)
13. Kang, J., Ranka, S.: Dynamic slack allocation algorithms for energy minimization on parallel machines. Journal of Parallel and Distributed Computing, Vol. 70, No. 5, 417–430. (2010)
14. Buyya, R., Ranjan, R., Calheiros, R.N.: Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In Proceedings of 2009 Conference on High Performance Computing & Simulation Conference, 1-11. (2009)
15. Beloglazov, A., Buyya, R.: Energy efficient resource management in virtualized cloud data centers. in Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 826–831. (2010)
16. Alsbatin, L., Oz, G., Ulusoy, A. H.: A Novel Physical Machine Overload Detection Algorithm Combined with Queiscing for Dynamic Virtual Machine Consolidation in Cloud Data Centers. The International Arab Journal of Information Technology, Vol. 17, No. 3. (2020). Available online: https://iajit.org/PDF/May%202020,%20No.%203/16897.pdf.
17. Kaushar, H., Ricchariya, P., Motwani, A.: Comparison of SLA based energy efficient dynamic virtual machine consolidation algorithms. International Journal of Computer Applications, Vol. 102, No. 16. (2014)
18. Sharifi, M., Salimi, H., Najafzadeh, M.: Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques. Journal of Supercomputing, Vol. 61, No.1, 46-66. (2012)
19. Deng, D., He, K., Chen, Y.: Dynamic virtual machine consolidation for improving energy efficiency in cloud data centers. In Proceedings of 4th international conference on cloud computing and intelligence systems. (2016)
20. Khoshkholghi, M. A., Derahman, M. N., Abdullah, A., Subramaniam, S., Othman, M.: Energy-efficient algorithms for dynamic virtual machine consolidation in cloud data centers. IEEE Access, Vol. 5, 10709-10722. (2017)

21. Shaw, S. B., Singh, A. K.: Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data centre. Computers & Electrical Engineering, Vol. 47, 241-254. (2015)
22. Nguyen, T. H., Francesco, M. D., Yla-Jaaski, A.: Virtual Machine Consolidation with Multiple Usage Prediction for Energy-Efficient Cloud Data Centers. IEEE Transactions on Services Computing, Vol. 99, 1-14. (2017)
23. Zhou, Z., Abawajy, J., Chowdhury, M., Hu, Z., Li, K., Cheng, H., Alelaiwi, A. A., Li, F.: Minimizing SLA violations and power consumption in cloud data centers using adaptive energy-aware algorithms. Future Generation Computer Systems, Vol. 86, 836-850. (2018)
24. Wang, X., Wang, Y.: Coordinating power control and performance management for virtualized server clusters. IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 2, 245-259. (2011)
25. Yadav, R. Zhang, W.: MeReg: Managing Energy-SLA Tradeoff for Green Mobile Cloud Computing. Wireless Communications and Mobile Computing, Vol. 2017. (2017)
26. Kakadia, D., Kopri, N., Varma, V.: Network-aware virtual machine consolidation for large data centers. In Proceedings of 3rd International Workshop on Network-Aware Data Management. (2013)
27. Forsman, M., Glad, A., Lundberg, L., Ilie, D.: Algorithms for automated live migration of virtual machines. Journal of Systems and Software, Vol. 101, 110-126. (2015)
28. Yue, M.: A simple proof of the inequality FFD (L)< 11/9 OPT (L)+ 1,for all l for the FFD bin-packing algorithm. Acta Mathematicae Applicatae Sinica (English Series), Vol. 7, No. 4, 321-331. (1991)
29. Coffman, E. G., Garey, M. R., Johnson, D. S.: Approximation algorithms for bin-packing: A survey. Approximation algorithms for NP-hard problems, 46-93. (1996)
30. Shi, L., Furlong, J., Wang, R.: Empirical evaluation of vector bin packing algorithms for energy efficient data centers. in IEEE Symposium on Computers and Communications, 9-15. (2013)
31. Calheiros, R. N., Ranjan, R., Beloglazov, A., Rose, C. A., Buyya, R.: CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, Vol. 41, No. 1, 23-50. (2011)
32. Minas, L., Ellison, B.: Energy efficiency for information technology: How to reduce power consumption in servers and data centers. Intel Press. (2009)
33. Fan, X., Weber, W. D., Barroso, L.A.: Power provisioning for a warehouse-sized computer. ACM SIGARCH Computer Architecture News, Vol. 35, No. 2, 13-23. (2007)
34. Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., Jiang, G.: Power and performance management of virtualized computing environments via lookahead control. Cluster Computing, Vol. 12, No. 1, 1-15. (2009)
35. Lange, K. D.: Identifying shades of green: The SPECpower benchmarks. Computer, Vol. 42, No.3, 95-97. (2009)
36. Beloglazov, A., Buyya, R.: Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science. (2010)
37. Ammar, A., Luo, J, Tang, Z, Wajdy, O.: Intra-Balance Virtual Machine Placement for Effective Reduction in Energy Consumption and SLA Violation. IEEE Access, Vol. 7. (2019)
38. Murtazaev, A., Oh, S.: Sercon: Server consolidation algorithm using live migration of virtual machines for green computing. IETE Techical Review, Vol. 28, No. 3, 212-231. (2011)
39. Vogels, W.: Beyond Server Consolidation. ACM Queue, Vol. 6, No. 1, 20-26. (2008)
40. Magesh, H., Smith, J.: Server consolidation through virtualization with quad-core intel xeon processors. White Paper by Intel Corporation and Infosys Technologies. (2008)
41. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and gray-box strategies for virtual machine migration. 4th USENIX Symposium on Networked Systems Design & Implementation. (2007)

42. Song, Y., Wang, H., Li, Y., Feng, B., Sun, Y.: Multi-tiered ondemand resource scheduling for vm-based data center. 9th IEEE/ ACM International Symposium on Cluster Computing and the Grid. (2009)
43. Park, K., Pai, V. S.: CoMon: A mostly-scalable monitoring system for PlanetLab. ACM SIGOPS Operating Systems Review, Vol. 40, No. 1, 65-74. (2006)

**Loiy Alsbatin** received his B.S. degree in Computer Engineering from Mutah University, Jordan, in 2008, his M.S. degree in Computer Engineering from Jordan University of Science and Technology (JUST), Jordan, in 2012, and his Ph.D. degree in Computer Engineering from Eastern Mediterranean University (EMU), in Famagusta, North Cyprus, in 2019. He is currently a faculty member in the Department of Computer Science at Shaqra University, Saudi Arabia. His current research interests include distributed system, cloud computing, resource management, and virtualization.

**Gurcu Oz** received her B.S, M.S. degrees from the Electrical and Electronic Engineering department and Ph.D. degree from the Computer Engineering Department of Eastern Mediterranean University, in Famagusta, North Cyprus. Currently, she is working in the Department of Computer Engineering of Eastern Mediterranean University. Her research interests include computer networks, wireless networks, distributed systems, cloud computing, system simulation, information security and network security.

**Ali Hakan Ulusoy** was born in Eskisehir, Turkey, on June 3, 1974. He graduated from the double major program of the department of Electrical and Electronic Engineering (EEE) and department of Physics in Eastern Mediterranean University (EMU) in 1996. He received his M.S. and Ph.D. degrees in EEE in EMU in 1998 and 2004, respectively. He joined Information Technology department, EMU, in 2004. His current research interests include wireless communications, receiver design, channel estimation, fuzzy systems, wireless networks, cloud computing, millimeter wave communications and healthcare system development.

*Received: November 22, 2018; Accepted: November 20, 2019*

Computer Science and Information Systems 17(1):51–70 https://doi.org/10.2298/CSIS181101016C

# Towards a software-based mobility management for 5G: An experimental approach for flattened network architectures.

Jesús Calle-Cancho[1], José-Manuel Mendoza-Rubio[2], José-Luis González-Sánchez[1], David Cortés-Polo[1], and Javier Carmona-Murillo[2]

[1] Research, Technological Innovation and Supercomputing Center of Extremadura (CénitS), 10071 Cáceres, Spain
{jesus.calle,joseluis.gonzalez,david.cortes}@cenits.es
[2] Department of Computing and Telematics Engineering, University of Extremadura 10003 Cáceres, Spain
{jmendozah, jcarmur}@unex.es

**Abstract.** The number of mobile subscribers, as well as the data traffic generated by them, is increasing exponentially with the growth of wireless smart devices and the number of network services that they can support. This significant growth is pushing mobile network operators towards new solutions to improve their network performance and efficiency. Thus, the appearance of Software Defined Networking (SDN) can overcome the limitations of current deployments through decoupling the network control plane from the data plane, allowing higher flexibility and programmability to the network. In this context, the process of handling user mobility becomes an essential part of future mobile networks. Taking advantage of the benefits that SDN brings, in this article we present a novel mobility management solution. This proposal avoids the use of IP-IP tunnels and it adds the dynamic flow management capability provided by SDN. In order to analyse performance, an analytical model is developed to compare it with NB-DMM (Network-based DMM), one of the main DMM (Distributed Mobility Management) solutions. Additionally, performance is also evaluated with an experimental testbed. The results allow handover latency in real scenarios and numerical investigations to be measured, and also show that SR-DMM achieves better efficiency in terms of signaling and routing cost than NB-DMM solution.

**Keywords:** SDN, DMM, IPv6 mobility, cost analysis, experimental evaluation.

## 1. Introduction

With the ever more rapid development and innovations of wireless communications, and the advancement of more powerful and smart mobile devices, the future mobile networks are expected to be able to provide new services according to the specific demands of the users. These advances have generated an exponential growth of global mobile data traffic which will increase sevenfold between 2017 and 2022, reaching 77.5 exabytes per month by 2022 [1]. The signaling load is expected to increase almost 50% faster than the growth in data traffic [2]. Driven by this massive wireless data traffic increase, efficient network management mechanisms have been revealed as one of the major challenges in next-generation mobile networks [3, 4]. These mechanisms must be able to dynamically control and allocate network resources to provide flexibility in the new 5G ecosystem [5].

One of these processes, involved in the network management, is the mobility support. Mobility management protocols are responsible for maintaining the active services while the user roams between different networks. For this purpose, the Internet Engineering Task Force (IETF) standardized IP mobility management protocols that are Mobile IPv6 (MIPv6) [6] and Proxy Mobile IPv6 (PMIPv6) [7]. The operation of these solutions is based on the existence of a central entity responsible for managing the movement of the mobile node. This agent maintains the location of MNs and redirects traffic to them. However, these centralized mobility management paradigms are not efficient when handling a large volume of mobile data traffic. Our previous works [8,9] establish the limitations and problems of these approaches such as non-optimal routes, centrally deployed mobility anchors (single point of failure) and reliability and scalability issues [10].

In order to address these problems, new distributed mobility management approaches are being proposed, in which the mobility anchors are positioned closer to the user with the aim of getting a flatter network. Moreover, SDN has emerged as an efficient network approach capable of supporting the dynamic nature of the next-generation wireless networks. In this article, we propose a mobility management solution, called SR-DMM, that combine SDN and DMM and improves the mobility management process in 5G networks, taking full advantages of the software capabilities of the SDN paradigm. The key benefit of the SR-DMM solution is that it manages to reduce significantly both signaling and routing cost, optimizing control and data plane. The signaling overhead is reduced because our proposal does not require the binding refresh process. Moreover, the complexity of the data plane and the tunnel management is also reduced avoiding the use of IP-IP tunnels during the movement of the users. Due to this, the proposed solution can reduce handover latency. These benefits have also been identified and validated through experimental evaluation in a testbed. In addition, an analytical framework has been developed in order to evaluate and compare our proposal with a previous legacy DMM proposal.

The rest of the paper is organized as follows. In Section 2, we briefly present background information about Distributed Mobility Management protocols. Then, Section 3 describes Software Defined Networking paradigm and SR-DMM is introduced. Section 4 shows the analytical model used to evaluate the proposal, and the numerical results of this analysis. Section 5 presents the performance evaluation through the experimental prototype. Finally, Section 6 concludes the paper.

## 2. Related work

Nowadays, most of the deployed architectures, such as 3GPP (Third Generation Partnership Project) networks, have a small number of centralized anchors managing the traffic of thousands of mobile users, but these centralized approaches have certain problems [8]. The evolution towards DMM approaches has shown improvements in better use of network resources [11, 12, 13, 14, 15]. Additionally, future mobile networks will be driven by software, relying on emerging technologies such as SDN [16, 17, 18].

Similarly to other mobility solutions, distributed mobility management protocols can be broadly classified in two categories, depending on the role of the mobile node in the handover process, namely those that require the active involvement of the MN (Host-

Based DMM) and those that not (Network-Based DMM). Thus, in this section, an overview of the distributed mobility management protocols is presented in order to then discuss its relationship with the Software Defined Networking paradigm and its implications for future mobile networks. A brief description of the existing solutions is also given for comparison with the SDN-based DMM solution proposed in this paper.

## 2.1. Host-Based DMM

This DMM proposal is based on Mobile IPv6 and is detailed in [13, 19] (Host-Based DMM, HB-DMM). HB-DMM extends mobility signaling and reuses many concepts such as the binding cache at the MN, binding cache at the mobility anchor or tunneling. Moreover, in [19], the authors attempt to improve the performance of mobility support by extending the MIPv6s HA to the AMA (Access Mobility Anchor), which is a new mobility anchor defined for the proposed Host-Based DMM approach. These AMAs are distributed at the edge of the access network level and the MN configures its address based on the provided network prefix from the AMA.

When an MN moves to an adjacent access network, served by another AMA, a new address is configured in the MN based on the network prefix obtained from the serving AMA at the new access network, while it keeps the previous address from the origin AMA. As a result of the signaling between the serving AMA and the origin AMA, a bidirectional tunnel is created between them through new signaling messages called Access Binding Update (ABU) and Access Binding Acknowledgement (ABA). As depicted in Fig. 1, this solution creates multiple tunnels between AMAs and, in cases where a high mobility rate exists, the system performance might be critically compromised by the frequent registrations and maintenance of multiple tunnels.

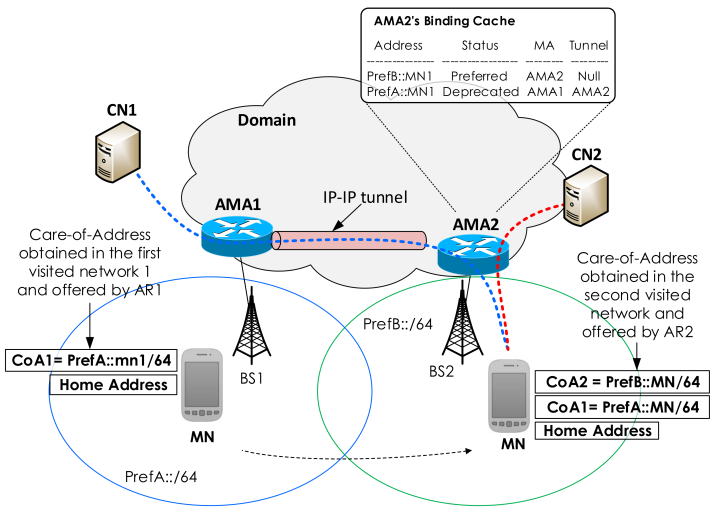
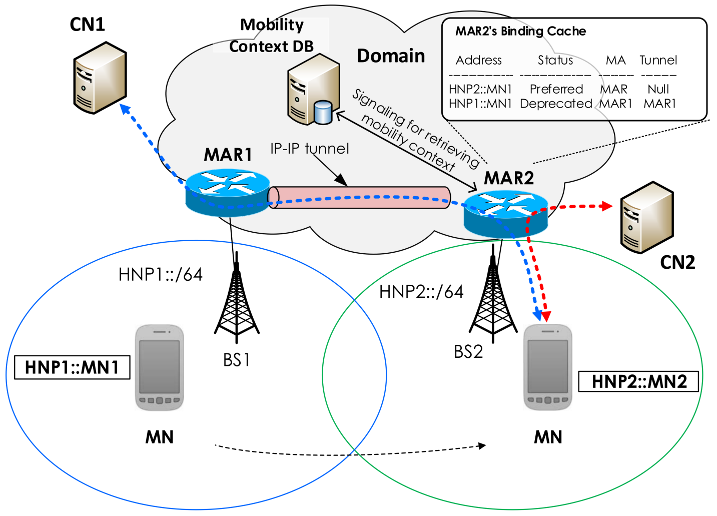

AMA2's Binding Cache
Address Status MA Tunnel
PrefB::MN1 Preferred AMA2 Null
PrefA::MN1 Deprecated AMA1 AMA2
CN1
Domain
CN2
AMA1
IP-IP tunnel
AMA2
Care-of-Address obtained in the first visited network 1 and offered by AR1
Care-of-Address obtained in the second visited network and offered by AR2
PrefB::/64
CoA1= PrefA::mn1/64
Home Address
BS1
BS2
CoA2 = PrefB::MN/64
CoA1= PrefA::MN/64
Home Address
MN
MN
PrefA::/64

**Fig. 1.** Host-Based DMM

## 2.2. Network-Based DMM

Network-Based DMM (NB-DMM) [20] is a Distributed Mobility Management approach that shares with PMIPv6 the fact that it is network-based. It exempts the MN from participating in any mobility signaling, so no network software upgrade is required at the MN for mobility support because distributed mobility anchors perform mobility signaling on behalf of the MN. This NB-DMM is one of the early proposals designed in the IETF for network-based DMM at the Distributed Mobility Management Working Group. In NB-DMM, the mobility management functionalities are moved to the access routers (AR) level in order to anchor the traffic closer to the MN. Each AR is required to have both mobility anchoring and location functionalities, and it is referred to as a mobility capable access router (MAR).

In NB-DMM, a new session is anchored at the current AR and initiated using the current IPv6 address. When a handover occurs before the end of the session, the data traffic of this session is tunneled between the current MAR and the anchoring MAR for this session. In order to achieve a network-based solution without the participation of the MN in the mobility signaling, the architecture is partially distributed and relies on a centralized database (Mobility Context DB, CMD). This DB stores ongoing sessions for the MNs; it stores the home network prefix currently allocated to the MN and their respective anchoring points. Thus, upon a handover, the new MAR retrieves the IP addresses of the anchoring MAR for the MNs sessions from the database. Then, the new MAR proceeds to update the location by sending a PBU to each anchoring MAR. Each anchoring MAR replies by a PBA. The basic operation of NB-DMM is depicted in Fig. 2.



MAR2's Binding Cache

| Address | Status | MA | Tunnel |
|---|---|---|---|
| HNP2::MN1 | Preferred | MAR | Null |
| HNP1::MN1 | Deprecated | MAR1 | MAR1 |

**Fig. 2.** Network-Based DMM

## 3. SDN-Based Distributed Mobility Management

Software Defined Networking is an emerging approach to designing, building and managing networks. This term has been coined in recent years and is currently attracting attention from universities and industry as important architecture for the management of traditional IP networks which are complex and very hard to manage. The SDN architecture is directly programmable, agile, centrally managed and open standards-based.

Furthermore, Fig. 3 describes the SDN functional architecture which consists of three main layers. The Infrastructure layer involves the physical network equipment, the control layer consists of the network controllers and the application layer involves functional applications.

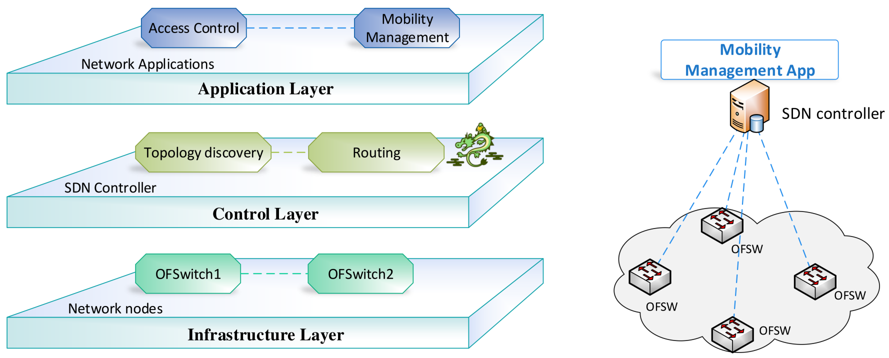


**Fig. 3.** SDN functional architecture

In this way, SDN provides innovation, improved performance and enhanced configurations. Therefore, this paradigm can be seen as a great opportunity to manage mobility efficiently in 5G networks.

### 3.1. SR-DMM solution

The objective of managing the mobility of MNs through SDN technology is to achieve solutions where the control plane is centralized and separated from the data plane, which is distributed. Mobility management is offered based on a service developed as an SDN application that will run on the network controller. This network controller manages a control plane that consists of generic hardware. Therefore, the main objective of our proposal (SDN Redirection DMM) is to provide flexibility, scalability and reliability to the future wireless communications by using SDN capabilities in order to manage mobility as a service. SR-DMM focuses on providing L3 improvements such as L3 handover latency reduction. However, there are other solutions, which include improvements at the link-layer [21, 22], but they are out of the scope of this paper.

In this work, network flexibility refers to the ability of a network to adapt its resources [23, 24]. SR-DMM offers flexibility through SDN programmability. The proposed solution implements a mechanism via open standards to optimally redirect flows when an MN moves through the network domain. Moreover, scalability is defined as the ability to, more specifically in the control plane, handle an increasing workload [25]. SR-DMM provides scalability by reducing the signaling overhead because our proposal does not require the binding refresh process. In the following sections, flexibility and scalability are measured analytically in terms of signaling cost and packet delivery cost respectively. These improvements are also measured experimentally in terms of handover latency. In addition, SR-DMM introduces reliability through centralized network controllers which must be capable of meeting real time requirements of the network [26].

The centralized network controller has a global vision of the entire network. This opportunity is taken advantage of by SDN application through OpenFlow channel. Therefore, mobility service knows the global status of the DMM domain data plane in order to make timely decisions.

On the other hand, the network controller provides capabilities to the edge switches through OpenFlow interfaces by anchoring the packet flow to each mobile node. This solution distinguishes between switches with anchor capabilities (edge switches) and switches without these capabilities. The SR-DMM application is developed on the network controller and it allows flow tables over OpenFlow Switches (OFSwitches) with anchor capabilities to be configured. Moreover, the SR-DMM application provides the edge OFSwitches with other functionalities such as neighbour discovery and access control.

Due to the global vision of the network controller, the proposal avoids overheads introduced by tunneling between mobility anchors in DMM solutions by performing flow redirections.

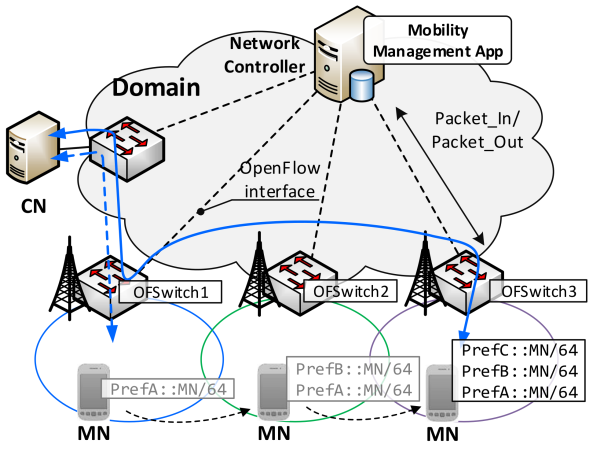


**Fig. 4.** Architecture of SR-DMM proposal

Fig. 4 shows the functional architecture of the SR-DMM solution where the access network consists of OpenFlow switches which are managed by the network controller. In this case, a switch located at the edge of the network (e.g. OF-Switch1) acts as an anchor for the flows opened by the mobile node when connected to this OF-Switch. When the MN moves through the DMM domain, it acquires new IPv6 prefixes from the visited networks.

Clearly, an analogy with the NB-DMM solution can be established: an SDN controller in SR-DMM is a similar entity to CMD (in NB-DMM) and OFSwitches are the entities analogous to the MAARs located on the edge network.

The following subsections show the important tasks of the SR-DMM proposal, such as the initial registration operation and the handover process.

## 3.2. SR-DMM: Initial registration

When an MN connects for the first time to an OFSwitch of the DMM domain, it sends a RS (Router Solicitation) message to this device. This OFSwitch, according to its flow table, encapsulates the RS in an OpenFlow PacketIn message and sends it to the network controller. The MN is authenticated and located by the network controller,which stores a bind with its MAC address, its identifier (MN-ID) and the anchor to which the terminal is currently associated. Then, the network controller retrieves the corresponding network prefix, creates an RA (Router Advertisement) message with this prefix, encapsulates this message into an OpenFlow PacketOut message and finally this message is sent to OF-Switch. When the MN receives the RA, it configures the IPv6 address that anchors to the OFSwitch (see Fig. 5).

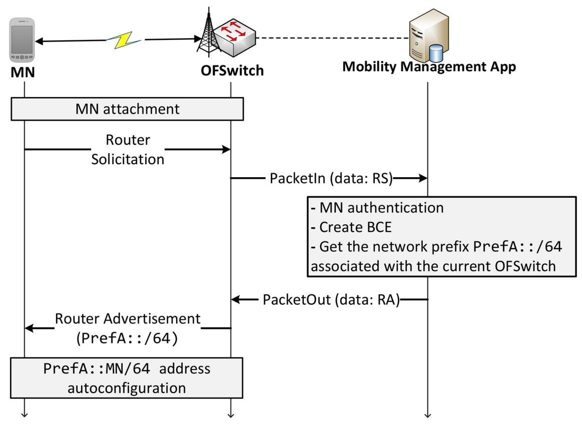

MN
OFSwitch
Mobility Management App
MN attachment
Router Solicitation
PacketIn (data: RS)
- MN authentication
- Create BCE
- Get the network prefix PrefA::/64 associated with the current OFSwitch
PacketOut (data: RA)
Router Advertisement (PrefA::/64)
PrefA::MN/64 address autoconfiguration

**Fig. 5.** SR-DMM proposal: initial registration operation

## 3.3. SR-DMM: Handover operation

During the handover operation, SR-DMM allows the location of the MN to be detected when the network controller receives a new RS message encapsulated into a PacketIn message. In this moment, the network controller de-encapsulates this message and verifies that the current OFSwitch is different from the previous one. If so, the SDN controller generates as many network prefixes as previously visited OFSwitches by the MN during its movement through the DMM domain. With these network prefixes, the mobile node could configure its IPv6 network addresses. This occurs when the MN receives an RA message from its new mobility agent.

The SR-DMM application sets a flow rule on each previous OFSwitch where the flows are anchored by the MN. The creation of flow rules for the OFSwitches visited by the MN is described in detail in Algorithm 1.

**Algorithm 1:** Flow rules calculation for the OFSwitches visited by the MN

```
Input:
   Mobile Node ($MN$),
   Previous OFSwitches list ($OFS_{list}$),
   Current OFSwitch ($COFS$)
Output:
   Flow rules list to install in OFSwitches ($FR$)
1  foreach $PrevOFS \in OFS_{list}$ do
2      // Get IPv6 address from the MN physical address and network prefix of the PrevOFS
3      $MN_{global\_addr} = getIPv6Address(MN_{hw\_addr}, PrevOFS_{prefix})$;
4      // Get local address used to connect PrevOFS and COFS
5      $MN_{local\_addr} = getIPv6Address(COFS, MN_{Id})$;
6      // Build OpenFlow rule to install on OFSwitches
7      $Rule = buildOpenFlowRule(PrevOFS, MN_{global\_addr}, MN_{local\_addr})$;
8      // Create list of all rules
9      $FR[PrevOFS] = Rule$
10 end
11 return $FR$;
```

The installed rule allows packets to redirect from previous anchors to current mobility agents of the MN by establishing an IPv6 destination address which is formed by the network prefix associated with the current OFSwitch and the identifier of the MN in the SR-DMM domain (MN-ID). The flow rule installed on the current mobility agent allows the original IPv6 address of the packets belonging to the previous sessions to be restored in order to maintain the continuity of the sessions transparently. Moreover, the flow rules installed on OFSwitches expire when the time elapses and the corresponding match does not exist. This event is received by the network controller and the expired sessions will not have mobility support (see Fig. 6).

Therefore, SR-DMM allows data traffic to be redirected from the previous OFSwitch to the current one transparently according to the MN location and without involving the MN in the control signaling. This process does not overload the network with the control headers introduced by the IP-IP tunnels.

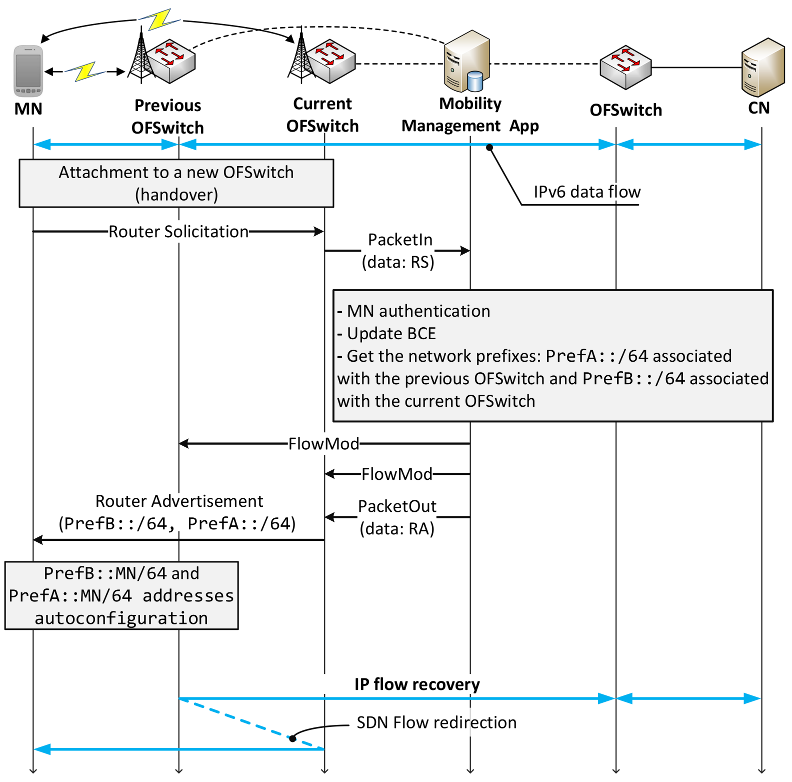

MN
Previous OFSwitch
Current OFSwitch
Mobility Management App
OFSwitch
CN
Attachment to a new OFSwitch (handover)
IPv6 data flow
Router Solicitation
PacketIn (data: RS)
- MN authentication
- Update BCE
- Get the network prefixes: PrefA::/64 associated with the previous OFSwitch and PrefB::/64 associated with the current OFSwitch
FlowMod
FlowMod
Router Advertisement (PrefB::/64, PrefA::/64)
PacketOut (data: RA)
PrefB::MN/64 and PrefA::MN/64 addresses autoconfiguration
IP flow recovery
SDN Flow redirection

**Fig. 6.** SR-DMM proposal: handover process

Other implementations establish an optimal route by applying flow rules to all OFSwitches involved in the path between CN and MN [27]. With SR-DMM it is only necessary to install new rules on the edge OFSwitches visited by the MN because the paths to each destination network are calculated previously and dynamically by the mobility service in order to calculate the optimal route between the previous OFSwitch and the current one. The SR-DMM solution does not need to use any specialized mobility agent, as distinct from other solutions such as [28]. All network devices of the DMM domain are generic and its functionality is established by the network controller, which chooses whether the switches act as mobility agents or not.

## 4. Analytical model

This section presents a cost analysis in terms of signaling cost and data packet delivery cost of our proposed solution and compares them with NB-DMM solution.

The analysis is performed on a domain which consists of N cells connected to different access nodes. These will be the first network devices with IP capability. The network topology is described in Fig. 7.

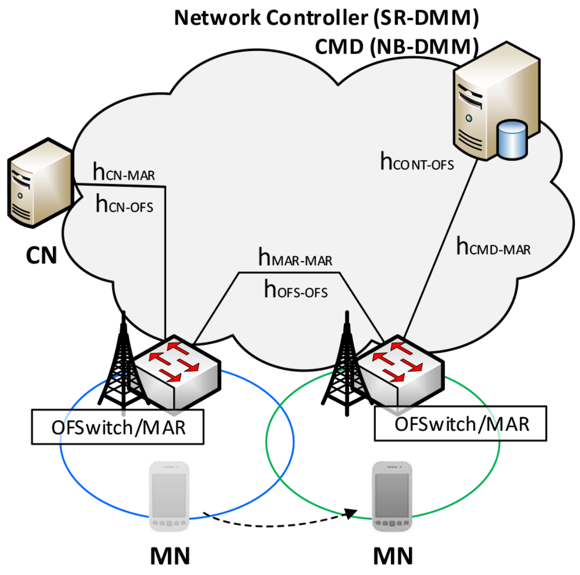


**Fig. 7.** Network topology used in analysis.

NB-DMM and SR-DMM solutions are evaluated through an analytical model in order to calculate signaling cost and packet delivery cost. Both solutions are also analysed on an experimental testbed, which is presented in the following sections.

The analytical model has been developed using the framework described in [15]. The packet transmission cost in IP networks is directly proportional to the number of hops between source and destination nodes. Hence, $h_{x,y}$ is defined as the hop distance between $x$ and $y$ network nodes.

## 4.1. Signaling cost evaluation

One of the main functionalities for any IP mobility management protocol is the process of ensuring that the MNs mobility session is kept up to date while an MN moves among networks. This requires control messages that need to be sent among the mobility agents in the network.

The total signaling cost of registration updates during a session is denoted by $C_s$. The signaling cost is the accumulative traffic load on exchanging signaling messages during the communication session of the MN. This cost depends on the size of the signaling messages and the number of hops in every L3 handover process during the time interval that the MN communication remains active. Therefore, for each movement into a new subnet, the Proxy Binding Update(PBU)/Proxy Binding ACK (PBA) message is sent to the CMD for NB-DMM proposal. Moreover, the binding cache is refreshed during the prefix lifetime (binding refresh process) and the prefix is deleted when there is no active session (deregistration process).

The SR-DMM approach notifies the handover using OpenFlow messages to interact with the forwarding table of the OFSwitches. The signaling control messages exchanged are FlowMod messages, which are typical of SDN architecture. Thus, the path between mobility agents is updated through FlowMod messages. Moreover, inactive network prefixes are removed when the timers of the flow table entries expire. In this case, the binding refresh mechanism is not necessary.

We refer to the total signaling cost as a sum of the three main components: the cost for the binding update after a handover; the cost for terminating a prefix that is no longer active; and the cost required to periodically refresh the bindings. Therefore, the signaling cost during MN movement for both NB-DMM and SR-DMM solutions are summarized in the following expressions:

$$\begin{aligned} C_s^{NB-DMM} = \mu_c \cdot ((S_{PBU} + S_{PBA}) \cdot h_{CMD-MAR} \\ \cdot (N_{pr} + 1) + 2 \cdot (S_{PBU} + S_{PBA}) \cdot h_{CMD-MAR} \\ + R_{BCE} \cdot (S_{PBU} + S_{PBA}) \cdot h_{CMD-MAR}) \end{aligned} \tag{1}$$

$$\begin{aligned} C_s^{SR-DMM} = \mu_c \cdot ((N_{pr} + 1) \cdot 2 \cdot S_{FlMod} \cdot h_{CONT-OFS} \\ + (S_{P-IN} + S_{RS} + S_{P-OUT} + S_{RA}) \cdot h_{CONT-OFS}) \end{aligned} \tag{2}$$

where $\mu_c$ is the subnet (i.e., cell) border crossing rate and $N_{pr}$ is the number of active prefixes per MN. $N_{pr}$ can be also defined as the number of MARs/OFSwitches which maintain some active session with the MN. According to [29], $N_{pr}$ is calculated as:

$$N_{pr} = \frac{\mu_c}{\delta} \tag{3}$$

where $1/\delta$ is the mean value of the active prefix lifetime while the MN is visiting a foreign network.

## 4.2. Packet delivery cost evaluation

The total data packet delivery cost for a session is defined as $C_{pd}$. This value is influenced by the size of the data messages multiplied by the number of hops needed to forward packets from the CN to the MN and vice versa. Thus, the expressions that represent the cost are as follows:

$$\begin{aligned} C_{pd}^{NB-DMM} = N_{p/s} \cdot ((N_{pr} - 1) \cdot (S_{DATA} + S_{IP}) \\ \cdot h_{MAR-MAR} + (S_{DATA} \cdot h_{CN-MAR}) \\ + (S_{DATA} \cdot h_{MN-MAR})) \end{aligned} \tag{4}$$

$$\begin{aligned} C_{pd}^{SR-DMM} = N_{p/s} \cdot ((N_{pr} - 1) \cdot S_{DATA} \cdot h_{OFS-OFS} \\ \cdot h_{OFS-OFS} + (S_{DATA} \cdot h_{CN-OFS}) \\ + (S_{DATA} \cdot h_{MN-OFS})) \end{aligned} \tag{5}$$

where $N_{p/s}$ is the packet transmission rate per active flow and $S_{DATA}$ is the size of these data messages. All parameters used in the analysis are shown in the Notations.

### 4.3. Numerical results

This subsection discusses the performance evaluation of both NB-DMM and SR-DMM proposals. Network topology used for analytical evaluation is shown in Fig. 7. The default values [15] [19] [30] are assumed to be as follows: $E[\mu_c] = [50 - 1800]$ s; $E[\delta] = 60$ s; $E[R_{BCE}] = 60$ s; $N_{p/s} = 3000$; $S_{PBU} = S_{PBA} = 76$ bytes; $S_{RS} = S_{RA} = 52$ bytes; $S_{P-IN} = 92$ bytes; $S_{P-OUT} = 103$ bytes; $S_{FlMod} = 116$ bytes; $S_{DATA} = 120$ bytes; $S_{IP} = 40$ bytes; $h_{CMD-MAR} = h_{CONT-OFS} = 2$ hops; $h_{MAR-MAR} = h_{OFS-OFS} = 2$ hops; $h_{CN-MAR} = h_{CN-OFS} = 5$ hops and $h_{MN-MAR} = h_{MN-OFS} = 1$ hop.

Fig. 8 shows the comparison of signaling cost as a function of the cell residence time, which varies from 50 to 1800 seconds. As could be expected, $C_u$ achieves the highest values when the cell residence time is low.

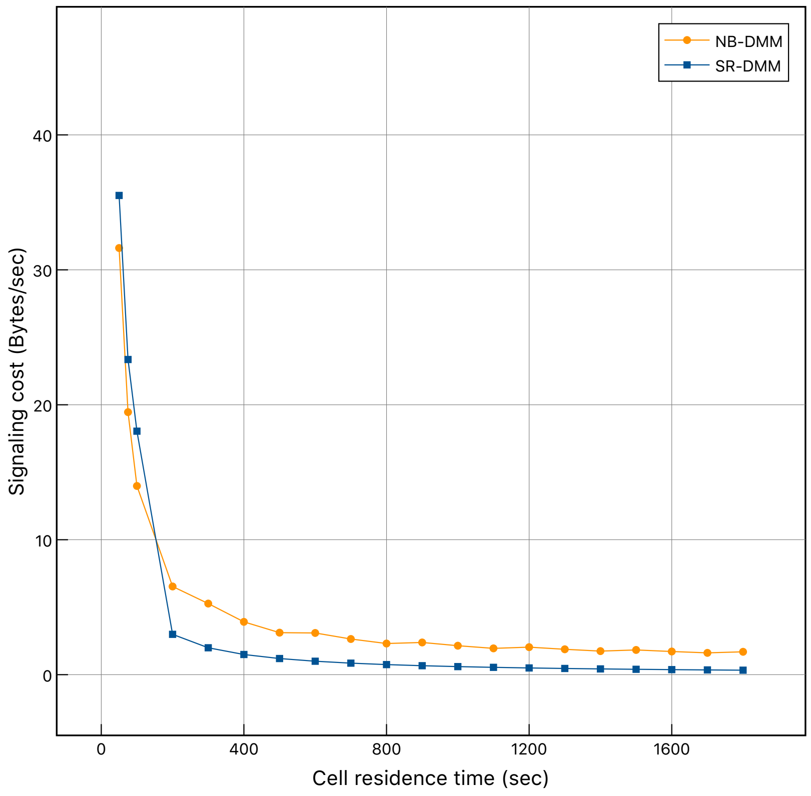


**Fig. 8.** Signaling cost versus cell residence time

When the cell residence time is low, signaling cost for the SR-DMM proposal is greater than the value of NB-DMM solution because NB-DMM is based on IPv6 protocols. Moreover, SR-DMM is an SDN-based proposal and messages exchange is performed through OpenFlow primitives which use TCP protocol.

However, we observe an enhanced performance of our proposal when the value of the cell residence time is increased. SR-DMM benefits from the options included by the SDN paradigm. Thus, our proposal does not require the binding refresh process and its signaling cost is lower than the signaling cost introduced by NB-DMM solution.

On the other hand, data packet delivery cost represents the cost of delivering data packets to an MN per unit time. Fig. 9 depicts the packet delivery cost as a function of the cell residence time. As can be observed, NB-DMM solution implies IP-IP tunneling including an IPv6 header between the previous and current mobility agents. However, with our proposal, additional IPv6 headers are not introduced. This is an important advantage on the data plane of the DMM-based solutions. Clearly, the SR-DMM proposal obtains the best results, optimizing both the control plane and data plane.

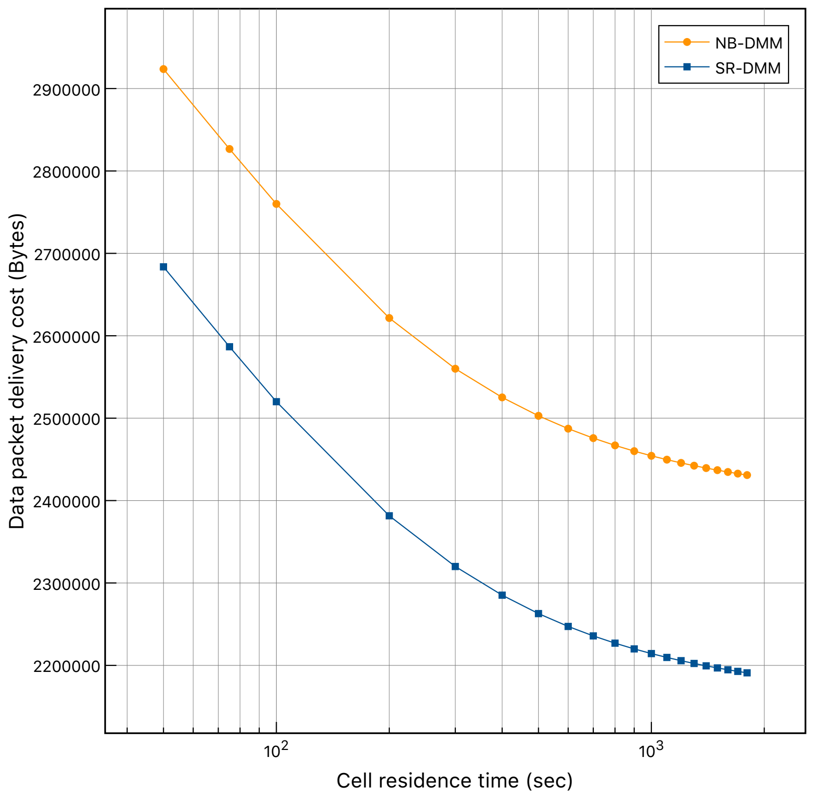


**Fig. 9.** Packet delivery cost versus cell residence time

## 5. Experimental evaluation

This section reports on the experimental evaluation conducted using real implementations of NB-DMM and our solution SR-DMM. The testbed deployed to perform the experiments of both solutions is depicted in Fig. 10. On the one hand, NB-DMM has been analyzed using a DMM implementation for GNU/Linux systems called MAD-PMIPv6 (Mobility Anchors Distribution for Proxy Mobile IPv6) [31]. This solution is written in C and runs on the Linux kernel. On the other hand, the control plane of SR-DMM is implemented through Ryu framework for SDN environments, using OpenFlow protocol as a control interface. Ryu fully supports IPv6. Thus, the SR-DMM service is deployed as an application on the network controller, which is running with Ryu.

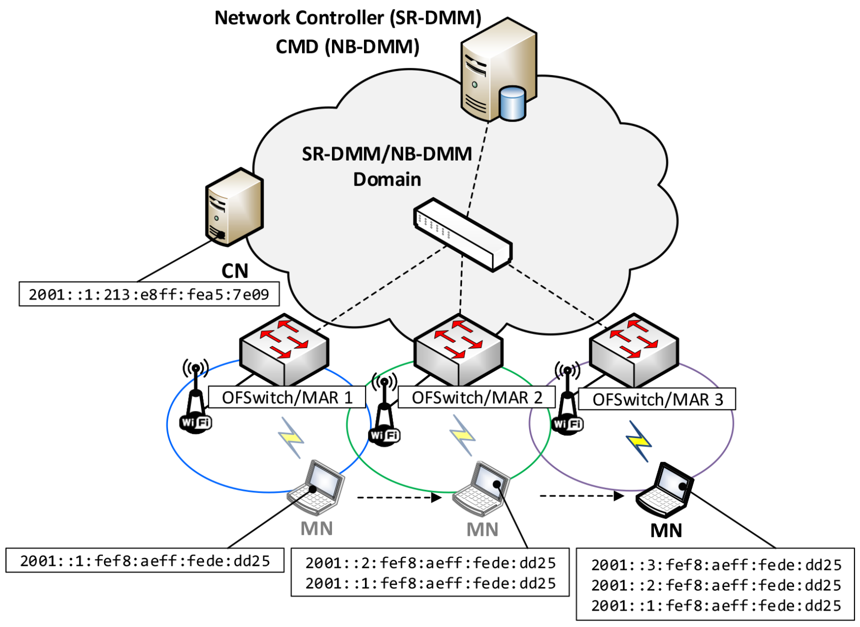

Network Controller (SR-DMM)
CMD (NB-DMM)
SR-DMM/NB-DMM
Domain
CN
2001::1:213:e8ff:fea5:7e09
OFSwitch/MAR 1
OFSwitch/MAR 2
OFSwitch/MAR 3
MN
MN
MN
2001::1:fef8:aeff:fede:dd25
2001::2:fef8:aeff:fede:dd25
2001::1:fef8:aeff:fede:dd25
2001::3:fef8:aeff:fede:dd25
2001::2:fef8:aeff:fede:dd25
2001::1:fef8:aeff:fede:dd25

**Fig. 10.** Network topology used in experimental evaluation

Both solutions have been evaluated on the same network topology. This access network consists of three OFSwitches, which provide access to the MNs through wireless interfaces by using IEEE 802.11g technology. A network prefix (64 bits) is associated with each OFSwitch/MAR. This prefix is calculated by the SR-DMM application on the network controller. However, in the NB-DMM implementation, the prefix is calculated by the MAR. Moreover, a CN is responsible for sending flows to the MN. SR-DMM does not require the TCP/IPv6 stack of all devices to be updated. However, MARs and CMD, in NB-DMM solution, run with a compiled Linux kernel with mobility features.

The experimental evaluation focuses on the handover latency. This parameter is defined as the time interval in which an MN does not have IP connectivity as a result of a handover process, which is caused by the nature of the mobility when a MN changes its point of attachment to the network and a disruption time exits. In fact, the number of packets lost during a handover is directly proportional to the handover latency.

In our experimental testbed, we use Wireshark at the MN to extract the events produced when repeating the following sequence: the MN attaches to OFSwitch/MAR 1 and CN starts an UDP stream to the MN. Then, the MN visits OFSwitch/MAR 2 and OFSwitch/MAR 3 before coming back to OFSwitch/MAR 1.

This experiment is repeated obtaining more than 600 handovers for both solutions. Fig. 11 depicts the empirical CDF for the values of the UDP stream recovery time in the testbed for SR-DMM and NB-DMM.

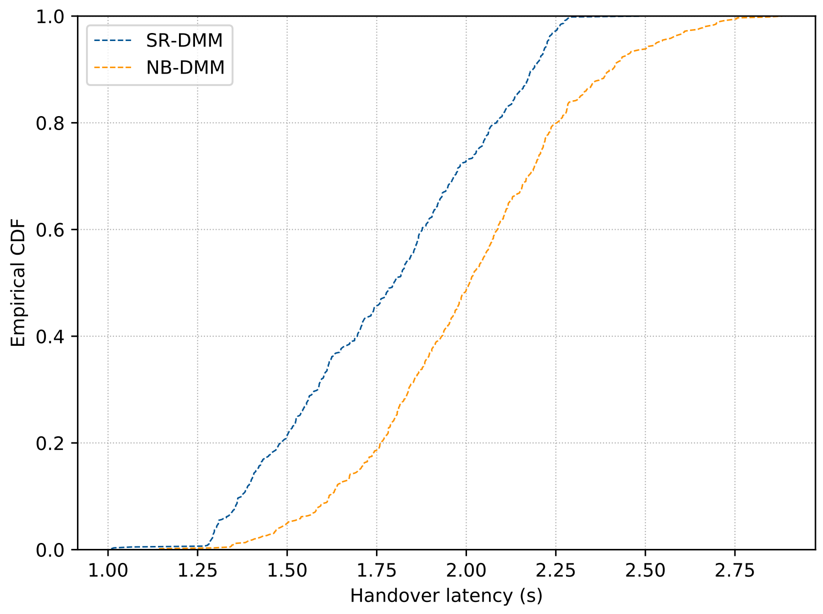


**Fig. 11.** Empirical CDF of the handover measurements

Moreover, Table 1 reports the mean and standard deviation values of these results for SR-DMM and NB-DMM proposals.

**Table 1.** SR-DMM handover latency measurements

| | Mean (s) | Std.Dev. (s) |
|---|---|---|
| SR-DMM | 1.78 | 0.29 |
| NB-DMM | 2.01 | 0.31 |

As can be seen in Table 1, the average handover latency is 1.78 seconds in SR-DMM tests. However, with NB-DMM the average handover latency is higher (2.01 seconds). During this handover disruption time, the mobile node cannot receive IPv6 packets until the session is restored by the SR-DMM/NB-DMM service.

Finally, other tests conducted consisted in measuring the different components of the handover latency while the MN roams among the three OFSwitches/MARs of our testbed (see Fig. 10). UDP traffic has been used in this experiment. We have measured three handover events which are detailed as follows:

– L2: the Layer-2 handover is the time required to perform an L2 switch from one OFSwitch/MAR to another. We measured this as the interval between two IEEE 802.11 control messages.
– LSDN: time between when the MN sends an RS to the OFSwitch in the visited network and the MN receives an RA with all IP parameters. NB-DMM does not spend LSDN time, because it does not use SDN mechanisms.
– L3: is measured as the interval between the last data packet received by the MN before the handover and the first data packet received or sent after the handover.

Fig. 12 explores in detail the components of the handover latency for the SR-DMM and NB-DMM solutions.

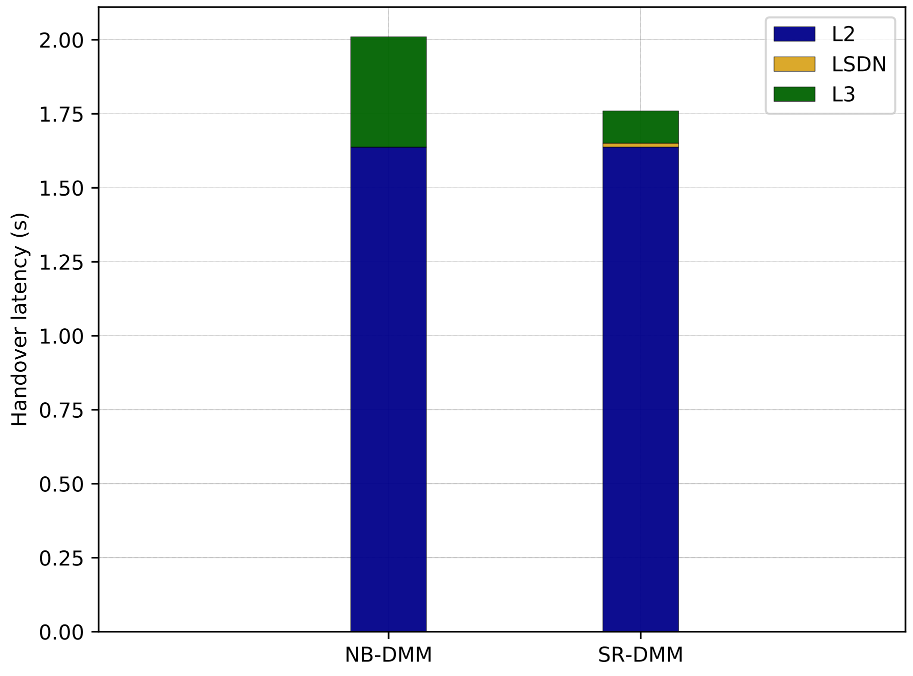


**Fig. 12.** Handover latency composition

As can be seen from the results, the L2 switch time is the major term for both proposals. The technology used in this level is IEEE 802.11g. The processing time of the network controller in SR-DMM (LSDN) is between 0.3 and 0.7 percent of the total handover latency. However, NB-DMM does not spent LSDN time because it does not require any SDN mechanisms. Moreover, IP flow recovery (L3) depends on types of solution: the NB-DMM solution uses 18% of the handover time and SR-DMM only uses 6%. The SR-DMM offers benefits due to SDN capabilities compared with NB-DMM.

## 6. Conclusions

Software-Defined Networking brings a natural solution to decouple the network control plane from the data plane for 5G environments, where Distributed Mobility Management is seen as a necessary paradigm in future mobile network deployments in order to flatten the network.

In this context, this article is focused on an analytic and experimental evaluation of an SDN-based DMM solution, called SR-DMM. The main objective is to provide flexibility and scalability to the mobility process in next generation networks. In order to compare the performance of our proposals with other legacy DMM solution such as NB-DMM, we have formulated an analytical model that reveals the benefits that SR-DMM brings to the network's performance in terms of signaling and packet delivery cost. This improvement can be achieved due to the implementation of SR-DMM, that reduces the complexity of the data plane and the tunnel management avoiding the use of IP-IP tunnels during the movement of the users. In addition, an experimental evaluation of SR-DMM and NB-DMM have been conducted in a real scenario. The results show that the own mechanisms of SDN introduce a minimum latency in the handover process. It also demonstrates that SDN can alleviate the complexity of mobility management and will be a key concept in the design of future mobile networks.

SR-DMM provides a simple implementation of the mobility management protocol without modifying the network nodes. Moreover, the data plane is simplified by avoiding the use of IP-IP tunnels. Thus, the packet delivery cost is improved with our proposal. On the other hand, in the experimental evaluation, SR-DMM also presents an enhanced scalability due to the global vision of the network controller. The SR-DMM offers benefits due to SDN capabilities compared to the legacy DMM approaches.

## Notations

$h_{x-y}$: Average hop distance between $x$ and $y$ nodes.
$N_{pr}$: Number of used active prefixes.
$N_{p/s}$: Packet transmission rate per active flow.
$R_{BCE}$: Rate of BCE refresh operations.
$S_{DATA}$: Size of a data packet.
$S_{PBU}$: Size of the PBU message.
$S_{PBA}$: Size of the PBA message.
$S_{FlMod}$: Size of the FlowMod message.
$S_{P-IN}$: Size of the PacketIn message.
$S_{P-OUT}$: Size of the PacketOut message.

$S_{RS}$: Size of the Router Solicitation message.
$S_{RA}$: Size of the Router Advertisement message.
$S_{IP}$: Size of the IPv6 tunnel header.
$\mu_c$: Subnet border crossing rate.

**Acknowledgments.** This work was supported in part by the Regional Fund, through Computing and Advanced Technologies Foundation of Extremadura (COMPUTAEX), European Regional Development Fund, and the Regional Ministry of Economy and Infrastructure of the Junta de Extremadura under project IB18003.

## References

1. Cisco Systems Inc. Cisco Visual Networking Index: Forecast and Trends, 2017-2022. White Paper. February 2019.
2. Nokia Siemens Networks. Signaling is growing 50% faster than data traffic. White Paper, 2012.
3. I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali and A. Habbal. Caching in Information-Centric Networking: Strategies, Challenges, and Future Research Directions. In *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1443-1474, 2018.
4. O. A. Khan, M. A. Shah, I. Ud Din, B. Kim, H. A. Khattak, J. J. P. C. Rodrigues, H. Farman and B. Jan. Leveraging Named Data Networking for Fragmented Networks in Smart Metropolitan Cities. In *IEEE Access*, vol. 6, pp. 75899-75911, 2018.
5. B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng, J. Pérez-Romero, I. Trajkovska, P. S. Khodashenas, L. Goratti, M. Paolino, E. Sfakianakis, F. Liberal, and G. Xilouris. Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN. Computer Standards and Interfaces. In *Computer Standards and Interfaces*, vol. 54, pp. 216-228, 2017.
6. C. Perkins, D. Johnson, and J. Arkko. Mobility Support in IPv6. IETF RFC 6275, July 2011.
7. S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. Proxy Mobile IPv6. IETF RFC 5213, August 2008.
8. J. Carmona-Murillo, I. Soto, F.-J. Rodríguez-Pérez, D. Cortés-Polo, and J.-L. González-Sánchez. Performance Evaluation of Distributed Mobility Management Protocols: Limitations and Solutions for Future Mobile Networks. In *Mobile Information Systems*, vol. 2017, Article ID 2568983, 15 pages, 2017.
9. J. Carmona-Murillo, V. Friderikos, and J.-L. González-Sánchez. A hybrid DMM solution and trade-off analysis for future wireless networks. In *Computer Networks*, vol. 133, pp. 17-32, 2018.
10. H. Chan, D. Liu, P. Seite, H. Yokota, and J. Korhonen. Requirements for Distributed Mobility Management. IETF RFC 7333, August 2014.
11. Shariq Haseeb and Ahmad Faris Ismail. Handoff latency analysis of mobile IPv6 protocol variations. In *Computer Communications*, Volume: 30, Issue: 4, 2007.
12. K. N. Ashraf, V. Amarsinh and D. Satish. Survey and analysis of mobility management protocols for handover in wireless network. In *Proc. 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, 2013*, pp. 413-420, 2007.
13. Mun-Suk Kim, SuKyoung Lee, David Cypher, and Nada Golmie. Performance analysis of fast handover for proxy Mobile IPv6. In *Information Sciences*, Volume: 219, pp. 208-224, 2013.
14. M.K. Murtadha, N.K. Noordin, B.M. Ali, and F. Hashim. Design and evaluation of distributed and dynamic mobility management approach based on PMIPv6 and MIH protocols. In *Wireless Networks*, Volume: 21, Issue: 8, pp. 2747-2763, 2015.

15. C. Makaya and S. Pierre. An Analytical Framework for Performance Evaluation of IPv6-Based mobility Management Protocols. In *IEEE Transactions on Wireless Communications*, 7(3), 972983, March 2008.
16. Qiang Wu, Chun-Ming Wu, and Wen Luo. Distributed mobility management with ID/locator split network-based for future 5G networks. In *Telecommunication Systems*. 2018.
17. A. N. Toosi, R. Mahmud, Q. Chi, and R. Buyya. Management and Orchestration of Network Slices in 5G, Fog, Edge, and Clouds. In *Fog and Edge Computing*. 2019.
18. I.U. Din, H. Asmat, and M. Guizani. A review of information centric network-based internet of things: communication architectures, design issues, and research opportunities. In *Multimedia Tools and Applications*, 1-16. 2018.
19. Jong-Hyouk Lee, Jean-Marie Bonnin, Ilsun You, and Tai-Myoung Chung. Comparative Handover Performance Analysis of IPv6 Mobility Management Protocols. In *Transactions on Industrial Electronics, IEEE*, Volume: 60, Issue: 3, March 2013.
20. Hassan Ali-Ahmad, Meryem Ouzzif, Philippe Bertin, and Xavier Lagrange. Performance Analysis on Network-Based Distributed Mobility Management. In *Wireless Personal Communications*, 74(4):1245-1263, 2014.
21. M. M. Sajjad, D. Jayalath, and C. J. Bernardos. A Comprehensive Review of Enhancements and Prospects of Fast Handovers for Mobile IPv6 Protocol. In *IEEE Access*, vol. 7, pp. 4948-4978, 2019.
22. I. Ullah, Z. Shah, and A. Baig. S-TFRC: An Efficient Rate Control Scheme for Multimedia Handovers. In *Computer Science and Information Systems*, vol. 13, no. 1, pp. 45-69, 2016.
23. Enio Kalji, Almir Maric, Pamela Begovic, and Mesud Hadzialic. A Survey on Data Plane Flexibility and Programmability in Software-Defined Networking. In *IEEE Access*,vol. 7, pp. 47804-47840, 2019.
24. M. He, A. M. Alba, A. Basta, A. Blenk, and W. Kellerer. Flexibility in softwarized networks: Classifications and research challenge. In *IEEE Communications Surveys Tutorials*, 2019.
25. K. Benzekki, A. El Fergougui, and A Elbelrhiti Elalaoui. Software-defined networking (SDN): a survey. In *Security and Communications Networks*, 2017.
26. S. Moazzeni, M. Reza Khayyambashi, N. Movahhedinia, and F. Callegati. On reliability improvement of Software-Defined Networks. In *Computer Networks*,vol. 133. pp. 195-211. 2018.
27. T.-T.Nguyen, C. Bonnet and J. Harri. SDN-based distributed mobility management for 5G networks. In *2016 IEEE Wireless Communications and Networking Conference*, pp. 17, April 2016.
28. F. Giust, L. Cominardi, and C. Bernardos Distributed mobility management for future 5G networks: overview and analysis of existing approaches. In *IEEE Communications Magazine*, vol. 53, no. 1, pp. 142 149, January 2015.
29. F. Giust, C. J. Bernardos, and A. De La Oliva Analytic Evaluation and Experimental Validation of a Network-based IPv6 Distributed Mobility Management Solution. *IEEE Transactions on Mobile Computing*, vol. 13, no. 11, pp. 2484-2497, 2014.
30. David Cortés-Polo, Jesús Calle-Cancho, Javier Carmona-Murillo, and José-Luis González-Sánchez. Future Trends in Mobile-Fixed Integration for Next Generation Networks: Classification and Analysis. In *International Journal of Vehicular Telematics and Infotainment Systems*, Volume: 1, 33-53, January 2017.
31. CJ. Bernardos et al. A PMIPv6-based solution for Distributed Mobility Management. draft-bernardos-dmm-pmip-09, 2017.

**Jesús Calle-Cancho** studied at the University of Extremadura, Spain, where he received his Beng (2011) and MEng (2013) in Computer Science Engineering. He is currently working as Supercomputing System Manager at the COMPUTAEX Foundation and the

CénitS center. His main area of research interest is IPv6 mobility management in next generation wireless networks.

**José-Manuel Mendoza-Rubio** received his Meng in Telematics Engineering (2017) at University of Extremadura. His main areas of interest are software-defined networks and mobility management in next-generation networks.

**José-Luis González-Sánchez** is a full time Associate Professor of the Computing Systems and Telematics Engineering department at the University of Extremadura, Spain. He received his Engineering degree in Computer Science and his Ph.D in Computer Science (2001) at the Polytechnic University of Cataluña, Barcelona, Spain. He has worked for years at several private and public organizations as a System and Network Manager. Currently he is also the General Manager of CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura).

**David Cortés-Polo** received the Computer Science degree and a Ph.D. in Telematics in 2015, both from University of Extremadura, Spain where he worked as a research and teaching assistant from 2011 to 2014. Since 2011, he has worked as Network Manager at COMPUTAEX Foundation. His main research interests include IP-based mobility management protocols, performance evaluation and quality of service support in future mobile networks.

**Javier Carmona-Murillo** received the Computer Science degree (with honors) and a Ph.D. in Telematics in 2015, both from University of Extremadura, Spain where he worked as a research and teaching assistant from 2005 to 2009. Since then, he has worked as an assistant professor at the Department of Computing and Telematics System Engineering in the same university. His main research interests include IP-based mobility management protocols, performance evaluation and quality of service support in future mobile networks.

*Received: November 1, 2018; Accepted: July 18, 2019.*

Computer Science and Information Systems 17(1):71–92 https://doi.org/10.2298/CSIS190518044O

# Instance-based classification using prototypes generated from large noisy and streaming datasets

Stefanos Ougiaroglou[1,2], Dimitris A. Dervos[1], and Georgios Evangelidis[2]

[1] Department of Information and Electronic Engineering, International Hellenic University, GR-57400, Sindos, Thessaloniki, Greece
stoug@uom.edu.gr,dad@it.teithe.gr

[2] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Str., GR-54006, Thessaloniki, Greece
gevan@uom.gr

**Abstract.** Nowadays, large volumes of training data are available from various data sources and streaming environments. Instance-based classifiers perform adequately when they use only a small subset of such datasets. Larger data volumes introduce high computational cost that prohibits the timely execution of the classification process. Conventional prototype selection and generation algorithms are also inappropriate for data streams and large datasets. In the past, we proposed prototype generation algorithms that maintain a dynamic set of prototypes and are appropriate for such types of data. Dynamic because existing prototypes may be updated, or new prototypes may be appended to the set of prototypes in the course of processing. Still, repetitive generation of new prototypes may result to forming unpredictably large sets of prototypes. In this paper, we propose a new variation of our algorithm that maintains the prototypes in a convenient and manageable way. This is achieved by removing the weakest prototype when a new prototype is generated. The new algorithm has been tested on several datasets. The experimental results reveal that it is as accurate as its predecessor, yet it is more efficient and noise tolerant.

**Keywords:** $k$-NN classification, Data reduction, Prototype generation, Data streams, Large datasets, Noisy data.

## 1. Introduction

Classification is a fundamental concept in data mining [10]. Many classification algorithms have been proposed in the last half century [12]. They aim at accurate class prediction of the unclassified instances on the basis of a set of already classified instances (training set). The quality of the training set as well as its size determine the efficiency and the effectiveness of the algorithm and consequently they are vital for all classifiers.

Handling training data-streams [1] and large training sets in classification systems has attracted the interest of the data mining research community. The goal is the reduction of the high computational cost involved. The problem is more intense in cases of instance-based classifiers because the whole training set should be examined for each unclassified instance in order to classify it. In addition, executing classification algorithms on devices with limited memory (e.g., sensors) is also an important issue, since otherwise, transferring data to powerful servers for processing is inevitable. In both cases, a simple and obvious approach is the use of a subset of the available data. However, this subset probably cannot represent the whole training set and may harm the classification accuracy.

Instance-based classification is characterized to comprise a lazy learner algorithm because instead of involving a discriminative function it directly processes unclassified instances against the training dataset. The k-Nearest Neighbors ($k$-NN) [6] classifier is a typical example of a lazy learner. When an unclassified instance is to be classified, the $k$-NN classifier identifies and retrieves the k nearest instances from the available training set on the basis of a pre-specified distance metric; most often, the Euclidean distance. The nearest instances are called neighbors. The unclassified instance is assigned to the class that dominates in its k-nearest neighbor set. This task is very simple. However, since all the training set members need to always be available in memory and all distances between the unclassified instance and the training data have to be computed, the approach comprises a memory and CPU intensive task. Yet another drawback is that the $k$-NN classifier is not noise-tolerant. Noise affects the classification process and reduces accuracy.

Data Reduction Techniques (DRTs) [9,24] can remedy the stated drawbacks by introducing a preprocessing stage to the training dataset. A set of representative prototypes is created (usually called a condensing set) from the initial training data. Although there are few exceptions [21], by definition, DRTs cannot handle data of an incremental nature (data streams), as well as large dataset that do not fit in the main memory. To overcome these drawbacks, we recently proposed the dynamic RHC (dRHC) [19], and the abstraction IB2 (AIB2) [17,18] algorithms. Both construct/maintain a dynamic condensing set from the data stream or large data set. Dynamic because the algorithms update continuously the condensing set by incrementally considering new instances as the latter emerge.

Both dRHC and AIB2 operate on a small set of prototypes without seriously degrading the accuracy achieved by the $k$-NN classifier that operates on the whole of the original training set. Yet, they both introduce an unpleasant phenomenon that constitutes the major motivation behind this work: as new prototypes are generated and they get appended to the condensing set, the size of the latter may exceed the size of the available memory. A second motive is that both algorithms are not noise tolerant. False new prototypes may be generated as a result of the noise present in the initial training set. Consequently, there is a clear need to filter out the noise present in the training set, and to maintain the size of the condensing set under control, up to a user specified threshold value.

In [16], we made a first attempt to address the aforementioned issues. In particular, we proposed dRHC-V2, which is a variation of dRHC. Contrary to dRHC, dRHC-V2 keeps the size of the condensing set in a convenient, manageable by the classifier, level by ranking the prototypes and removing the least important ones. As soon as the condensing set size exceeds a user-specified threshold value, the condensing set prototypes are ranked on a calculated weight value, and the necessary number of low ranked prototypes are removed from the condensing set in order to maintain the set threshold. The experimental measurements presented in [16] show that dRHC-V2 filters the noisy data and keeps the classification accuracy at high levels. However, dRHC-V2 is not a one pass algorithm. It utilizes a ranking procedure that may not cope well with fast data streams. Even a quick sort algorithm may be inappropriate for very fast data streams, where new instances arrive at fast rates. This is another motive of the present work.

We propose a new variation of AIB2 called AIB2-V2. Like dRHC-V2, AIB2-V2 incorporates a mechanism for prototype weighting and removal. However, AIB2-V2 is of a more proactive nature. Weight values are calculated again as in the case of dRHC-V2 and as a new prototype enters the condensing set, the weakest (i.e., the one with the lowest weight) of the existing prototypes is removed from the set. In other words, the newly generated prototype replaces the weakest one. Contrary to dRHC-V2, AIB2-V2 does not rank the prototypes. The prototype with the minimum weight is removed. Noisy prototypes tend to involve low weight values. Hence, they are the first to be removed as new prototypes get appended to the (dynamically updated) condensing set. The experimental results show the new algorithm to be faster and more noise-tolerant than AIB2, with no sacrifice in accuracy and computational complexity.

The paper is organized as follows: In section 2, the fundamental issues about DRTs and their limitations are briefly presented. The section also includes a recap of the AIB2, dRHC and dRHC-V2 algorithms. Section 3 considers in detail the AIB2-V2 algorithm. In Section 4, the latter is experimentally compared to dRHC-V2 and to their predecessors on sixteen datasets. The experimental study is complemented with a statistical validation using the Wilcoxon signed rank test [7,23]. Section 5 suggests new directions for future work and concludes the paper.

## 2. Background knowledge

### 2.1. Data Reduction Techniques

DRTs [9,24] pre-process the training data and construct a set of prototypes that is then used by the $k$-NN classifier. DRTs can be grouped into two categories:

- Prototype Selection (PS) algorithms [9] select prototypes from the initial training set. PS algorithms can be also grouped into two subcategories:
  - PS-editing algorithms aim to remove noise from the training set and to "clean" the borders between classes. This way, the classification accuracy is improved.
  - PS-condensing algorithms aim for data condensation, i.e., the construction of a small set of prototypes (condensing set) that represents the initial training data.
- Prototype Generation (PG) algorithms [24] generate prototypes by summarizing on instances. As in the case of PS-condensing algorithms, the goal is data condensation.

Most PS-condensing and PG algorithms construct condensing sets by removing "internal" instances. These instances do not determine the borders between the classes and can be removed without accuracy loss. Thus, PS-condensing algorithms try to select only the instances that are close to the borders. These instances are called close-border instances and are essential to classification. PG algorithms generate many prototypes for the close-border areas and few for the "internal" areas. Unfortunately, most of PS and PG algorithms cannot handle noise and this leads to lower data reduction rates. In the case of PS-condensing algorithms, an instance with wrong class label is considered as close-border instance and it is erroneously included in the condensing set, along with its neighboring instances. In the case of PG algorithms, more prototypes are generated for noisy data areas because neighboring instances of different classes cannot be summarized. Consequently, for training sets with noise, editing should be applied beforehand.

Although there are some exceptions (e.g., the IBL algorithms [3,5]), DRTs are memory-based. All training data are assumed to reside in main memory. Hence, DRTs are unsuitable for large datasets that cannot fit into memory and they cannot be used on devices with limited memory. In addition, DRTs are appropriate only for static datasets. Except for few exceptions [21], they cannot process new incoming data, after the construction of the condensing set. Equivalently, they cannot dynamically update the condensing set. Suppose that a DRT constructs a condensing set by considering a training set $TS$. Also, suppose that new training data $D$ arrive. For the construction of an updated version of the condensing set, the DRT needs to operate from scratch on the complete training set $TS \cup D$. This means that the entire training instances set need be considered. Hence, DRTs are inappropriate for data streams where new training instances become gradually available. The dRHC [19] and AIB2 [17,18] algorithms are PG algorithms that can be used in such environments.

## 2.2. Review of dRHC and dRHC-V2

Dynamic RHC (dRHC) constitutes a descendant of the Reduction through Homogeneous Clusters (RHC) algorithm [19]. Inherent to the latter is the concept of cluster homogeneity. RHC utilizes $k$-means clustering. Suppose that a training set $C$ contains $D$ classes. Initially, the whole training set is considered as an unprocessed non-homogeneous cluster. RHC averages out over the instances of each class in $C$ and calculates a mean (centroid) for each class. Then, $k$-means runs over $C$ by using these class-centroids as initial seeds. The result is $D$ clusters. Each instance in $C$ is assigned to one of the $D$ clusters. Subsequently, the $D$ clusters are considered. For each one homogeneous cluster (i.e., involving instances of only one class), the cluster-mean comprises a representative prototype placed in the condensing set. On the other hand, for each non-homogeneous cluster, the algorithm proceeds recursively. When no non-homogeneous clusters are left, RHC terminates. This way, each homogeneous cluster contributes a (representative) prototype to the condensing set. Like most PG algorithms, RHC generates few prototypes for the "internal" data areas and more prototypes for close-border data areas. Like most DRTs, RHC is memory-based and as such it cannot manage data that cannot fit in memory and data streams.

The dRHC algorithm can manage large and / or streaming datasets. This is accomplished by considering the training data in the form of data segments. The size of the data segment is fixed and it can be adjusted according to the available memory. In the case of data streams, dRHC utilizes a buffer the size of the data segment that is set to accept incoming instances. When the buffer gets full, its content is moved forward for processing. Analogously, large datasets that cannot fit into memory are divided into equally sized data segments appropriate to the device's memory, and the latter are processed sequentially. The algorithm includes two main stages: stage 1 is the *initial condensing set construction*. It is executed only once utilizing the first data segment. It is identical to RHC with one difference: each one prototype is stored alongside with a weight attribute value equal to the number of instances it represents. Stage 2 is the *condensing set update*. It is executed following the arrival of data segment number two and onwards. It processes the prototypes of the current condensing set against the instances of a new data segment and constructs a new set of clusters. It then proceeds in a way similar to that of RHC.

Figure 1 illustrates an example of the execution of the *condensing set update* stage. Sub-figure (a) presents an existing condensing set that has three prototypes with the corresponding weights. Suppose that a new segment with seven new instances arrives (Sub-figure (b)). Each new instance carries a weight value equal to one. At first, each new instance is assigned to the cluster of the nearest prototype (Sub-figure (c)). No instance has been assigned to cluster B. Hence, the corresponding prototype remains unchanged. The instances assigned to cluster A are of the same class as the class of the prototype in A. Thus, A remains homogeneous. Therefore, the weighted mean in A is computed and it is placed in the condensing set alongside with its new weight value. In effect, the prototype "moves" towards the new instances to represent better this data area (Sub-figure (d)). Cluster C becomes non-homogeneous after the assignment of the new instances. For each discrete class in C, the algorithm computes a weighted class mean and executes $k$-means. Two new homogeneous clusters emerge (Sub-figures (d) and (e)). Eventually, dRHC computes a weighted cluster mean for each cluster as well as the corresponding weights. They constitute new prototypes and they are placed in the condensing set (Sub-figure (f)).

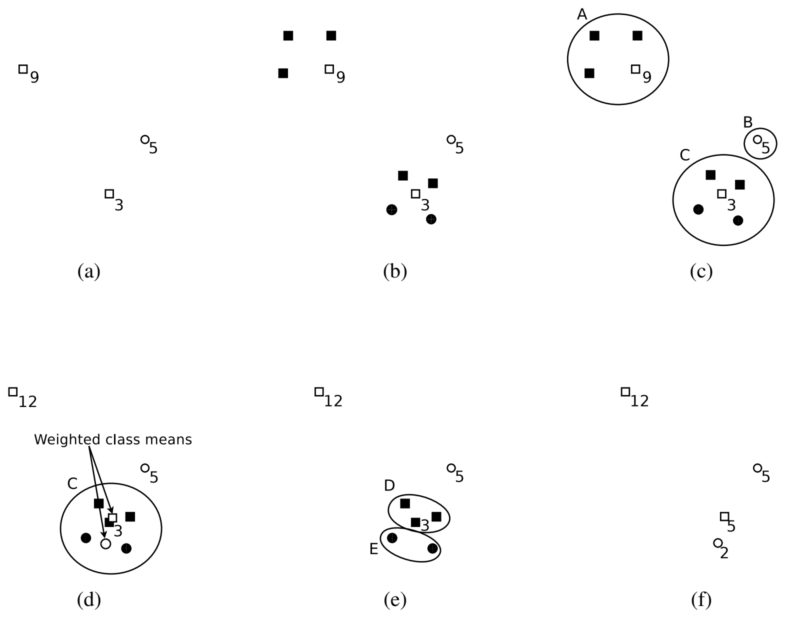


**Fig. 1.** Example of execution of the *condensing set update* stage of dRHC

RHC and dRHC are compared to each other and against state-of-the-art PS [11,3,14,26] and PG [22] algorithms in [19]. The results obtained reveal that dRHC involves the lowest preprocessing cost (i.e., it is the fastest to execute) and constructs the most compact condensing set without sacrificing accuracy. A week point is that noise in the training data results into having a larger number of non-homogeneous clusters which, consequently, leads to a lower reduction rate and to a higher preprocessing cost during the prototypes construction stage.

The dRHC-V2 algorithm constitutes a dRHC variation that retains the size of the condensing set within acceptable levels. The desirable maximum size of the condensing set comprises a user-specified input parameter ($T$). dRHC-V2 executes in a way similar to dRHC with one difference: as soon as the size of the condensing set exceeds the set number of $T$ prototypes, the least important of the latter are removed from the condensing set, to save space. In effect, the mechanism for prototypes removal comprises a post-processing step of the *condensing set update* stage. The input parameter $T$ is set by considering the available memory, the noise, and the desirable trade-off between computational cost and accuracy.

The stated functionality is implemented by having dRHC-V2 rank the prototypes according to their importance, following the execution of each *condensing set update* stage. Next, only the top $T$ prototypes are retained. The prototype weights are vital for the ranking procedure. It is reminded that the prototype weight depicts the number of instances represented by the prototype in question. A straightforward approach could be to remove the necessary number of prototypes with the lowest weights. However, this would not comprise a fair criterion. An old prototype probably has a higher weight value than that of a prototype generated at a subsequent *condensing set update* stage. Consequently, prototypes generated at later stages would be prone for removal as compared to older ones. Also, if the weight is adopted to comprise the sole ranking criterion, an old prototype of high weight that has survived many executions of the *condensing set updates* will probably tend to be favored to survive against all recently generated prototypes and will thus remain permantly in the condensing set. Therefore, prototype weight should not by itself comprise the sole contributor to the calculation of the prototype's rank measure.

To achieve fairness in prototype ranking, dRHC-V2 takes into consideration the weight as well as the age of the prototype. Thus, dRHC-V2 holds a counter of the data segments that have been processed at any one given instance in time. Each prototype has an extra attribute that denotes the number ($r$) of the data segment corresponding to its generation. In this respect, $r$ depicts the instance in time when the prototype in question got appended to the condensing set. Following the completion of each *condensing set update* stage, dRHC-V2 re-calculates the rank measure for all the condensing set prototypes. The measure is called Average number of Arrivals ($AnA$). This measure incorporates the prototype's weight ($w$) and age ($r$) contributors. More specifically, the prototype's $AnA$ rank measure is calculated as follows:

$$AnA = \frac{w}{ds - r + 1}$$

where $ds$ is the (sequence) number of the current data segment, $r$ is the number of the data segment corresponding to the prototype's generation, and $w$ is the prototype's weight value, i.e. the number of instances it represents. In other words, the denominator reflects the age of the prototype. A prototype is weak when it has low $AnA$ value.

Considering the above, prototypes generated by the latest *condensing set update* stage have $AnA$ values equal to their weight ($w$). For existing prototypes, their $Ana$ values are (re-)calculated by dividing their weights $w$ by their (updated) age values. Prototypes updated to represent new instances during the *condensing set update* stage have their $AnA$ value (re-)calculated by diving their new weight $w$ value by their (updated) age value.

The post-processing step of dRHC-V2 executes following the completion of the *initial condensing set construction* stage, as well as the execution of each one subsequent *condensing set update* stage. It operates on an already constructed condensing set, with given a maximum condensing set size $T$. When the size of the updated condensing set is found to exceed the set maximum $T$, the training set is trimmed to only contain the top $T$ prototypes with the highest $AnA$ values.

Noisy prototype instances present in the dRHC generated condensing sets usually relate to low $AnA$ values in dRHC-V2. As such, they are weak and comprise the first candidates to be removed when the $T$ threshold value is reached in dRHC-V2. Thus, contrary to dRHC, dRHC-V2 can deal with datasets that involve noise and, for such datasets, it achieves higher classification accuracy compared to dRHC.

Despite its fairness in treating the newly generated protoypes, dRHC-V2 is not tuned for handling the concept drift phenomenon [25] that may be present in data streams. Newly generated prototypes or prototypes updated during a *condensing set update* stage are in no way favored to survive and remain in the condensing set by removing older ones.

## 2.3. Review of AIB2

The Abstraction IB2 (AIB2) algorithm is a prototype generation version of the well-known IB2 condensing algorithm [2,3]. Contrary to most DRTs, the latter is a one-pass algorithm and constructs its condensing set in an incremental manner[3]. This means that IB2 can add new prototypes to an already constructed condensing set without needing to retain the instances that had been used for the initial construction of the condensing set. From this point of view, IB2 is suitable for data streams and for large datasets that cannot fit into the device's memory. IB2 starts by moving the first training instance to the condensing set. For each following training instance $inst$, IB2 examines the current condensing set and retrieves the nearest prototype $p$ to $inst$. If $inst$ has a different class from $p$, it is moved to the condensing set. Otherwise, it is ignored.

AIB2 inherits all the properties of IB2. In addition, AIB2 not only appends new prototypes to an already constructed condensing set; it may also update existing prototypes. The motive behind the development of AIB2 is that prototypes should be the centroids of the instances they represent. Thus, if the examined instance has the same class label with its nearest prototype in the current condensing set, the examined instance is not ignored as in the case of IB2. Instead, it contributes to the shaping of the condensing set by updating the nearest prototype. To accomplish this, AIB2 assigns a weight value to each prototype. The weight value denotes the number of instances that the prototype represents and it is used to move the prototype in the data space. In effect, the nearest prototype moves towards the examined instance.

Considering the above, AIB2 improves on the representation effectiveness of the condensing set prototypes in comparison with IB2. Consequently, higher classification accuracy is achieved. Moreover, by having prototypes act as centroids for the instances they represent, AIB2 reduces the number of prototypes in the condensing set. In this respect, higher reduction rates and lower computational cost are achieved when compared to IB2. Contrary to dRHC and dRHC-V2, AIB2 does not consider the dataset in the form of data segments. Each instance is processed individually. Like dRHC, AIB2 is also subject to the noise effect: a noisy instance nearest to a prototype is likely to be of a different class. As such, its inclusion in the condensing set increases the number of prototypes in the latter.

[3] According to the reviews [9,24], DRTs can be either incremental or decremental. This depends on how they construct the condensing set. Here, the term incremental refers to the algorithm's ability to update an already constructed condensing set

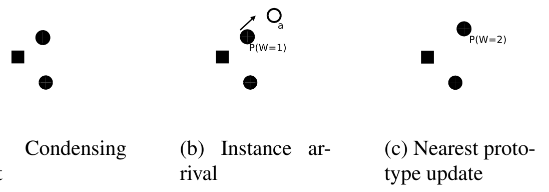


**Fig. 2.** AIB2 example: repositioning an existing prototype. CS prototypes are colored black. The new instance is colored white. Shapes indicate classes.

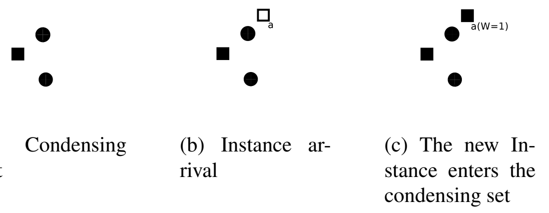


**Fig. 3.** AIB2 example: new prototype enters the condensing set. CS prototypes are colored black. The new instance is colored white. Shapes indicate classes.

AIB2 execution examples are depicted in Figures 2 and 3. Initially, the condensing set includes three prototypes. Suppose that a new circle instance $a$ arrives (Figure 2(b)). Since $a$ is closer to a prototype $P$ of the same class, $P$ moves towards $a$ and its weight is increased by one (Figure 2(c)). On the other hand, suppose that a new square instance, $a$, arrives (Figure 3(b)). Since $a$ is closer to a prototype of a different class, $a$ enters the condensing set forming a new prototype the weight of which is set to one (Figure 3(c)).

## 3. The AIB2-V2 Algorithm

One may claim that dRHC and AIB2 are also appropriate for data streams and large datasets. This is not true since the repetitive generation of new prototypes is likely to lead to a very large condensing set, one that is inappropriate for instance-based classification. This fact renders dRHC and AIB2 inapplicable, especially on infinite data streams [13]. Therefore, there is a need of a mechanism for the size of the condensing set to remain compact. The dRHC-V2 algorithm is seen to handle large datasets and data streams, yet its prototype ranking stage may render its use problematic especially in cases of very fast (i.e., high velocity) data streams. Even a quick sort procedure may not cope with very fast rates of instance arrivals. Therefore, dRHC-V2 may involve a large queue of data segments that await their turn for processing. In case of high speed data streaming, prototypes ranking should be avoided.

Yet another weakness of dRHC-V2 is that it first appends a set of new prototypes to an already existing condensing set and it then removes the prototypes with the lowest

$AnA$ values (a.k.a. the weakest prototypes). Since the number of new prototypes into the condensing set during each *condensing set update* stage is not known beforehand, the reverse cannot be implemented. Thus, the size of the condensing set may temporarily exceed the size of available memory, an issue to be taken into consideration when setting the $T$ parameter for dRHC-V2.

AIB2-V2 is a simple variation of AIB2 that successfully addresses the stated dRHC-V2 drawbacks. Like AIB2, AIB2-V2 does not consider the new data as data segments. Each new instance is considered individually. During each iteration of the algorithm, an existing prototype either gets updated or a new prototype is generated. Like in dRHC-V2, AIB2-V2 is given a $T$ threshold value. When the generation of a new prototype results into a condensing set whose size exceeds $T$, the weakest prototype gets discarded. Since, only one prototype is discarded, there is no need for the algorithm to rank the prototypes. When a prototype need be removed to retain the size of the condensing set within the set $T$ value, AIB2-V2 calculates the $AnA$ value of each prototype and removes the weakest one. One pass over the prototypes set suffices. In case of a tie involving two or more prototypes having the same (lowest) $AnA$ value, the oldest one gets removed.

At each one point in time (clock tick), one instance arrives and it is examined against the existing condensing set. There exist no data segments in the logic of the AIB2-V2 algorithm. A clock tick corresponds to incrementing the time value by 1. The newly generated prototype is assigned the current time value. A prototype created at "time" $r$ is of age $t - r$ when the current time is $t$. Hence, the value of $AnA$ is calculated as follows:

$$AnA = \frac{w}{t - r + 1}$$

with $w$ being the given prototype's weight.

The pseudocode in Algorithm 1 presents the AIB2-V2 algorithm. Like AIB2, when an instance $inst$ from the training set $TS$ has the same class as its nearest prototype $nn$ in the current condensing set $CS$, $nn$'s attributes are updated by taking into consideration its current weight and the attributes of $inst$. Of course, since $inst$ is from now on to be represented by $nn$, the weight of $nn$ is increased by 1 (lines 32–35). The difference between AIB2-V2 and AIB2 is depicted in lines 16–22 and 27–29. The algorithm starts to remove prototypes when $|CS| > T$. The for-loop in lines 12–23 performs the single pass over the prototypes and finds the nearest prototype $nn$. Simultaneously, it computes the corresponding $AnA$ values and marks the prototype $p\_min$ with the lowest $AnA$ value. Next, if $inst$ enters $CS$, $p\_min$ is removed from $CS$ (line 28). $p\_min$ is retrieved only when the size of the condensing set exceeds the set $T$ threshold value. Contrary to the ranking procedure executed by dRHC-V2 for each data segment, here, there is no extra cost for finding the prototype with the lowest $AnA$. It is retrieved by the single pass over the prototypes. It is worth noting that the new prototype enters at the end of $CS$. Thus, the prototypes in $CS$ are stored sorted by their age values, from the oldest to the newest. Therefore, in case of ties, $p\_min$ is the oldest prototype with the lowest $AnA$ value.

As already stated, both dRHC and AIB2 are not noise-tolerant. In the case of their -V2 variations, it is noted that noisy prototypes usually involve low $AnA$ values and they are amongst the first candidates for removal from the condensing set. In this respect, it may safely be assumed that both dRHC-V2 and AIB2-V2 are noise tolerant algorithms, provided that in the course of condensing set construction the pre-specified $T$ value is exceeded for the algorithms to start removing prototypes from the condensing set. This

**Algorithm 1** AIB2-V2

```
Input: $TS, T$ Output: $CS$
1: $time \leftarrow 1$
2: $CS \leftarrow \varnothing$
3: move first instance $inst$ of $TS$ to $CS$
4: $inst_{weight} \leftarrow 1$
5: $inst_r \leftarrow time$
6: while there exist instances in $TS$ do
7:   $time \leftarrow time + 1$
8:   $inst \leftarrow$ next instance in $TS$
9:   $minimum\_AnA \leftarrow \infty$
10:  $p\_min \leftarrow NULL$
11:  $nn \leftarrow NULL$
12:  for each prototype $p \in CS$ do
13:    if $p$ is the nearest prototype of $inst$ then
14:      $nn \leftarrow p$
15:    end if
16:    if $|CS| > T$ then
17:      $p_{AnA} \leftarrow \frac{p_{weight}}{time - p_r + 1}$
18:      if $p_{AnA} < minimum\_AnA$ then
19:        $minimum\_AnA \leftarrow p_{AnA}$
20:        $p\_min \leftarrow p$
21:      end if
22:    end if
23:  end for
24:  if $nn_{class} \neq inst_{class}$ then
25:    $inst_{weight} \leftarrow 1$
26:    $inst_r \leftarrow time$
27:    if $|CS| > T$ then
28:      $CS \leftarrow CS - \{p\_min\}$
29:    end if
30:    $CS \leftarrow CS \cup \{inst\}$
31:  else
32:    for each attribute $attr(i)$ of $nn$ do
33:      $nn_{attr(i)} \leftarrow \frac{nn_{attr(i)} \times nn_{weight} + inst_{attr(i)}}{nn_{weight} + 1}$
34:    end for
35:    $nn_{weight} \leftarrow nn_{weight} + 1$
36:  end if
37:  $TS \leftarrow TS - \{inst\}$
38: end while
39: return $CS$
```

is yet one more issue to be taken into consideration when setting the value of the $T$ parameter. Setting $T$ to a high value effectively disables the noise tolerant character of the algorithm, and setting it too low ends up in having the algorithm remove useful prototypes from the condensing set, especially in cases when the size of the latter is relatively small.

## 4. Performance Evaluation

### 4.1. Experimental Setup

The performance of AIB2-V2 was tested against dRHC-V2, dRHC and AIB2 using fourteen well-known and widely-used datasets distributed by the KEEL repository[4] [4]. Their profile is summarized in Table 1. It is worth mentioning that in [19] and [18], dRHC and AIB2 were evaluated against five state-of-the-art DRTs by utilizing the same fourteen datasets. More specifically, dRHC and AIB2 were experimentally evaluated against the CNN-rule [11], IB2 [3,2], PSC [14,15], ENN-rule [26] and RSP3 [22]. In this respect, when dRHC-V2 and AIB2-V2 are compared to dRHC and AIB2, they are “indirectly” compared to these five data reduction techniques. We did not include the PS and PG algorithms reviewed in [21] because they either focus on concept drift detection or introduce high computational cost. Consequently, they are inappropriate to be compared against AIB2-V2 and dRHC-V2. It is worth mentioning that the reader may execute all dRHC-V2, AIB2-V2, their predecessors and the aforementioned algorithms using the stated datasets at the publicly accessible WebDR[5]. environment [20].

**Table 1.** Dataset description

| Dataset | Size | Attributes | Classes |
|---|---|---|---|
| Letter Image Recognition (LIR) | 20,000 | 16 | 26 |
| Magic G. Telescope (MGT) | 19,020 | 10 | 2 |
| Pen-Digits (PD) | 10,992 | 16 | 10 |
| Landsat Satellite (LS) | 6,435 | 36 | 6 |
| Shuttle (SH) | 58,000 | 9 | 7 |
| Texture (TXR) | 5,500 | 40 | 11 |
| Phoneme (PH) | 5,404 | 5 | 2 |
| Balance (BL) | 625 | 4 | 3 |
| Pima (PM) | 768 | 8 | 2 |
| Ecoli (ECL) | 336 | 7 | 8 |
| Yeast (YS) | 1,484 | 8 | 10 |
| Twonorm (TN) | 7,400 | 20 | 2 |
| MONK 2 (MN2) | 432 | 6 | 2 |
| KddCup (KDD) | 141,481 | 36 | 23 |

[4] `http://sci2s.ugr.es/keel/datasets.php`
[5] `https://atropos.uom.gr/webdr`

The LIR, PD, SH, PH, TXR datasets are all noise-free. It is expected that both dRHC-V2 and AIB2-V2 will perform better on noisy datasets by achieving higher classification accuracy than dRHC and AIB2, respectively. Thus, we built two additional datasets with noise by introducing a 10% of random noise to the PD and LS datasets. The noise was introduced by setting 10% of the training instances to a randomly chosen (different) class label. We refer to these datasets as PDN and LSN, respectively.

The algorithms were coded in C, adopting the Euclidean distance as the distance metric. We randomized the datasets that were distributed sorted on the class label. The KDD dataset involves a large number of duplicates, their ranges varying widely. We removed duplicates and normalized the attributes to the $[0, 1]$ range. Furthermore, we removed the nominal and the fixed-value attributes that exist in KDD. No other transformation was applied to all other datasets. For each one dataset and algorithm, Accuracy (ACC), Reduction Rate (RR), and Preprocessing Cost (PC) in terms of distance computations were calculated. The average values of the three reported measurements were obtained via five-fold-cross-validation. As expected, the higher the reduction rates, the fewer distances are computed during the classification step and, as a consequence, the lower is the computational cost introduced by $k$-NN classifier. Therefore, we did not measure the cost of the classification process.

Since the concept drift phenomenon is not present in the datasets used, we did not implement any special evaluation method for data streams, like test-then-train [8]. Classification accuracy was estimated by running $k$-NN classification with $k = 1$. The PC cost measurements conducted do not include the cost overhead introduced by the prototypes ranking in the case of dRHC2-V2. The cost of ranking is $\mathcal{O}(n \log n)$ on average when a quicksort approach is used. When the threshold is reached, dRHC-V2 ranks the prototypes after the arrival of each data segment. AIB2-V2 avoids the cost of ranking.

Both dRHC and dRHC-V2 consider data in data segments. Datasets are split into data segments of a specific size. Table 2 lists the segment size and the number of segments used for each dataset. It is worth noting that the experiments conducted empirically in [19] reveal that the size of the data segment does not influence the performance of dRHC. Hence, the experimental measurements reported do not involve different segment sizes.

Both AIB2-V2 and dRHC-V2 utilize the $T$ threshold value, that is a ceiling value for the condensing set size. If the condensing set size exceeds $T$, both algorithms remove prototypes from the condensing set in order to maintain its size equal to $T$. A number of runs were conducted, with $T$ assuming a value from a set of percentages of the size of the condensing set constructed by dRHC, namely 85%, 70%, 55%, and 40%. This is done in order to be fair when comparing the two algorithms, since dRHC and and AIB2 achieve similar data reduction rates. Table 2 lists the $T$ values used.

## 4.2. Experimental Results

Two types of experiments were conducted. The first type focuses on measuring the performance, following the arrival of all data. The second type measures the data reduction rate (i.e., the size of the condensing set) and the classification accuracy achieved following the arrival and the processing of each one data segment.

Table 3 lists the performance measurements conducted following the arrival of all data segments. Table 3 lists the accuracy achieved by the 1-NN classifier operating on

**Table 2.** Segment size and $T$ parameter values

| **Dataset** | **Segment size** | **Segments** | **CS Size** $T = 85\%$ | **CS Size** $T = 70\%$ | **CS Size** $T = 55\%$ | **CS Size** $T = 40\%$ |
|---|---|---|---|---|---|---|
| LIR | 2,000 | 8 | 1,608 | 1,324 | 1.040 | 757 |
| MGT | 1.902 | 8 | 3,283 | 2,703 | 2,124 | 1,545 |
| PD | 1,000 | 9 | 207 | 171 | 134 | 97 |
| LS | 572 | 9 | 510 | 420 | 330 | 240 |
| SH | 1,856 | 25 | 197 | 162 | 128 | 93 |
| TXR | 440 | 10 | 189 | 156 | 122 | 89 |
| PH | 500 | 9 | 649 | 534 | 420 | 305 |
| BL | 100 | 5 | 93 | 77 | 60 | 44 |
| PM | 100 | 7 | 182 | 150 | 118 | 86 |
| ECL | 100 | 3 | 71 | 58 | 46 | 33 |
| YS | 396 | 3 | 492 | 405 | 318 | 232 |
| TN | 592 | 10 | 233 | 192 | 151 | 110 |
| MN2 | 115 | 3 | 9 | 8 | 6 | 4 |
| KDD | 4,000 | 29 | 752 | 620 | 487 | 354 |
| PDN | 1,000 | 9 | 2,072 | 1,706 | 1,341 | 975 |
| LSN | 572 | 9 | 1,249 | 1,029 | 808 | 588 |

the condensing set of each algorithm. The best results are highlighted in boldface. Moreover, Table 3 lists the reduction rates achieved by the algorithms and the corresponding preprocessing costs in terms of millions of distance computations. Although, DRTs are adopted when the conventional $k$-NN classifier cannot be applied due to its limitations, for reference, Table 3 presents the accuracy measurements achieved by applying $k$-NN on the original complete training set.

The accuracy measurements for AIB2-V2 are quite promising (see in Table 3). In cases of noise-free datasets, the $k$-NN classifier that operates on condensing sets built by AIB2-V2 achieves accuracy values comparable to those of the dRHC and AIB2 algorithms at a lower (classification stage) computational cost, and at a lower preprocessing cost. In cases of datasets that contain noise, the gain is higher. AIB2-V2 and dRHC-V2 are measured to achieve higher classification accuracy than dRHC and AIB2. This happens because noisy data originating prototypes relate to low importance (i.e. $AnA$) values and as such they are removed from the condensing set. Hence, in cases of datasets with a high level of noise (e.g., MGT, PDN, LSN, BL), AIB2-V2 and dRHC-V2 with the lowest $T$ value were found to achieve even higher accuracy than the conventional $k$-NN classifier. It is noted that the lower is the $T$ value used, the higher the reduction rate and the lower the preprocessing cost achieved. Since AIB2-V2 and dRHC-V2 adopt a ceiling value for the maximum condensing set size and maintain it throughout the whole execution, the reduction rate (RR) and processing cost (PC) results obtained were as expected; the size of the condensing set and the processing cost increase until the set maximum value $T$ is reached and then they remain constant, for all the $T$ values used.

It is not clear which of AIB2-V2 and dRHC-V2 are more accurate. In Table 3, dRHC-V2 is seen to construct its condensing set by computing fewer distances than AIB2-V2.

However, it ranks the prototypes and it may be problematic in cases of very fast data streams. Also, the average measurements (AVG) depicted in the last row of Table 3 suggest that dRHC-V2 and AIB2-V2 can achieve even better accuracy than their predecessors by avoiding the arbitrary growth in size of the condensing set, and by reducing the preprocessing cost.

Tables 3 lists performance rates measured after the arrival of the last data segment, i.e., when all data are processed. An additional set of experiments conducted involved the measuring of accuracy and condensing set sizes following the processing of each one data segment by executing the algorithms on ten datasets. With the exception of the Shuttle (SH) dataset, the largest datasets were used for this set of experimental runs. In the case of the Shuttle (SH) dataset accuracy does not present an essential variance from one data segment to another and for this reason the dataset was excluded. For AIB2-V2, whereby processing does not involve the use of data segments, accuracy values were measured utilizing condensing set snapshots as they were following the processing of $s$ prototypes, where $s$ is the "Segment size" value listed in Table 2.

Figures 4– 13 present the results obtained from this new set of experiments. Each figure involves two diagrams. Diagram (a) plots the size of the condensing set following the arrival of each one data segment (numbered 1,2,...) on the $x$ axis. The size of the condensing set is seen to rise and start to level off when $T$ is reached. Diagram (b) plots the accuracy achieved following the processing of each (subsequent) data segment. It is worth noting that for the MGT, PDN and LSN datasets which involve a relatively high level of noise, the classification accuracy increases considerably when the prototype removing mechanism is enabled. Thus algorithms that adopt low $T$ values perform better in cases of noisy datasets. In general, we observe accuracy tends to rise relatively fast for the first few data segments that are being processed and it then tends to level off either gradually (e.g. for LS, PH) or a bit more abruptly (LIR, PD, TXR and KDD).

## 4.3. Wilcoxon Signed Rank Test results

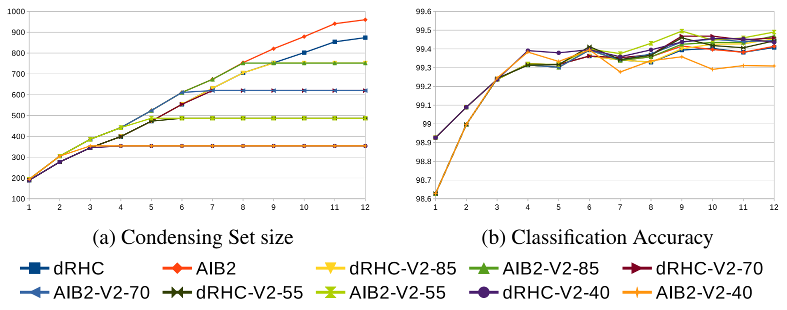
The experimental results obtained are herewith complemented by the Wilcoxon signed rank test results [7,23] in order to statistically confirm the validity of the ACC measurements presented in Table 3. The Wilcoxon signed rank test compares all the algorithms in pairs, considering the accuracy achieved against each one dataset. All four versions of dRHC(-V2) and AIB2(-V2) were considered, using a number of $T$ values. Since both dRHC-V2 and AIB2-V2 dominate in terms of the RR and PC results obtained, there was no need to include the corresponding measures in the test.

Table 4 presents the Wilcoxon signed rank test results obtained. The column labeled "w/l/t" lists the number of wins, losses and ties for each one comparison test. The column labeled "Wilcoxon" value (last column) lists a figure that quantifies the significance of the measured difference between the two algorithms compared. When it is lower than 0.05, the difference is statistically significant.

Despite the fact that AIB2-V2 is seen to involve more wins over dRHC-V2, the results in Table 3 indicate that there exists no statistically significant difference between the two algorithms. Furthermore, there is no statistical difference between AIB2-V2 and AIB2. In the case of dRHC-V2 and dRHC, the former is seen to outperform the latter when $T$=85%, and when $T$=70%. There is no statistical difference between the two when $T$=55%, and when $T$=40%. The results obtained reveal that both dRHC-V2 and AIB2-V2 can be used

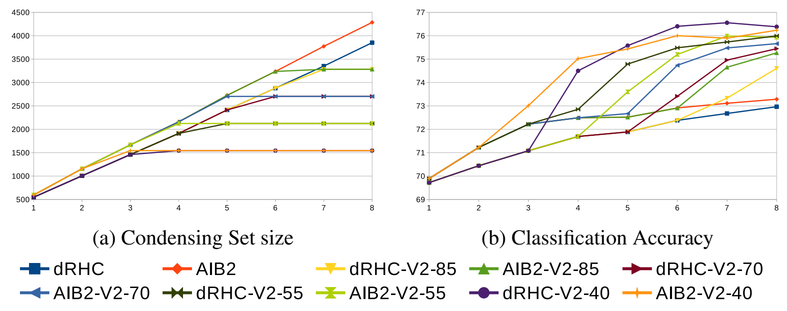
**Table 3.** Comparisons in terms of Accuracy (ACC(%)), Reduction Rates (RR(%)) and Preprocessing Cost(M)

| Data | *T*% | 1NN | dRHC | dRHC-V2 | AIB2 | AIB2-V2 | dRHC | | dRHC-V2 | | AIB2 | | AIB2-V2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | RR | PC | RR | PC | RR | PC | RR | PC |
| LIR | 85: | 95.83 | 93.920 | 93.530 | **94.145** | 93.775 | 88.18 | 19.574 | 89.95 | 19.179 | 88.14 | 20.098 | 89.95 | 19.387 |
| | 70: | | | 92.835 | | 92.750 | | | 91.73 | 17.721 | | | 91.73 | 17.488 |
| | 55: | | | 91.660 | | 91.440 | | | 93.50 | 15.362 | | | 93.50 | 14.820 |
| | 40: | | | 88.845 | | 89.075 | | | 95.27 | 12.293 | | | 95.27 | 11.337 |
| MGT | 85: | 78.14 | 72.965 | 74.606 | 73.286 | 75.268 | 74.62 | 26.025 | 78.42 | 25.851 | 71.87 | 33.079 | 78.42 | 31.259 |
| | 70: | | | 75.447 | | 75.662 | | | 82.24 | 24.414 | | | 82.24 | 28.551 |
| | 55: | | | 75.920 | | 75.994 | | | 86.04 | 21.709 | | | 86.04 | 24.648 |
| | 40: | | | **76.393** | | 76.236 | | | 89.85 | 17.730 | | | 89.85 | 19.513 |
| PD | 85: | 99.35 | 98.490 | **98.508** | 98.326 | 98.226 | 97.23 | 1.438 | 97.65 | 1.410 | 97.19 | 1.381 | 97.65 | 1.336 |
| | 70: | | | 98.235 | | 98.062 | | | 98.06 | 1.317 | | | 98.06 | 1.234 |
| | 55: | | | 97.953 | | 97.316 | | | 98.48 | 1.155 | | | 98.48 | 1.057 |
| | 40: | | | 97.471 | | 96.543 | | | 98.90 | 0.925 | | | 98.90 | 0.811 |
| PDN | 85: | 89.52 | 75.864 | 81.832 | 77.265 | 84.125 | 72.28 | 9.502 | 76.44 | 9.407 | 70.04 | 11.572 | 76.44 | 11.045 |
| | 70: | | | 85.862 | | 86.672 | | | 80.60 | 8.908 | | | 80.60 | 10.128 |
| | 55: | | | 89.065 | | 88.719 | | | 84.75 | 7.964 | | | 84.75 | 8.786 |
| | 40: | | | **91.394** | | 90.175 | | | 88.91 | 6.504 | | | 88.91 | 6.998 |
| LS | 85: | 90.60 | 88.500 | 88.671 | 89.417 | **89.573** | 88.35 | 1.531 | 90.09 | 1.511 | 86.77 | 1.916 | 90.09 | 1.785 |
| | 70: | | | 88.920 | | 89.402 | | | 91.84 | 1.422 | | | 91.84 | 1.615 |
| | 55: | | | 88.687 | | 88.873 | | | 93.59 | 1.261 | | | 93.5 | 1.378 |
| | 40: | | | 88.314 | | 88.733 | | | 95.34 | 1.029 | | | 95.34 | 1.075 |
| LSN | 85: | 81.99 | 76.566 | 79.487 | 77.653 | 81.663 | 71.44 | 3.511 | 75.74 | 3.477 | 68.28 | 4.349 | 75.74 | 4.101 |
| | 70: | | | 81.974 | | 82.828 | | | 80.01 | 3.288 | | | 80.01 | 3.737 |
| | 55: | | | 83.357 | | 83.807 | | | 84.31 | 2.932 | | | 84.31 | 3.213 |
| | 40: | | | 84.646 | | **84.880** | | | 88.58 | 2.390 | | | 88.58 | 2.532 |
| SH | 85: | 99.82 | 99.695 | 99.667 | **99.726** | 99.671 | 99.50 | 7.977 | 99.58 | 7.614 | 99.45 | 8.041 | 99.58 | 7.420 |
| | 70: | | | 99.624 | | 99.621 | | | 99.65 | 6.911 | | | 99.65 | 6.540 |
| | 55: | | | 99.590 | | 99.593 | | | 99.72 | 5.946 | | | 99.72 | 5.453 |
| | 40: | | | 99.400 | | 99.243 | | | 99.80 | 4.729 | | | 99.80 | 4.127 |
| TXR | 85: | 99.02 | 97.600 | 97.291 | **97.691** | 97.400 | 94.95 | 0.685 | 95.71 | 0.669 | 94.91 | 0.660 | 95.71 | 0.637 |
| | 70: | | | 98.836 | | 96.909 | | | 96.46 | 0.617 | | | 96.46 | 0.575 |
| | 55: | | | 96.218 | | 96.255 | | | 97.23 | 0.533 | | | 97.23 | 0.481 |
| | 40: | | | 95.182 | | 94.982 | | | 97.98 | 0.429 | | | 97.98 | 0.369 |
| PH | 85: | 90.10 | 85.381 | **85.667** | 85.067 | 85.474 | 82.34 | 1.638 | 84.99 | 1.615 | 81.50 | 1.883 | 84.99 | 1.808 |
| | 70: | | | 85.104 | | 85.363 | | | 87.65 | 1.515 | | | 87.65 | 1.651 |
| | 55: | | | 84.955 | | 84.327 | | | 90.29 | 1.333 | | | 90.29 | 1.420 |
| | 40: | | | 83.845 | | 83.623 | | | 92.95 | 1.077 | | | 92.95 | 1.119 |
| BL | 85: | 78.40 | 70.560 | 74.080 | 68.480 | 77.600 | 78.12 | 0.029 | 81.40 | 0.029 | 70.56 | 0.037 | 81.40 | 0.032 |
| | 70: | | | 77.600 | | 78.080 | | | 84.60 | 0.027 | | | 84.60 | 0.028 |
| | 55: | | | 79.840 | | 79.520 | | | 88.00 | 0.025 | | | 88.00 | 0.024 |
| | 40: | | | **81.790** | | 77.760 | | | 91.20 | 0.021 | | | 91.20 | 0.018 |
| PM | 85: | 68.36 | 63.925 | 66.792 | 67.321 | 66.796 | 65.11 | 0.064 | 70.41 | 0.063 | 64.75 | 0.062 | 70.41 | 0.060 |
| | 70: | | | 67.572 | | 64.972 | | | 75.61 | 0.060 | | | 75.61 | 0.056 |
| | 55: | | | **69.913** | | 67.445 | | | 80.81 | 0.055 | | | 80.81 | 0.049 |
| | 40: | | | 67.184 | | 68.481 | | | 86.02 | 0.046 | | | 86.02 | 0.040 |
| ECL | 85: | 79.78 | 71.462 | 74.732 | 72.963 | 72.963 | 68.92 | 0.015 | 73.61 | 0.015 | 68.70 | 0.011 | 73.61 | 0.011 |
| | 70: | | | 76.216 | | 74.460 | | | 78.44 | 0.015 | | | 78.44 | 0.011 |
| | 55: | | | 75.316 | | 71.185 | | | 82.90 | 0.014 | | | 82.90 | 0.009 |
| | 40: | | | **77.094** | | 70.597 | | | 87.73 | 0.013 | | | 87.73 | 0.007 |
| YS | 85: | 52.02 | 48.379 | 49.256 | 48.247 | 49.258 | 51.23 | 0.306 | 58.59 | 0.306 | 47.10 | 0.366 | 58.59 | 0.349 |
| | 70: | | | 49.864 | | 49.932 | | | 65.91 | 0.306 | | | 65.91 | 0.321 |
| | 55: | | | 50.743 | | 49.594 | | | 73.23 | 0.278 | | | 73.23 | 0.278 |
| | 40: | | | **51.415** | | 49.662 | | | 80.47 | 0.244 | | | 80.47 | 0.222 |
| TN | 85: | 94.88 | 93.081 | 93.838 | 93.054 | 94.865 | 95.37 | 0.695 | 96.06 | 0.688 | 93.47 | 1.080 | 96.06 | 0.917 |
| | 70: | | | 94.514 | | 95.203 | | | 96.76 | 0.654 | | | 96.76 | 0.822 |
| | 55: | | | 94.986 | | 95.351 | | | 97.45 | 0.590 | | | 97.45 | 0.701 |
| | 40: | | | 95.459 | | **95.568** | | | 98.14 | 0.495 | | | 98.14 | 0.551 |
| MN2 | 85: | 90.51 | **97.680** | 96.517 | 93.293 | 91.890 | 96.88 | 0.004 | 97.63 | 0.004 | 93.18 | 0.005 | 97.63 | 0.003 |
| | 70: | | | 95.589 | | 86.792 | | | 97.80 | 0.004 | | | 97.80 | 0.003 |
| | 55: | | | 91.190 | | 81.489 | | | 98.27 | 0.004 | | | 98.27 | 0.002 |
| | 40: | | | 80.281 | | 81.251 | | | 98.84 | 0.004 | | | 98.84 | 0.001 |
| KDD | 85: | 99.71 | 99.424 | 99.449 | 99.414 | **99.469** | 99.22 | 54.70 | 99.34 | 53.555 | 99.21 | 58.705 | 99.34 | 56.947 |
| | 70: | | | 99.464 | | 99.444 | | | 99.45 | 49.811 | | | 99.45 | 52.493 |
| | 55: | | | 99.443 | | 99.443 | | | 99.57 | 43.476 | | | 99.57 | 45.319 |
| | 40: | | | 99.353 | | 99.309 | | | 99.69 | 34.603 | | | 99.69 | 35.609 |
| AVG | 85: | 86.75 | 83.343 | 84.620 | 83.459 | 84.876 | 82.73 | 7.981 | 85.35 | 7.837 | 68.28 | 8.952 | 85.35 | 8.569 |
| | 70: | | | 85.479 | | 84.760 | | | 87.92 | 7.312 | | | 87.92 | 7.828 |
| | 55: | | | **85.552** | | 84.397 | | | 90.51 | 6.415 | | | 90.51 | 6.727 |
| | 40: | | | 84.879 | | 84.132 | | | 93.10 | 5.158 | | | 93.10 | 5.271 |



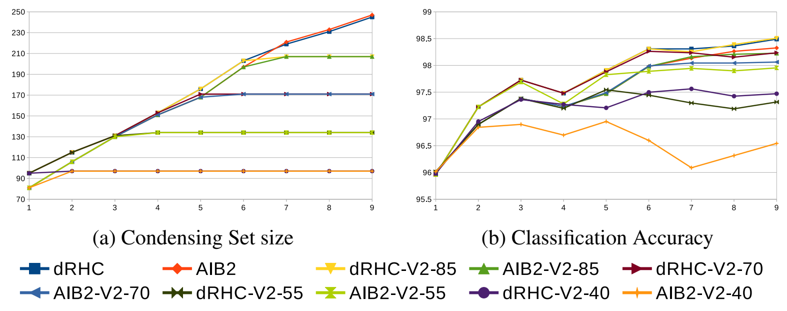
(a) Condensing Set size (b) Classification Accuracy

**Fig. 4.** LIR: Condensing Set Size and Classification Accuracy per data segment



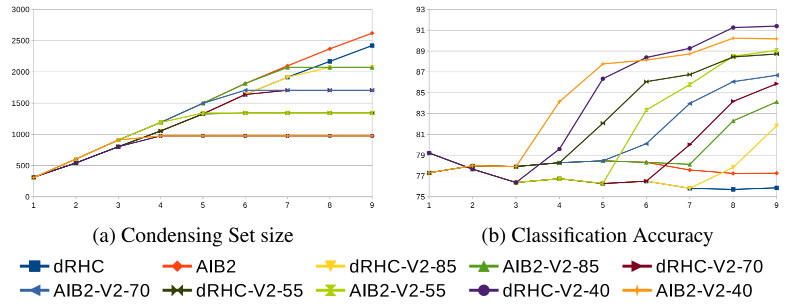
(a) Condensing Set size (b) Classification Accuracy

**Fig. 5.** MGT: Condensing Set Size and Classification Accuracy per data segment



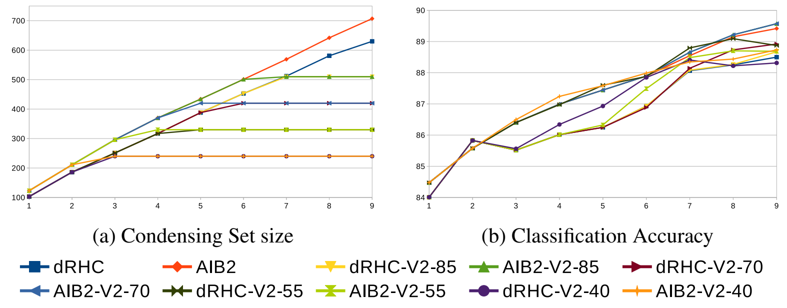
(a) Condensing Set size (b) Classification Accuracy

**Fig. 6.** PD: Condensing Set Size and Classification Accuracy per data segment



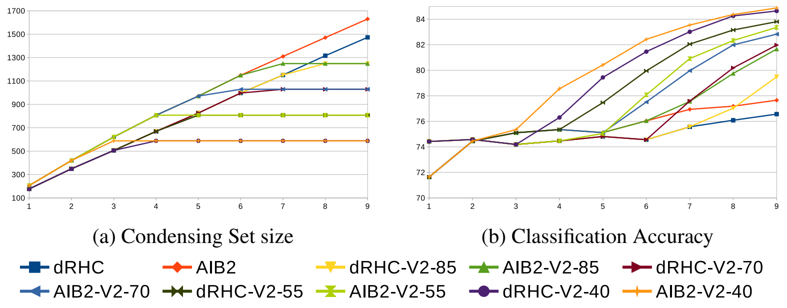
(a) Condensing Set size (b) Classification Accuracy

**Fig. 7.** PDN: Condensing Set Size and Classification Accuracy per data segment



(a) Condensing Set size (b) Classification Accuracy

**Fig. 8.** LS: Condensing Set Size and Classification Accuracy per data segment



(a) Condensing Set size (b) Classification Accuracy

**Fig. 9.** LSN: Condensing Set Size and Classification Accuracy per data segment

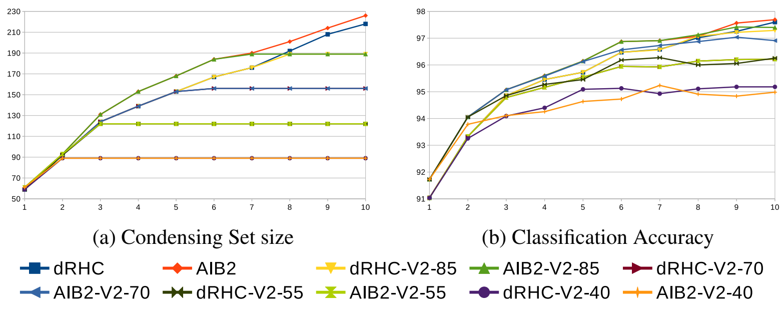


**Fig. 10.** TXR: Condensing Set Size and Classification Accuracy per data segment

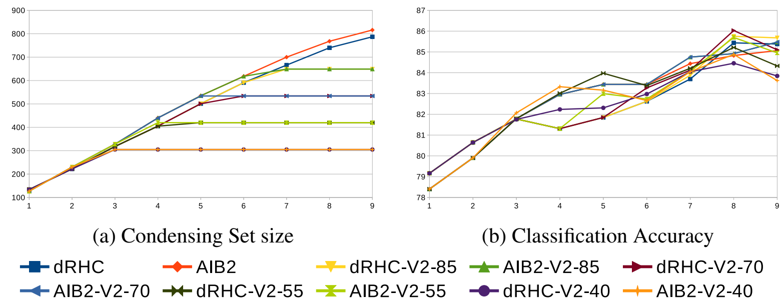


**Fig. 11.** PH: Condensing Set Size and Classification Accuracy per data segment

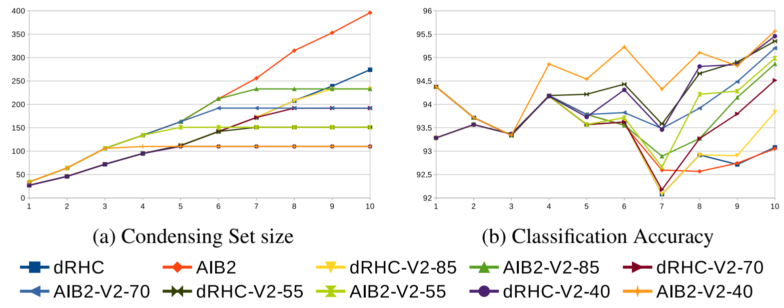
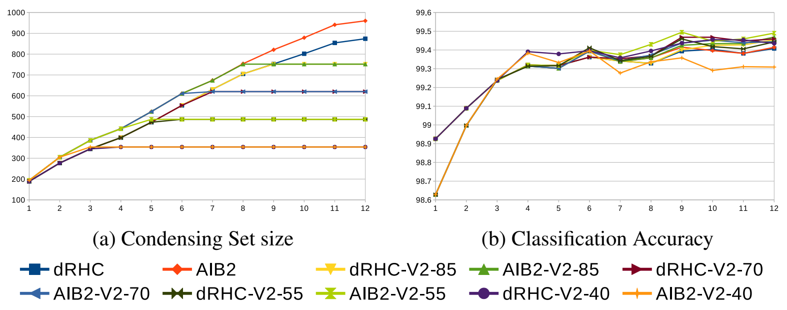


**Fig. 12.** TN: Condensing Set Size and Classification Accuracy per data segment



(a) Condensing Set size (b) Classification Accuracy

**Fig. 13.** KDD: Condensing Set Size and Classification Accuracy per data segment

instead of dRHC and AIB2 without loss of accuracy when there is need for a condensing set with a fixed size.

**Table 4.** Results of Wilcoxon signed rank test

| **Methods** | **Accuracy** | |
|---|---|---|
| | **w/l/t** | **Wilcoxon** |
| dRHC vs dRHC-V2 ($T$=85%) | 4/12/0 | 0.030 |
| dRHC vs dRHC-V2 ($T$=70%) | 5/11/0 | 0.026 |
| dRHC vs dRHC-V2 ($T$=55%) | 6/10/0 | 0.121 |
| dRHC vs dRHC-V2 ($T$=40%) | 8/8/0 | 0.326 |
| AIB2 vs AIB2-V2 ($T$=85%) | 6/9/1 | 0.132 |
| AIB2 vs AIB2-V2 ($T$=70%) | 7/9/0 | 0.255 |
| AIB2 vs AIB2-V2 ($T$=55%) | 8/8/0 | 0.756 |
| AIB2 vs AIB2-V2 ($T$=40%) | 7/9/0 | 0.836 |
| dRHC-V2 ($T$=85%) vs AIB2-V2 ($T$=85%) | 4/12/0 | 0.179 |
| dRHC-V2 ($T$=70%) vs AIB2-V2 ($T$=70%) | 8/8/0 | 0.918 |
| dRHC-V2 ($T$=55%) vs AIB2-V2 ($T$=55%) | 6/9/0 | 0.061 |
| dRHC-V2 ($T$=40%) vs AIB2-V2 ($T$=40%) | 6/10/0 | 0.352 |
| dRHC vs AIB2 | 7/9 | 0.408 |

## 5. Conclusion and Directions for Further Work

A new noise-tolerant prototype generation algorithm is considered in detail. It maintains a fixed size condensing set by monitoring a stream of training data, or by managing a large dataset that cannot fit in memory. The new algorithm is code-named AIB2-V2 and has some common characteristics with dRHC-V2. Both comprise improved variations of the AIB2 and dRHC algorithms, respectively. Contrary to dRHC-V2, AIB2-V2 avoids ranking when replacing its prototypes. When a new prototype enters the condensing set and the threshold limit has been reached, the pre-marked prototype with the lowest $AnA$ is removed. The experimental study yields promising results. Even when the condensing set generated by the new algorithm is less than half the size of that generated by the AIB2 algorithm, there is no loss in accuracy and in many cases the accuracy achieved by AIB2-V2 is even higher. By having the size of the condensing set to vary up to a pre-specified maximum value $T$ and practically remain constant, the preprocessing costs involved remain low and constant throughout the execution of each one of the two algorithms. Moreover, the latter can be implemented to execute on devices with limited memory.

Suggested directions for future work include the development of new variations of noise-tolerant prototype generation algorithms that will fully exploit the potential of handling data streams involving the concept drift. This could be achieved by further increasing the value of the importance measure for newly generated prototypes, next to that of old prototypes involving static attribute values in the course of time. Moreover, we also plan to introduce parallelism to the DRTs, in order to speed up the construction of the condensing set.

## References

1. Aggarwal, C.: Data Streams: Models and Algorithms. Advances in Database Systems Series, Springer Science+Business Media, LLC (2007)
2. Aha, D.W.: Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. Int. J. Man-Mach. Stud. 36(2), 267–287 (Feb 1992), http://dx.doi.org/10.1016/0020-7373(92)90018-G
3. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (Jan 1991), http://dx.doi.org/10.1023/A:1022689900470
4. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing 17(2-3), 255–287 (2011)
5. Beringer, J., Hüllermeier, E.: Efficient instance-based learning on data streams. Intell. Data Anal. 11(6), 627–650 (Dec 2007), http://dl.acm.org/citation.cfm?id=1368018.1368022
6. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. 13(1), 21–27 (Sep 2006), http://dx.doi.org/10.1109/TIT.1967.1053964
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (Dec 2006), http://dl.acm.org/citation.cfm?id=1248547.1248548
8. Gama, J.a., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 329–338. KDD '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1557019.1557060
9. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. 34(3), 417–435 (Mar 2012), http://dx.doi.org/10.1109/TPAMI.2011.142
10. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems, Elsevier Science (2011)
11. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14(3), 515–516 (1968)
12. James, M.: Classification algorithms. Wiley-Interscience, New York, NY, USA (1985)
13. Lallouet, A., Law, Y.C., Lee, J.H., Siu, C.F.: Constraint Programming on Infinite Data Streams. In: Walsh, T. (ed.) International Joint Conference on Artificial Intelligence.

pp. 597–604. Barcelone, Spain (Jul 2011), https://hal.archives-ouvertes.fr/hal-01009693
14. Olvera-Lopez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: A new fast prototype selection method based on clustering. Pattern Anal. Appl. 13(2), 131–141 (2010)
15. Olvera-Lpez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: Object selection based on clustering and border objects. In: Kurzynski, M., Puchala, E., Wozniak, M., Zolnierek, A. (eds.) Computer Recognition Systems 2, Advances in Soft Computing, vol. 45, pp. 27–34. Springer (2008)
16. Ougiaroglou, S., Arampatzis, G., Dervos, D.A., Evangelidis, G.: Generating fixed-size training sets for large and streaming datasets. In: Kirikova, M., Nørvåg, K., Papadopoulos, G.A. (eds.) Advances in Databases and Information Systems. pp. 88–102. Springer International Publishing, Cham (2017)
17. Ougiaroglou, S., Evangelidis, G.: AIB2: An abstraction data reduction technique based on ib2. In: Proceedings of the 6th Balkan Conference in Informatics. pp. 13–16. BCI '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2490257.2490260
18. Ougiaroglou, S., Evangelidis, G.: Efficient data abstraction using weighted IB2 prototypes. Comput. Sci. Inf. Syst. 11(2), 665–678 (2014), http://dx.doi.org/10.2298/CSIS140212036O
19. Ougiaroglou, S., Evangelidis, G.: RHC: a non-parametric cluster-based data reduction for efficient k-NN classification. Pattern Analysis and Applications 19(1), 93–109 (2014), http://dx.doi.org/10.1007/s10044-014-0393-7
20. Ougiaroglou, S., Evangelidis, G.: WebDR: A web workbench for data reduction. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 8726, pp. 464–467. Springer Berlin Heidelberg (2014), http://dx.doi.org/10.1007/978-3-662-44845-8_36
21. Ramrez-Gallego, S., Krawczyk, B., Garca, S., Woniak, M., Herrera, F.: A survey on data preprocessing for data stream mining. Neurocomput. 239(C), 39–57 (May 2017), https://doi.org/10.1016/j.neucom.2017.01.078
22. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition 37(7), 1561–1564 (2004)
23. Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures. A Chapman & Hall book, Chapman & Hall/CRC (2011)
24. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. Trans. Sys. Man Cyber Part C 42(1), 86–100 (Jan 2012), http://dx.doi.org/10.1109/TSMCC.2010.2103939
25. Tsymbal, A.: The problem of concept drift: definitions and related work. Tech. Rep. TCD-CS-2004-15, The University of Dublin, Trinity College, Department of Computer Science, Dublin, Ireland (2004)
26. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Mach. Learn. 38(3), 257–286 (Mar 2000), http://dx.doi.org/10.1023/A:1007626913721

**Stefanos Ougiaroglou** holds a B.Sc in Computer Science (2004) from Alexander TEI of Thessaloniki, Greece, a M.Sc. in Computer Science (2006) from Aristotle University of Thessaloniki, Greece and a PhD in Computer Science (2014) from University of Macedonia, Thessaloniki, Greece. His PhD studies were supported by a Scholarship from State Scholarships Foundation (I.K.Y) of Greece. From 2016, he is postdoctoral researcher at the department of Applied Informatics of University of Macedonia. Moreover, S. Ougiaroglou holds a Certificate of Pedagogical and Teaching Competence (2010), ASPAITE, Greece. Currently, he is a laboratory lecturer at the department of Information and Electronic Engineering of the International Hellenic University (IHU), Thessaloniki, Greece where he teaches Databases, Data Structures, Data Mining, Algorithms and Programming, Operating Systems, Web Languages and Technologies and Development of Internet Applications. His research interests include Data Mining algorthms, Data streams, Data management for Mobile Computing and Educational Technology. He has authored several journal and conference papers in these fields. Personal webpage: `https://www.iee.ihu.gr/~stoug/`

**Dimitris A. Dervos** (`https://d-a-d.weebly.com`) is with the Department of Information and Electronic Engineering of the International Hellenic University in Thessaloniki, Greece. He holds a BSc degree in Physics and a Ph.D degree in Computer Science from the Aristotle University of Thessaloniki, plus an M.A. degree in Physics, and an M.Sc. degree in Computer Engineering from the University of Southern California. His teaching record includes undergraduate and graduate level topics ranging from Algorithms and Data Structures to Database Technology and Data Mining taught at academic institutions in Greece, and in the U.K. His research interests primarily focus in the areas of Bibliography Data Analysis, and Data Mining. He has led the research team who received the 2005 Thomson ISI / ASIS&T Citation Analysis Research Grant.

**Georgios Evangelidis** is a Professor at the Department of Applied Informatics, School of Information Sciences, University of Macedonia, Thessaloniki, Greece. He is the Director of the Software and Data Engineering Laboratory of the Department since 2016. He holds a BSc in Mathematics (1987, Aristotle University, Thessaloniki), and MSc and PhD in Computer Science (1990 and 1994, Northeastern University, Boston, MA). His research interests and published work (about 120 papers and book chapters) are in the area of Information Management: Databases, Data Mining, Information Retrieval, Analysis of Bibliographic Databases. He teaches Databases, Data Mining and Geographic Information Systems. He has been coordinator and/or participant in more than 20 Greek government and EU funded research and development projects. Personal webpage: `http://users.uom.gr/~gevan/`

*Received: May 18, 2019; Accepted: November 18, 2019.*

Computer Science and Information Systems 17(1):93–115 https://doi.org/10.2298/CSIS180601015G

# Climate Change Opinions in Online Debate Sites

Adrian Groza[1], Pinar Ozturk[2], Radu Razvan Slavescu[1], and Anca Marginean[1]

[1] Technical University of Cluj-Napoca
400128 Cluj-Napoca, Romania
{Adrian.Groza,Radu.Razvan.Slavescu,Anca.Marginean}@cs.utcluj.ro
[2] Norwegian University of Science and Technology
Trondheim, Norway
pinar@ntnu.no

**Abstract.** Debate sites in social media provide a unified platform for citizens to discuss controversial questions and to put forward their ideas and arguments on the issues of common interest. Opinions of citizens may provide useful knowledge to stakeholders but manual analysis of arguments in debate sites is tedious, while computational support to this end has been rather scarce. We focus here on developing a technical instrumentation for making sense of a set of online arguments and aggregating them into usable results for policy making and climate science communication. Our objectives are: (i) to aggregate arguments posted for a certain debate topic, (ii) to consolidate opinions posted under several but related topics either in the same or different debate site, and (iii) to identify possible linguistic characteristics of the argumentative texts. For the first objective, we propose a voting method based on subjective logic [13]. For the second objective, we assess the semantic similarity between two debate topics based on textual entailment [28]. For the third objective, we employ various existing methods for lexical analysis such as frequency analysis or readability indexes. Although we focused here on the climate change, the method can be applied to any domain.

**Keywords:** online debate analysis, aggregation of individual opinions, web text analysis, decision support for policy making.

## 1. Introduction

Policy makers, managers and social scientists are interested in opinions of stakeholders on issues of environmental, societal and political consequences. Although social media has proven to be a precious data source for studying how people use public arena for communicating their ideas and opinions [7,34], debate sites have not been in research focus to the same extent as other online platforms. The objective of this work is to investigate what kind of information can be extracted from individual opinions posted on debate sites. We focus here on the climate change problem because it is a matter that interests many people who may have different opinions and arguments. However, the method is general and can be used in other areas as well.

Debate sites are structured according to topics, (e.g. ”global warming”). Anybody may post a question (e.g., ”Is global warming affecting the planet?”) or a hypothesis (e.g., “Global warming is affecting the planet”), and anybody can post his or her opinion related to this question/hypothesis. In the rest of this paper we use ”hypothesis” regardless of the initiating post being in affirmative or interrogative format. The responses are votes (e.g., yes/no, pro/against or agree/disagree), which are optionally accompanied by an argumentative text. From the debate analysis perspective, the debate sites therefore possess a distinguished advantage: people's opinions about a debate topic are intrinsically labeled as pro or against, which enables automated extraction of labeled arguments.

It is not unusual that the same or similar hypotheses are discussed in more than one thread in the same debate site, even synchronously, because the debate sites do not offer a service for detecting such redundancy. For example, we noticed that *"Climate change affects the earth"* and *" Global warming affects our planet"* were debated at almost the same time within the same debate community. Moreover, the same hypothesis can be posted on the distinct debate community, where it may attract more (or less) negative (or positive) arguments. Thus, there is need for computational methods to handle these situations in order to have a clearer picture on what is debated related to a topic of interest. Hence, we propose here a computational method and a tool to facilitate a high level view on what is debated online.

There are several challenges in making sense of online debates. First, redundancy occurs when a person posts an existing hypothesis again, with a different wording. Second, the number of responses vary significantly across topics/hypotheses which makes it difficult to compare the degree of support for two hypotheses, one with tiny and the other with massive discussions. Third, different hypotheses may be considered the same for a specific purpose, for example, of a policymaker and hence the responses to them may need to be merged. Fourth, there are several debate sites, which we call "communities", independent from each other but discussing similar or the same topics. Gathering a consolidated opinion across these communities will provide a better insight into public opinions. However, it is not trivial to assess the semantic similarity between hypotheses and hence to extract collective opinions of people from distinct debate sites.

We used debate sites to extract an annotated corpus of climate change arguments in natural language. The motivation is that existing corpora for climate change are based either on media [4], or tweets [14,23]. Both sources do introduce specific disadvantages for natural language processing. First, arguments conveyed in media are too large and sparse within an article or news. Second, arguments in tweets do not follow a specific set of grammar rules. We consider that arguments from debate sites are more adequate for natural language processing (NLP), as arguments are smaller than media documents and they are grammatically more correct than tweets. Moreover, the existing corpora contain arguments labeled as pro or against either manually by external human annotators or automatically (e.g. based on machine learning). Differently, the arguments in our corpus are labeled by the conveyor of the argument himself/herself. That is, the confidence in the labels is higher. Hence, such a corpus can be useful for researchers in natural language arguments or argument mining.

Our objectives are (1) to aggregate arguments posted for a certain hypothesis, (2) to consolidate opinions posted under several but related hypotheses either in the same or different debate site, and (3) to identify possible linguistic characteristics of the argumentative texts. Note that we reserve the term *aggregation* for a summary of opinions under a specific hypothesis posted in one thread, while *consolidation* is used whenever two separated threads about a topic can semantically be merged.

For the first objective, we proposed a vote-based method based on Subjective Logic [13]. For the second objective, we assess the semantic similarity between two hypotheses based on textual entailment [28]. For the third objective, we employ various existing lexical analysis instrumentations such as frequency analysis or readability indexes. Although we focused here on the climate change, the method can be applied to any domain.

A social scientist using our ARGSENSE tool can obtain answers related to the following research questions:

$Q_1$**:** Are the arguers within a community apriori prone to accept or to reject a hypothesis?
$Q_2$**:** Which hypotheses are most (dis)believed or (un)popular in a community?
$Q_3$**:** Do the pro arguments have a different lexicon than the counter ones?
$Q_4$**:** Does an interrogation have more pros or more cons arguments than an affirmation?
$Q_5$**:** Are the pro arguments more readable than the con arguments?
$Q_6$**:** Is the length of hypothesis correlated with the number of arguments it receives?
$Q_7$**:** Does the formulation of the hypothesis itself (e.g., interrogative or affirmative) influence the degree of interest in the debate?

In the rest of the paper, section 2 browses related work on analysing climate change arguments. Section 3 introduces the climate change argument corpus that we harvested from debate sites, and the architecture of ARGSENSE for supporting the analysis of the online debates. Section 4 presents our vote-based method for argument aggregation based on subjective logic. Section 5 presents a method based on textual entailment for consolidating opinions of related debate topics. Section 6 applies opinion aggregation and opinion consolidation in the climate change domain. Section 7 applies lexical analysis for supporting social scientists on the climate change corpus. Finally, we review the findings through a concluding section.

## 2. Related work

Related work is restricted to our running scenario: climate change. We approach related work from three perspectives. First, we introduce different approaches for analysing online arguments on global warming. Second, we browse the existing corpora on climate change. Third, we present related tools that support understanding of climate change.

### 2.1. Opinion aggregation

We are aware of the limitations and risks of aggregating data from online sources. For example, people from online communities disagree far more on climate change than climate experts do. The scientific community has reached a consensus that the rise of average temperature is mostly caused by human activity. It has been argued by Boussalis et al. [4] that the lack of awareness or understanding of the scientific evidence is due to a "coordinated and well-funded counter-movement of climate skeptics". Relevant to the topic of climate change denial is the investigation in [32]. Here, Washington et al. have analysed the argumentative patterns in climate change literature and have identified five types of climate change denial argument: i) conspiracy theory, ii) fake expert, iii) impossible expectations, iv) misrepresentation or logical fallacy v) cherry-picking. From the argumentation technologies viewpoint, these results [32] open the way towards argumentation schemes [31] for climate change.

In the same line of aggregating expert opinions instead of non-expert arguments is the work of HaDoung et al. that compared several procedures to aggregate expert opinion based on an Transferable Belief Model. The approach in [10] has been tested with 16 expert real-world datasets on climate sensitivity. The experts are firstly clustered into fields of taught. In each group, beliefs are aggregated using a cautious conjunction operator. Across groups, a non-interactive disjunction is used. In our case, the arguers are partitioned into different debate sites. Within a community, the argument properties (belief, disbelief, ignorance) are consolidated based on the similarity, contradictory and entailment relations.

Our method for representing individual votes is inspired from the subjective logic of [32]. Lioma et al. have used subjective logic for interactive information retrieval [18]. Subjective logic has been used to model representation of information needs as uncertain beliefs. In [18], the same information need can have various textual representation. Similarly, our debate topics or hypotheses have different supporting arguments.

For consolidation of opinions posted under different (but semantically related) hypothesis, we used natural language processing techniques such as textual entailment. Textual entailment is used here to compute similarity, contradiction and entailment between hypotheses. The latter one includes a supervised machine learning component, that benefits from our crawled labeled arguments. Moreover, in line with [2] our method for detecting similarity, contradiction and entailment relies also on external knowledge resources like Wordnet [22] and VerbOcean [8].

### 2.2. Climate change debate corpora

Existing corpora for climate science are based on media documents [4], or tweets [14,23]. Boussalis et al. [4] has collected 16.000 documents for compiling a corpus of contrarian literature on climate change. 19 organisations known to argue for climate skepticism were included. The corpus was used to analyse the skeptical discourse on global warming over the period 1998-2013. Kirilenko et al. [14] have collected 1.8 million tweets on climate change between 2012 and 2013 in five languages with 41% of the daily discussion of climate change on Twitter originates from the USA and 13% from the UK. Tweets have also been collected for the stance detection dataset [23]. The corpus contains 4870 tweets in five domains: ”atheism”, ”climate change is a real concern”, ”feminist movement”, ”Hillary Clinton” and ”legalisation of abortion”. Each tweet was annotated by at least eight respondents in the CrowdFlower (http://www.crowdflower.com) platform.

From the natural language perspective, our climate change corpus has a technical advantage over the tweets-based corpora or document-based corpora. The advantage comes from the size and structure of the arguments. Tweets are charactered by flexible grammar structure, and lots of links or hash-tags. Hence, natural language processing is based more or less on lexical analysis within a statistical framework. Differently, media documents or journal articles are large and hence it is difficult to automatically filter the relevant information. Our corpus contains small arguments supporting and attacking a debate topic. The size of each pair of arguments makes it possible to effectively apply technical instrumentations such as textual entailment.

Regarding the size of each text, the corpus of Kwon et al. [16] is similar. Kwon at al. have collected 119 public comments about Environmental Protection Agency’s proposed emission standard rule on hazardous pollutants. The comments have been classified in three classes: support, oppose or propose a new idea. Each comment has been annotated by at least two coders. The inter-annotator agreement based on Cohen's Kappa coefficient [15] has been only 0.62. This low value signals the uncertainties rising when the comments are manually labeled by human annotators. Our larger corpus (11.653 arguments compared to 119 comments) does not have this labeling uncertainty, as each label was available from the debate sites. Hence, an advantage of our corpus is that the classification of an argument as positive or negative is given by the conveyor of the argument and not by an external annotator.

Chalaguine and Schulz [6] have focused on how convincing arguments are in online debates. The corpus contains arguments collected from 32 debates on 16 topics from *createdebate.com* and *procon.org*. The collected arguments have been used to generate 16k pairs of arguments. Each pair has been classified by human annotators as more convincing versus less convincing. Note that the arguments are not restricted to only one domain (i.e., climate change). The more topics in the corpus, the more difficult for machine learning to learn the language model. Note also that assessing the strength of an arguments in 16K pairs is a highly subjective task, given the bias due to personal beliefs, preferences and background of the annotators. Moreover, the randomly generated pairs contain arguments from different topics, hence more difficult to assess their strength. Chalaguine and Schultz have used the corpus by extracting 70 features for each pair (POS, statistics on texts, etc.) and fed these features to a neural network.

Hence, our climate change corpus provides the following advantages compared to existing corpora used in the argumentation mining community. First, the size of each argument is relatively small (compared to large scientific documents) and has a proper grammar (compared to tweets). Second, the corpus is restricted to a single domain (i.e. climate change), hence it facilitates building a better language model, in case machine learning is used. Third, classification of arguments into pros and cons is guaranteed to reflect the conveyor's real intention, and does not include inherent errors of manual annotation [16,23].

## 2.3. Tools for climate change understanding

The ARGSENSE tool aims to enhance understanding on climate change topic as it appears in public arena. Henceforth, this section browses related tools used to support stakeholders to understand climate science.

Launched in November 2014, the Web-based tool VisAdap [1] aims to increase understanding of anticipated risks from climate change. These climate change risks are addressed from the perspective of the target group of home-owners. The challenge is that home-owners have socio-cognitive barriers to adapt to the new risks caused by climate change. Risk impact of climate change is anticipated for a given region over the coming 40-60 years. With 16,000 users within 8 months [1], VisAdap has proved a popular tool to support individual decision making when buying or building a house. Currently, VisAdap is designed based on norms and values of home-owners collected from direct interviews. Instead of interviews, ARGSENSE relies on arguments conveyed in the public arena. In the same line of understanding people opinions on a narrow topic, ARGSENSE can perform topic-based analysis. The topic can be selected based on the target stakeholders, including the home-owners. For home-owners, ARGSENSE can report on the arguments related to debate topics such as flooding or storm damage. ARGSENSE is able to signal topics which are not accepted by a community. This result guidelines a policymaker regarding which information to communicate on platforms like VisAdap.

Launched in 2016, the AGCLIMATE hub (http://AgClimate4U.org) aims to transform heterogeneous climate change datasets into usable information in agriculture. Here, the stakeholders are crop farmers and agricultural advisors [3]. To understand their needs, methods from social sciences have been used: surveys, focus groups, interviews, and network analysis. Findings from these methods have been used to design various decision support tools for graphical visualisation of climate data, crop fields, or irrigation investment. ARGSENSE follows the same ”useful to usable” paradigm: the argumentation dataset is analysed from different perspectives to present usable findings to a policymaker. Data in AGCLIMATE is obtained by querying the Web services of various providers (i.e. National Oceanic Atmospheric Administration, Midwestern Regional Climate Center, USDA National Agricultural Statistics Service) of climate change structured data. Differently, data in ARGSENSE is obtained through crawling and it is in textual form. Hence, natural language processing techniques (such as textual entailment) were used to aggregate arguments. These aggregation contributes to increase usability for policymakers and social change agents.

The two works above [1,3] have focused on understanding the views on climate change of *people*. Differently, Mayer et al. [21] have recently focused on understanding mental models on climate change of the *scientists*. Mayer et al. [21] have argued that understanding the decisions for model design are important for the informed use of tools built on that models. The research method has been to qualitatively analyse semi-structured interviews with eleven interdisciplinary experts (i.e., climate scientists, economists, decision analysts) who lead projects in the field of climate risk management. For managing climate change risks, Mayer et al. have been interested in analysing decision's *justification* and *explanations*. In the same line of focusing on expert knowledge and not public arguments, Huang et al. [12] have developed an expert system for integrating climate change impact in the petroleum industry with the aim to support formulation of the relevant adaptation policies. Similarly, Qin et al. [27] have proposed an expert system to assess the impact of climate change on socio-economic and environmental factors. These relevant factors are further used by the expert system to formulate adaptation policies. In this paper, we were interested to analyse, people's *arguments*. These may lead to an interesting interplay between arguments, justifications and explanations [17,30].

The VA-TURF tool [20] aims to assess the vulnerability of coastal fisheries ecosystems. VA-TURF includes socio-economical aspects and enhances planning of coastal communities to climate change impacts. The research method has been to assess vulnerability directly by fishers, barangay leaders, residents, and local executive staff. These stakeholders can discuss within the VA-TURF system on risks associated to a particular climate change scenario. ARGSENSE can also provide insights on a particular climate change topic. Yet the target communities are different in VA-TURF and ARGSENSE : VA-TURF encourages local community-level actions. Differently, ARGSENSE analyses arguments conveyed by a debate community that is globally distributed.

The ArgueApply has been launched in 2017 as a mobile application for generic debates [26]. The argumentation model of ArgueApply is based on four different relations: support, strong support, attacks, and strong attacks. The voting method counts strong support twice compared to the support relations. This refinement is necessarily in our view to overcome cases in which most of the semantics of abstract argumentation output the empty set. Differently, we have only the support and attack relations. Instead, our approach based on subjective logic can be used to distinguish between the following cases: i) let a debate topic supported by 3 arguments and rejected by 2; ii) consider also a debate topic supported by 30 arguments and attacked by 20. The ignorance level in our argumentation model allows to distinguish between such arguments. The topic with 30/20 arguments is considered more accepted in a debate community. The capacity of modeling ignorance level is why we used subjective logic for our argumentation model. Moreover, our ARGSENSE tool complements its vote-based method with text analysis capabilities (textual entailment, lexical analysis).

## 3. Methods and tool

This section describes 1) the climate change argument corpus that we crawled for our experiments, and 2) the architecture of the ARGSENSE tool that we developed to facilitate the analysis of online debates.

First, we created a corpus (denoted $cc$) for the *Climate Change* domain from the three debate sites we selected: ForAndAgainst (henceforth $faa$), Debate.org ($deb$) and Debatepedia ($dbp$). All debate hypotheses discussing climate change were filtered based on the Wikipedia glossary of climate change. First, the crawled opinions are automatically structured in tuples $\langle h, t, l\rangle$, where $h$ represents the debate hypothesis, $t$ the argument in natural language (optional, hence may be empty), and $l$ is the label of the vote pro (i.e., yes or agree) or cons (no or disagree) (see Table 1). Note that the label (pro or con) is provided by the conveyor of the argument. This label of the argument is automatically crawled from the webpage. This nice feature of the debate sites makes them an ideal source for extracting arguments which are already classified (i.e. labeled).

**Table 1.** Sample of tuples $\langle h, t, l\rangle$ in the climate change corpus.

| Hypothesis ($h$) | Argument ($t$) | Label ($l$) |
|---|---|---|
| Climate change is man-made. | Human carbon emissions have accelerated global warming ... | pro |
| Climate change is man-made. | The climate has changed through history due to natural cycles. | cons |
| Should government adopt emissions trading to combat global warming? | Emissions trading encourages investments in technologies. | pro |

There are 1,793 hypotheses in the corpus, and total 11,653 separate responses, i.e., arguments for the whole hypotheses repertoire. The $cc$ corpus was obtained by crawling three debate communities: $faa$ with 142 debates containing 877 arguments, $deb$ with 742 topics containing 6,026 arguments, $dbp$ with 909 debates on climate change attracting 4,750 arguments. With the resulted total of 11,653 arguments, the climate change corpus is, to our knowledge, the largest corpus of labeled arguments on climate change [3].

[3] The ARGSENSE tool and the climate change corpus are available at http://users.utcluj.ro/$\sim$agroza/projects/argclime.

Second, we built the ARGSENSE tool to support the analysis of people's opinions expressed in debate sites. The system is helpful for social scientists and policymakers in getting an insight into people's attitudes toward the controversial issues of worldwide interest. ARGSENSE has two architectural components relying on a *vote-based method* and *text-based methods* respectively (see Fig. 1).

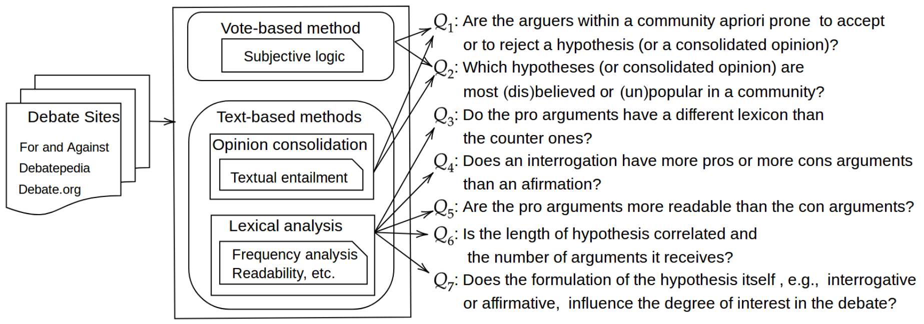

Debate Sites
For and Against
Debatepedia
Debate.org
Vote-based method
Subjective logic
Text-based methods
Opinion consolidation
Textual entailment
Lexical analysis
Frequency analysis
Readability, etc.
Q1: Are the arguers within a community apriori prone to accept or to reject a hypothesis (or a consolidated opinion)?
Q2: Which hypotheses (or consolidated opinion) are most (dis)believed or (un)popular in a community?
Q3: Do the pro arguments have a different lexicon than the counter ones?
Q4: Does an interrogation have more pros or more cons arguments than an afirmation?
Q5: Are the pro arguments more readable than the con arguments?
Q6: Is the length of hypothesis correlated and the number of arguments it receives?
Q7: Does the formulation of the hypothesis itself , e.g., interrogative or affirmative, influence the degree of interest in the debate?

**Fig. 1.** ARGSENSE investigation domain. A voting method based on subjective logic is proposed to rank the debate topics based on belief, disbelief and popularity in a community of arguers. An opinion consolidation method is proposed to aggregate arguments from related debate topics. This supports a more accurate view on the same questions $Q_1$ and $Q_2$. Lexical analysis uses off-shelf frequency analysis tools to support social scientists and science communicators with questions $Q_3$ to $Q_7$.

The vote-based method takes tuples $\langle h, t, l \rangle$ by crawling the debate sites and aggregates votes $l$ (of type pro or cons) for the same debate topic $h$. The aggregation is based on subjective logic. Subjective logic allows also to quantify belief and disbelief in $h$, but also the degree of ignorance in a community with respect to a debate topic $h$. The vote-based method helps a social scientist with answer to questions $Q_1$ and $Q_2$.

Text-based methods have two components: opinion consolidation and lexical analysis.

By *opinion consolidation* we mean the operation of aggregating arguments of semantic similar hypotheses. The semantic similarity relation is computed using textual entailment. Textual entailment identifies hypotheses representing the same debate topic, but posted using different words (e.g. *Climate change is manmade* and *Global warming is caused by humans*). Such related hypotheses can also be posted in different debate communities or posted in the same community but at different time points. To better support the social scientist, we need to consider all the arguments posted for or against all the hypotheses representing the same debate topic. We call this process opinion consolidation. Opinion consolidation is based on textual entailment and it represents the main conceptual proposal of this research.

Note that the vote-based method can be applied either on a single topic or on the consolidated topic that includes arguments from all related debates. In Fig. 1, this is illustrated by the fact that questions $Q_1$ and $Q_2$ can be applied both on a single hypothesis or on the consolidated debate topic. Throughout the paper, we use the term ”aggregation” for a summary of opinions (vote-based) under a specific hypothesis, while ”consolidation” is used whenever two separately posted hypotheses can semantically be merged.

Lexical analysis identifies linguistic features of argumentative texts. One can investigate if a community of people uses specific linguistic patterns, and whether these patterns depend upon the topic or whether the discourse is supporting or countering. The results are presented visually through graphs, rankings, and the identified lexical patterns. The methods used for lexical analysis are not new - we use readability indexes, sequential pattern mining, statistical analysis. Instead, these features help a social scientist or policymaker for answering questions $Q_3$ to $Q_7$.

## 4. Aggregation of arguments for an individual hypothesis

Now we describe the method for translating the individuals' arguments posted under one thread of particular hypothesis in one debate site into an *aggregated opinion*.

To represent aggregated opinions we use subjective logic [13,9], which originally was developed for belief representation in knowledge bases. In subjective logic, an opinion $\omega$ on a given state of a system $x$ is represented in terms of four quantities: $\omega_x = (b_x, d_x, u_x, a_x)$, where $b_x$ represents an individual's degree of belief that the particular state $x$ is true, $d_x$ stands for disbelief and shows the belief that a state is false, and $u_x$ is the uncertainty about the state. The parameter $a_x$ is a measure of the *prior* probability of the truth value of $x$. In our case, the state $x$ represents the hypothesis $h$ for which people have provided arguments.

Differently from [13], we prefer the term *ignorance* instead of *uncertainty*, as it fits better to our task of assessing the degree in which a community is interested in a specific topic. Differently from [13], we also introduce the notion of community, to count only the arguments conveyed within a community or arguers.

The aggregated opinion of a community $\alpha$ about a hypothesis *h* is defined by:

**Definition 1.** *The opinion $\omega_h^\alpha$ regarding the perceived truth value of hypothesis h by community $\alpha$ is a quadruple $\omega_h^\alpha = \langle b_h, d_h, i_h, a_h^\alpha \rangle$, where $b_h$ represents the degree of belief (amount of evidence supporting h), $d_h$ represents the disbelief (amount of evidence attacking h) and $i_h$ represents the degree of ignorance about h with*

$$b_h + d_h + i_h = 1, \qquad \{b_h, d_h, i_h\} \in [0,1]^3 \tag{1}$$

*The parameter $a_h^\alpha$ is a measure of the prior probability of the truth value of $h$ in the community $\alpha$. Hence, $a_h^\alpha$ is a feature of the community $\alpha$. With no apriori information about $\alpha$, we consider that a hypothesis has equal chances to be accepted or rejected.*

In our framework, evidence for $h$ are the arguments supporting or attacking $h$. For community $\alpha$, let $\mathcal{A}_h^+$ be the set of arguments supporting $h$, and $\mathcal{A}_h^-$ the set of arguments attacking $h$. Let $e_h = |\mathcal{A}_h^+|$ be the number of arguments supporting $h$, and $n_h = |\mathcal{A}_h^-|$ the number of arguments attacking $h$. The parameters $b_h$, $d_h$ and $i_h$ are computed with:

$$b_h = \frac{e_h}{e_h + n_h + 1/a_h^\alpha} \tag{2}$$

$$d_h = \frac{n_h}{e_h + n_h + 1/a_h^\alpha} \tag{3}$$

$$i_h = \frac{1/a_h^\alpha}{e_h + n_h + 1/a_h^\alpha} \tag{4}$$

Example 1 illustrates the opinion $\omega_h^\alpha$ for the $h$=*"Climate change is man-made"*.

*Example 1.* Assume $h$="Climate change is man-made" receives $A_h^+ = \{t_1, t_2, t_3, t_4, t_5\}$ and $A_h^- = \{t_6, t_7, t_8\}$. With no apriori information about community $\alpha$ ($a^0 = 0.5$), we have $b_h = 5/(5+3+2) = 5/10$, $d_h = 3/(5+3+2) = 3/10$, $u_h = 2/(5+3+2) = 2/10$. That is the opinion $\omega_h^\alpha = \langle 0.5, 0.33, 0.22, 0.5 \rangle$.

For particular values for $b_h$, $d_h$ or $i_h$, special types of opinions can be defined: i) *vacuous opinion*: $i_h = 1$ (maximum ignorance, when no argument is available for $h$); ii) *dogmatic opinion*: $i_h = 0$ (no ignorance; theoretically, this happens if the number of arguments is infinite); iii) *neutral opinion*: $b_h = d_h$; iv) *equidistant opinion*: $b_h = d_h = i_h$; v) *pure opinion*: $b_h = 0$ or $d_h = 0$; vi) *negative opinion*: $b_h < d_h$ (when $d_h = 1$ we have an *absolute negative opinion*);vii) *positive opinion*: $b_h > d_h$.

The fourth parameter $a^\alpha$ is global to the community $\alpha$ where $h$ is debated. With no apriori information regarding the acceptance of $h$ by a community of agents, $a^\alpha$ defaults to $0.5$. More accurate representation of $a^\alpha$ is obtained on the basis of the distribution of positive and negative opinions. Let $\mathcal{P}^\alpha$ be the set of hypotheses in a debate community $\alpha$ having more positive opinions than negative ones, given by $\mathcal{P}^\alpha = \{h \in \mathcal{H}^\alpha | e_h > n_h\}$. Let $\mathcal{N}^\alpha$ be the set of hypotheses in the community $\alpha$ having more negative opinions, given by $\mathcal{N}^\alpha = \{h \in \mathcal{H}^\alpha | n_h > e_h\}$. With this interpretation we have:

$$a^\alpha = \frac{|\mathcal{P}^\alpha|}{|\mathcal{P}^\alpha| + |\mathcal{N}^\alpha|}, \qquad \forall h \in \mathcal{H}^\alpha \tag{5}$$

The remaining $\mathcal{E}^\alpha = \mathcal{H}^\alpha \setminus \mathcal{P}^\alpha \setminus \mathcal{N}^\alpha$ is the set of neutral hypotheses in $\alpha$.

A topic $h$ is not necessarily independent from all other topics in the same community. There can be topics claiming the contrary of $h$ or topics claiming the same idea of $h$ but with different linguistic expressions. Therefore, we are interested next in exploiting these inter-relations between hypotheses in $\alpha$, to obtain a clearer and consolidated opinion.

## 5. Consolidation of opinions from related hypotheses

If two hypotheses are semantically close to each other, we may want to consolidate the opinions expressed for them, because it may give more information about people's attitude towards the underlying debate topic. Such hypotheses may be posted in one debate site or different ones. The question is then how to judge semantic closeness between two hypotheses. Our computational method uses three relations for semantic closeness: similarity, contradiction and entailment.

*Example 2 (Similar hypotheses).* Consider $g$="Climate change is manmade" and $h$="Global warming is human made". Since $g$ is similar to $h$, their supporting and attacking arguments can be aggregated.

Let $h, g \in \mathcal{H}^{\alpha}$, $e_h = |\mathcal{A}_h^+|$, $n_h = |\mathcal{A}_h^-|$, $e_g = |\mathcal{A}_g^+|$, $n_g = |\mathcal{A}_g^-|$.

**Definition 2 (Consolidating opinions for similar hypotheses).** *If $h$ is similar to $g$ ($h \sim g$) then the number of positive and negative arguments for computing the consolidating opinion $\hat{\omega}_h^{\alpha}$ are:*

$$\hat{e}_h = \hat{e}_g = e_h + e_g \tag{6}$$

$$\hat{n}_h = \hat{e}_g = n_h + n_g \tag{7}$$

*Example 3 (Contradictory hypotheses).* Let $g$="Climate change is a natural cycle". As $h$ claims the opposite of $g$, the supporting arguments for $h$ are the attacking arguments for $g$, while the supporting arguments for $g$ attack $h$.

**Definition 3 (Consolidating opinions for contradictory hypotheses).** *If $h$ contradicts $g$ ($h \sim \neg g$) then the number of positive and negative opinions for computing the consolidated opinion $\hat{\omega}_h^{\alpha}$ are:*

$$\hat{e}_h = \hat{n}_g = e_h + n_g \tag{8}$$

$$\hat{n}_h = \hat{e}_g = n_h + e_g \tag{9}$$

*Example 4 (Entailed hypotheses).* Let $k$="Climate-induced changes are likely to cause effects involving many species of plants and animals" and $l$="Animals can be affected by climate changes". As $k$ entails $l$, supporting arguments for $k$ also support the particular claim $l$. But the supporting arguments for $l$ do not necessarily support the more general hypothesis $k$. Instead, the attacking arguments of $l$ also attack $k$. Arguments attacking $k$ do not necessarily attack $l$.

**Definition 4 (Consolidating opinions for entailing hypotheses).** *If $h$ entails $g$ ($h \xrightarrow{ent} g$) then the number of positive and negative arguments for computing the consolidating opinion $\hat{\omega}_h^{\alpha}$ are:*

$$\hat{e_h} = e_h \tag{10}$$

$$\hat{e}_g = e_h + e_g \tag{11}$$

$$\hat{n}_h = n_h + n_g \tag{12}$$

$$\hat{n}_g = n_g \tag{13}$$

Three properties hold for our consolidation method:

1. less ignorance: based on the consolidated values $\hat{e_h}$ for supporting arguments and $\hat{n}_h$ for attacking arguments.
2. belief consistency: if $h$ entails another hypothesis $g$, then $b_h$ is expected to be smaller than $b_g$. That is: $(h \xrightarrow{ent} g) \Rightarrow (\hat{b}_h \leq \hat{b}_g)$.
3. sub-additivity of belief: if $\hat{b}_h + \hat{b}_{\neg h} < 1$.

The technical difficulty is to automatically identify these three relations: similarity, contradiction and entailment. For this task, we used the Excitement Open Platform for Textual Entailment (EOP) [19,24]. From EOP, the Biutee algorithm [28] was preferred due to its ability to interleave knowledge from lexical resources (e.g. WordNet, VerbOcean, Wikipedia) with the language model obtained with supervised learning. Biutee converts the text into the hypothesis via a sequence of transformations. The sequence of transformations is run over the syntactic representation of the text. On the parse tree, different entailment transformations can be applied, like lexical rules (e.g. CO2 $\rightarrow$ gas) or paraphrasing rules (e.g. A affects Y $\leftrightarrow$ Y is affected by X). As these relations are usually insufficient, they are complemented with transformations from a language model. The language model is learned based on a corpus of labeled pairs of text and hypothesis. The logistic-regression is the default algorithm used by Biutee. Given all possible transformations, Biutee applies the Stern et al. search algorithm [29] to find a proof that transforms the text into the hypothesis. The availability of this proof is another reason of using Biutee in our approach.

Algorithm 1 formalises our entailment-based method for computing consolidated opinions. The method starts by training the textual entailment machinery with the available tuples $\langle h, t, l \rangle$ of labeled arguments. Here we exploited the advantage that the arguments are already labeled as pro or cons by their own creators. Based on the labeled pairs, we used the max entropy classification algorithm to generate a language model for climate change arguments. The resulted model contains linguistic patterns in the climate change corpus for entailment and contradiction between each hypothesis $h$ and its supporting and attacking arguments $t$.

Our trick was to use this learned model to compute now the entailment relations $l$ between pairs of hypotheses $\langle h_1, h_2, l \rangle$ instead of a pair of hypothesis and one of its arguments $\langle h, t, l \rangle$. Hence, we fed Biutee (line 11) with: i) two hypotheses $h$ and $g$, ii) the model for the climate change corpus, and iii) lexical knowledge bases like WordNet or VerbOcean (see Algorithm 1). Biutee will interleave domain-specific knowledge (encapsulated in the $model$) and domain-independent knowledge (i.e. WordNet, VerbOcean) to search for contradictory or entailment relations between $h$ and $g$. If a contradictory relation is found, then the parameters $\hat{e}_h$ and $\hat{n}_h$ are computed based on Equations (8) and (9). If an entailment relation is found between $h$ and $g$, we check if the relation is symmetric (i.e. $g$ entails $h$ too). In this case (line 14), we consider the two hypotheses are semantically similar and equations (6) and (7) are applied. Otherwise, we apply equations (10), (11), (12) and (13). Note that the same hypothesis can be in various relations with other hypotheses at the same time: contradiction (line 9), entailment (lines 18-19), or similarity (lines 14-16).

## 6. Opinion aggregation and consolidation on the climate chance

This section applies opinion aggregation and opinion consolidation on the climate change corpus. We start by ranking the topics in the climate change corpus based on degree of belief, disbelief, or ignorance. Then we identify similar topics and we consider all their arguments in order to make a more clear picture on the ongoing debates.

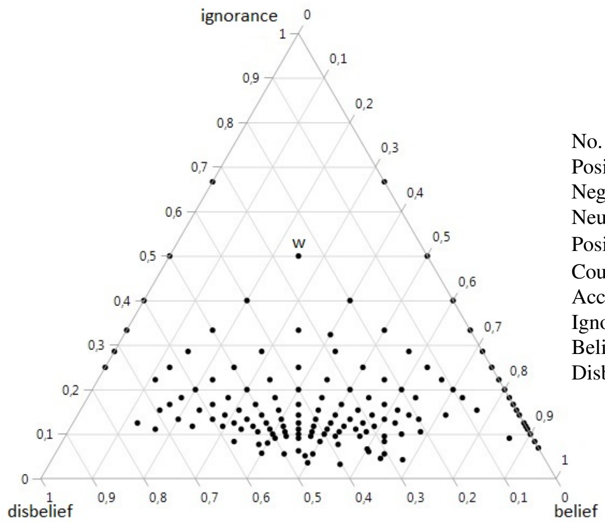
**Algorithm 1:** Consolidating opinions with textual entailment.

```
1:  **Input**: $\alpha$, corpus of hypotheses and labeled arguments $\langle h, t, l \rangle$
2:  **Input**: $kb$, lexical knowledge bases (e.g. WordNet, VerbOcean)
3:  **Output**: $\hat{\omega}_h^\alpha$, consolidated opinion for each $h$ in $\alpha$
4:  for $\langle h, t, l \rangle \in \alpha$ do
5:      $model \leftarrow trainBiutee(h, t, l)$
6:  end for
7:  for $h \in \mathcal{H}^\alpha$ do
8:      $e_h \leftarrow |\mathcal{A}_h^+|, n_h \leftarrow |\mathcal{A}_h^-|$
9:      for $g \in \mathcal{H}^\alpha \setminus \{h\}$ do
10:         $e_g \leftarrow |\mathcal{A}_g^+|, n_g \leftarrow |\mathcal{A}_g^-|$
11:         $rel \leftarrow BiuteeEntail(h, g, model, kb)$
12:         if $rel \equiv \neg$ then
13:             $\hat{e}_h = \hat{n}_g \leftarrow e_h + n_g$
14:             $\hat{n}_h = \hat{e}_g \leftarrow n_h + e_g$
15:         end if
16:         if $rel \equiv \xrightarrow{ent}$ then
17:             if $BiuteeEntail(g, h, model, kb) \equiv \xrightarrow{ent}$ then
18:                 $\hat{e}_h = \hat{e}_g \leftarrow e_h + e_g$
19:                 $\hat{n}_h = \hat{e}_g \leftarrow n_h + n_g$
20:             else
21:                 $\hat{e}_g \leftarrow e_h + e_g$
22:                 $\hat{n}_h \leftarrow n_h + n_g$
23:             end if
24:         end if
25:         $\hat{\omega}_h^\alpha \leftarrow computeConsolidatedOpinion(\hat{e}_h, \hat{n}_h, a^\alpha)$
26:     end for
27:     return $\hat{\omega}_h^\alpha$
28: end for
```

## 6.1. Opinion aggregation

The voting based method based on subjective logic applied on the climate change corpus provides insights regarding $Q_1$: *Are the arguers within a community apriori prone to accept or reject a hypothesis?* In the climate change corpus a hypothesis has on average 4.5 supporting arguments and 3.62 attacking arguments. With $|\mathcal{P}^{cc}| = 943$ positive hypotheses and $|\mathcal{N}^{cc}| = 453$ we have $a^{cc} = 0.67$. On average, the degree of belief is a little larger than disbelief. Hence, members of the communities from which the arguments were collected seem to be prone to accept a given hypothesis.

All hypotheses in climate change corpus are depicted with barycentric coordinates in Fig. 2. Closer to the top are the hypotheses with high ignorance. Neutral opinions are on the median from the top. On the right part are positive opinions, and on the left part are the negative ones. By pressing on each of the opinion point, ARGSENSE provides details on that hypothesis or set of hypotheses.



| | | |
|---|---|---|
| No. of hypotheses | cc | = 1,793 |
| Positive hypotheses | $\mathcal{P}^{cc}$ | = 943 |
| Negative hypotheses | $\mathcal{N}^{cc}$ | = 453 |
| Neutral hypotheses | $\mathcal{E}^{cc}$ | = 397 |
| Positive arguments | $\mathcal{A}^{+}_{cc}$ | = 6,662 |
| Counter arguments | $\mathcal{A}^{-}_{cc}$ | = 4,991 |
| Acceptance prone | $a^{cc}$ | = 0.67 |
| Ignorance interval | $i$ | $\in$ [0.04..0.66] |
| Belief interval | $b$ | $\in$ [0.14..0.93] |
| Disbelief interval | $d$ | $\in$ [0..0.72] |

**Fig. 2.** Depicting 1,793 hypotheses in the climate change corpus with barycentric coordinates. Each point depicts the set of hypotheses with the same coordinates. For instance, the opinion point $w = \langle 0.25, 0.25, 0.5, a^0 \rangle$ corresponds to 145 neutral hypotheses.

Figure 2 also shows that no vacuous opinions exist in our climate change corpus. The highest degree of ignorance is 0.66, given by hypotheses with only one argument. Still, there are 194 opinions with this high degree of ignorance, representing 11% from the total of 1793 hypotheses. Among them, 88% are pure positive and 12% are purely negative. With 35 positive arguments and 25 counter arguments, the hypothesis with the smallest degree of ignorance is *"Global warming is a natural cycle"*. Note that the second hypothesis with the smallest ignorance is *"Mankind is the main cause of global warming"*, which claims the opposite of $h_0$. There are 57 purely negative opinions, with the highest disbelief assigned to *"Is there a climate change conspiracy behind global warming and* *global cooling theories?"* (5 negative arguments and 0 positive). Instead, there are 396 purely positive opinions. The purely positive opinion with the highest degree of belief is *"The Kyoto protocol would harm the American economy"*. With 22 supporting arguments and 0 against, it has a belief of 0.91. The percentage of pure opinions (25%) is quite high. No equidistant opinion exists in the corpus. There are 22% neutral opinions. Most of them are supported by one argument and attacked by one argument. There are 53% positive opinions and 25% negative opinions.

**Table 2.** Answering to $Q_2$: Which hypotheses are most believed/disbelieved or popular/unpopular in a community?

| *Most believed hypothesis* | $e_h$ | $n_h$ | $b_h$ | $d_h$ | $i_h$ |
|---|---|---|---|---|---|
| The Kyoto protocol would harm the American economy. | 22 | 0 | 0.94 | 0 | 0.06 |
| Colonizing the Moon is critical for human survival. | 18 | 0 | 0.92 | 0 | 0.08 |
| Solar shading is a just response to irreversible global warming. | 18 | 0 | 0.92 | 0 | 0.08 |
| *Most disbelieved hypothesis* | $e_h$ | $n_h$ | $b_h$ | $d_h$ | $i_h$ |
| People can relax. Global warming is a sham. | 3 | 13 | 0.17 | 0.74 | 0.09 |
| Is cap-and-trade better at reducing emissions? | 3 | 13 | 0.17 | 0.74 | 0.09 |
| Are oil sands bad for climate change? | 3 | 10 | 0.21 | 0.69 | 0.1 |
| Is injecting sulphur dioxide into the atmosphere a good idea? | 3 | 8 | 0.24 | 0.64 | 0.12 |
| *Most popular hypothesis* | $e_h$ | $n_h$ | $b_h$ | $d_h$ | $i_h$ |
| Global warming is a natural cycle. | 35 | 25 | 0.57 | 0.41 | 0.02 |
| Mankind is the main cause of global warming. | 28 | 26 | 0.5 | 0.47 | 0.03 |
| Should we actually have a purge? | 19 | 18 | 0.49 | 0.47 | 0.04 |
| Manmade global climate change is real and a threat. | 16 | 18 | 0.45 | 0.51 | 0.04 |
| *Most unpopular hypothesis* | $e_h$ | $n_h$ | $b_h$ | $d_h$ | $i_h$ |
| Global warming causes earthquakes. | 1 | 0 | 0.4 | 0 | 0.6 |
| The sun causes global warming. | 1 | 0 | 0.4 | 0 | 0.6 |
| All natural disasters are related to global warming. | 1 | 0 | 0.4 | 0 | 0.6 |

By ranking the topics based on belief, disbelief, and popularity, ARGSENSE supports $Q_2$ as depicted in Table 2. Note that the most believed hypotheses are all pure opinions, given by no counter arguments for them ($d_h = 0$). Differently, none of the four most disbelieved topics is pure, given by the existence of pro arguments for them ($b_h > 0.17$). With 35 pro and 25 con arguments, the most popular topic has an ignorance of 0.02. Note that the first two most popular hypotheses belong to the same topic - real cause of global warming - but they claim opposite statements. From the ignorance value perspective, this result is consistent with the interpretation that the cause of global warming has been the most interesting topic for the arguers. From the psychological perspective, the result might indicate that the way in which the topic of the debate is formulated does influence the output of the debate: in both cases people seem to rather support the claim in the topic. The "natural cycle" hypothesis $h$ is supported with a belief of $b_h = 0.57$, while the "human cause" hypothesis $g$ is also supported with $b_g = 0.5$, even if $g$ claims the opposite thing as $h$. One might expect that believing in $h$ means a disbelief in $g$ or a belief in $g$ would be consistent with a disbelief in $h$. Given the nature of each debate, various factors may contribute to the above belief inconsistency at the community level.

From the opposite perspective, bottom part of Table 2 presents the hypotheses that people seem not to be interested in. There are 409 debates with only one positive argument and no attacking argument. Comparing the results of the most popular with the most ignored topics indicates that popular hypotheses are more general. Hidden variables, like time of issuing the debate, might be a cause of this lack of interest[4].

## 6.2. Consolidating opinions across hypotheses

The language model of climate change corpus was obtained by training the Biutee with on the $cc$ corpus with 6,662 entailment pairs and 4,991 non-entailment. The entailment pairs correspond to pairs of hypotheses with supporting arguments, while non-entailment correspond to pairs of hypotheses with attacking arguments. We used the max entropy classification algorithm to generate the language model. WordNet [22] and VerbOcean [8] were used as external knowledge resources. From Wordnet, the following relations were considered during the search process for a useful transformation: synonym, derivationally related, hypernym, instance hypernym, member holonym, part holonym, substance meronym, entailment. Only the first sense was used for a depth limit of 2 in the Wordnet taxonomy.

We illustrate one entailment proof computed with the Biutee method. The proof lists the transformations applied on the parse tree of the hypothesis $h_1$ (see Table 3). Knowledge from Wordnet and VerbOcean was used, but also the learned model from the training examples. The resulted sentence *Climate change can affect animals* is compared with the hypothesis $h_2$, and the entailment label is applied. Note that the confidence for this decision is 0.61. On the entire corpus we have a confidence of 0.65. This value is similar to cases in which texts are manually labeled by two human annotators. For instance, Kwon et al. have reported an inter-annotator agreement of 0.62 [16].

Having entailment/non-etailment relations computed for the debate topics, we can now apply our consolidation method to aggregate arguments of similar topics, as exemplified in the next subsection. To illustrate the consolidation method in case of entailment consider the pair of hypotheses *h="Mankind is the main cause of global warming"* and *g="Global warming is real"*. $h$ non-explicitly assumes that global warming is real and questions only its cause. Note that the assumption of $h$ is the claim in $g$. Therefore we consider $h$ entails $g$. In our corpus, we found that $h$ is supported by 28 arguments and attacked by 26, while $g$ is supported by 4 arguments and attacked by 4. That is $b_h = 0.5$ and $b_g = 0.46$. Because $h \xrightarrow{ent} g$ and $b_h > b_g$, the consistency property of belief does not hold for $h$ and $g$. Instead, after applying the consolidation method in case of entailment, the consolidated belief becomes consistent and also the ignorance decreases. Based on equations (10) and (11), $\hat{e}_g = e_h + e_g = 28 + 4 = 32$ and $\hat{n}_h = n_h + n_g = 26 + 4 = 30$, while $\hat{e}_h = e_h = 28$ and $\hat{n}_g = n_g = 4$. The consolidated opinion for $h$ is $\hat{\omega}_h^{cc} = \langle 0.47, 0.5, 0.03, 0.67 \rangle$ and for $g$ is $\hat{\omega}_g^{cc} = \langle 0.85, 0.11, 0.04, 0.67 \rangle$. As the consolidated belief $\hat{b}_h < \hat{b}_g$, the belief consistency property holds between the entailing hypothesis $h$ and $g$.

[4] For instance, the Marrakesh Climate Change Conference - November 2016 has not managed to trigger many debates, as all the debates site were invaded by debates related to the USA elections.

**Table 3.** Proof of entailment between two debate topics $h_1$ and $h_2$.

$h_1$: Climate-induced changes are likely to cause a series of cascading effects involving many species of plants and animals.
$h_2$ : Animals can be affected by climate changes.
Proof: Entailment, score = 0.6172
Substitute: "climate-induced"(JJ) to: "climate"(NN) (Multi-Word, remove words)
Substitute: "involve"(VBG) to: "affect"(VERB) with confidence:0.5 (Verb ocean)
Substitute: "animal"(NNS) to: "animals"(NNS)
Syntactic extraction rule: "Relative Clause - Extract reduced relative clause to independent sentence"
The part-of-sentence (bag of words) is: "affect effects "
Insert <(5) "by", IN, prep> under <(12) "affect", VERB>("by" costs -5.4143)
Insert <(2) "can", MD, aux> under <(12) "affect", VERB>("can" costs -7.5089)
Existing-Word-Insert <(7) "change", NNS, pobj> under <(5) "by", IN> ("change" costs -8.2133)
Existing-Word-Insert <(6) "climate", NN, nn> under <(7) "change", NNS> ("climate" costs -10.7516)
Syntactic substitution rule: "Coordination - Delete a verbal nominal or adjectival conjunct"
The part-of-sentence (bag of words) is: "and Animals plants "
Syntactic substitution rule: "Possessive - Substitute an "of-construction" with a nn modifier"
The part-of-sentence (bag of words) is: "Animals of species"
Move node <(18) "animals"> to <(12) "affect"> with relation 'nsubjpass'(costs -3.00, change context)

*Climate* changes are likely to cause a series of cascading effects involving many species of plants and animals.
Climate changes are likely to cause a series of cascading effects *affecting* many species of plants and animals.
Climate changes are likely to cause a series of cascading effects affecting many species of plants and *animal*.
Climate change *can by affecting* many species of plants and animals.
Climate change *can affecting* many species of animals.
Climate change *can affect* animals.

To illustrate the sub-additive property of consolidated belief, consider the contradictory hypothesis $h$=*"Mankind is the main cause of global warming."* and $k$=*"Global warming is a natural cycle"*. Semantically, $h$ is opposite of $k$. In the climate change corpus, the non-additive property does not hold for $h$ and $k$ ($b_h = 0.5$, $b_k = 0.57$). Instead, after applying the accrual of arguments in case of the contradictory relation, the belief becomes consistent and also the ignorance decreases. Based on equations (8) and (9), $\hat{e}_h = \hat{n}_k = e_h + n_k = 28 + 25 = 53$ and $\hat{n}_h = \hat{e}_g = n_h + e_k = 26 + 35 = 61$. the consolidated opinion for $h$ is $\hat{\omega}_h^{cc} = \langle 0.46, 0.53, 0.01, 0.67\rangle$ and for $g$ is $\hat{\omega}_g^{cc} = \langle 0.53, 0.47, 0.01, 0.67\rangle$. As for the consolidated belief $\hat{b}_h + \hat{b}_g = 0.46 + 0.53 = 0.99 < 1$, then the belief consistency property holds.

Opinion consolidation was used here as a general method for enriching the set of arguments for a given hypothesis, thus diminishing its ignorance.

## 7. Argumentative-text characteristics

Argumentative text characteristics are used by social scientists, policymakers or science communicators to better understand the communities of arguers and to design effective ways to communicate science or policies to target audience. ARGSENSE is able to analyse differences between linguistic patterns used in pro and counter arguments, to assess the correlation between the popularity of a debate with how the debate topic was posted, or to compute the readability of pro and counter arguments. We exemplify the lexical analysis of ARGSENSE by answering questions $Q_3$ to $Q_7$ on the climate change corpus.

$Q_3$: *Do the pro arguments have a different lexicon than the counter arguments?* Different lexicon might be an indicator to the social scientist that one party of the debate is sensible to different aspects as the other party.

To detect possible differences, we searched for the most frequent words in pro and cons arguments. For instance, if we denote by $f_{20}^+$ and $f_{20}^-$ the sets of the 20 most frequent words in the set of pros and cons, we obtained $f_{20}^+ \backslash f_{20}^- = \{emissions,\ greenhouse,\ water\}$ and $f_{20}^- \setminus f_{20}^+ = \{ice,\ increase,\ opponent\}$. These results suggest that proponents of climate change are concerned with emissions and greenhouse, while the opponents rise arguments related to ice. Interestingly, the ice related counter-argument is a common misconception related to climate change [11]. ARGSENSE was able to signal that this misconception is also spread over the debate sites.

$Q_4$: *When does a debate get more pros than cons, when formulated as a statement or as a question?* We are interested whether posting a hypothesis in affirmative or interrogative form could modify its chances to accumulate more arguments on one side or another. In the climate change corpus, 382 affirmative hypotheses received more pro arguments and 83 of them got more counter arguments. For interrogative topics, 561 got more pros and 370 more counter arguments. Fisher's exact test indicates a very strong statistical correlation ($p < 0.0001$) between the type of hypothesis and its chances to get more positive than negative arguments. The odds ratio value for the given example is 3.04, showing that the chances to have a winner are more than three times higher when the sentence is in affirmative than in interrogative.

$Q_5$: *Which is the readability of the arguments conveyed in a debate?* This provides an insight on the writing and reading comprehension skills of a community of arguers. An expert in science communication uses such readability indexes to adapt its arguments to the target audience. The science communicator should balance between simplifying the text and retaining technical details.

ARGSENSE is able to compare pro and counter arguments based on six readability indexes (Table 4). Coleman Liau and Automated Readability indexes rely on counting characters, words and sentences. The other indexes consider number of syllables and complex words. For more about readability formulas the reader is referred to [33]. No matter the readability index, the values for the positive and negative arguments are extremely similar. That is, no side uses more complex words than the other. The science communicator has to design ways to convey scientific results with the same readability indexes as the target audience or community or arguers [5].

**Table 4.** Readability indexes for pro ($\mathcal{A}_{cc}^+$) and against ($\mathcal{A}_{cc}^-$) arguments.

| Readability index | Flesch Kincaid Reading Ease | Flesch Kincaid Grade Level | Gunning Fog Score | SMOG | Coleman Liau | Automated Readability |
|---|---|---|---|---|---|---|
| $\mathcal{A}_{cc}^+$ | 58.40 | 8.73 | 11.21 | 8.73 | 11.80 | 8.01 |
| $\mathcal{A}_{cc}^-$ | 59.77 | 8.58 | 11.17 | 8.62 | 11.45 | 7.77 |

$Q_6$: *Is there a correlation between the length of a hypothesis and the number of its arguments?* We investigated whether heuristics like "the shorter the hypotheses, the more arguments" can be used by a debater to decide how to formulate the debate topic. The average number of words in $\mathcal{H}^{dbp}$ is 8.67. The correlation between the length of the hypothesis and the ignorance on it is -0,01. Similarly, the average number of words in $\mathcal{H}^{deb}$ is 9.67. The correlation between the length of the hypothesis and the ignorance on it is 0.12. Based on these two low values, we can conclude that for both communities $deb$ and $dbp$, the length of the hypothesis does not influence the number of arguments.

$Q_7$: *Does a query trigger more interest than a statement?* We analysed if a debate topic posted as *Will the planet adapt to global warming?* will attract more arguments than the version *The planet will adapt to global warming*. We evaluated the ignorance level for each hypothesis in the affirmative and interrogative form. Fig. 3 gives the cumulated percentages of hypotheses in affirmative and interrogative format for a given ignorance threshold. For example, if we specify an ignorance threshold 0.1, there are 12.75% of interrogative hypotheses, but only 7.40% of affirmative hypotheses. The percentage of interrogative hypotheses is always higher than its affirmative counterpart, which makes us believe interrogative hypotheses have higher chances to get more intense discussion.

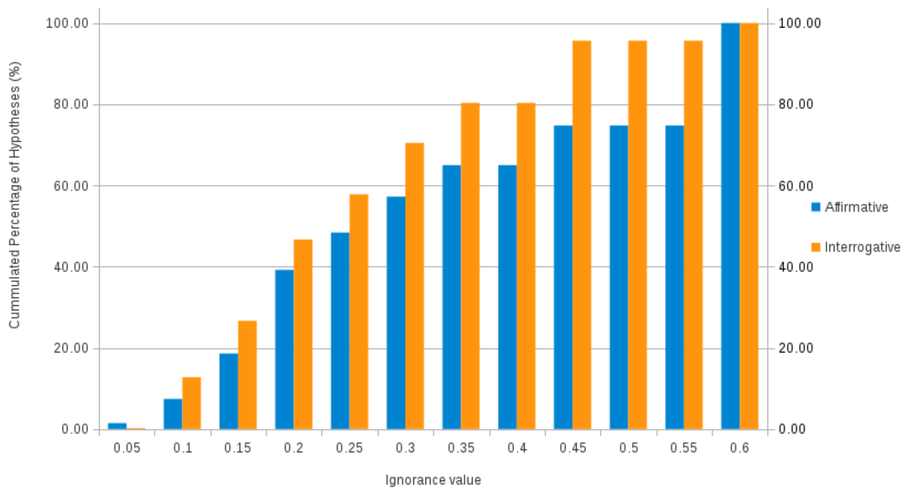


**Fig. 3.** Interrogative hypotheses attract more arguments than claiming hypotheses.

To summarise, the findings of lexical analyses related to questions $Q_3$ to $Q_7$ are: 1) proponents of climate change are more interested in: *emissions*, *greenhouse* and *water*, while the opponents convey more counter-arguments regarding *ice*; 2) affirmative hypotheses have three times higher chances to win compared to interrogative hypotheses ($p < 0.0001$); 3) pros and cons have the same readability values; 4) the length of the hypothesis does not influence the number of arguments for it; 5) interrogative hypotheses have higher chances to attract more arguments than in affirmative form.

## 8. Conclusions

We analysed arguments conveyed in public arena related to climate change. The four contributions of this research are: 1) the argumentative climate change corpus, 2) the

ARGSENSE social software for understanding public opinion on climate change, 3) the computational methods for aggregating and consolidating arguments, and 4) the lexical analysis of climate change arguments.

First, the climate change corpus is, to our knowledge, the largest corpus of labeled arguments on climate change.

Second, ARGSENSE makes sense of a set of arguments on climate change and aggregates them into usable results for policy making and climate science communication. ARGSENSE employs a voting method based on Subjective Logic to rank the debate topics according to their belief, disbelief or popularity within a community of arguers. Thus, ARGSENSE is in line with the recent trend to support scientific discovery [25] and to enhance the climate science cyber-infrastructure from *useful to usable* decision support tools [3].

Third, social sciences need to extend their instruments to be able to measure worldview on debate issues. Our method enhances the capabilities of social science to measure public support, disagreement or ignorance. It employs textual entailment to find similarity, contradiction or entailment between natural language arguments.

Fourth, we investigated in which way those interested in promoting public engagement need to pay attention towards linguistic aspects of communicating climate science. Our lexical analysis performed on arguments conveyed by people found that: 1) conveyors of pro arguments are interested in: *emissions*, *greenhouse* and *water*, while conveyors of con arguments in *ice*; 2) sequences in positive arguments do not overlap with sequences in negative arguments; 3) affirmative hypothesis have three times higher chances to win compared to interrogative hypotheses ($p < 0.0001$); 4) both pros and cons have the same readability; 5) no influence has been found between the length of the hypothesis and the number of arguments for it; 6) interrogative hypotheses have higher chances to attract more arguments than in affirmative form. Such lexical findings can be used by stakeholders to figure out how to communicate their point of view more effectively to the public.

One limitation of our approach regards the difficulty accurately assess the semantic similarity between topics. Even if the confidence in computing this semantic similarity is in the same range with the confidence of human annotators, it remains quite low: 0.65. This value is an average computed on the entire corpus. To tackle this limitation, one could explore the following two main directions. One direction is to apply the opinion consolidation method only to pairs of hypotheses for which the entailment/nonentailment is computed with high confidence. Another direction is to fine tune the parameters of the Biutee method related to: i) learning algorithm, ii) search process, or iii) external knowledge bases. We envisage the following possibilities. First, there are parameters of the learning algorithm used to build the language model. We run experiments only the max entropy classification algorithm. Second, there are parameters of the search step used to build the proof for entailment or nonentailment. We used all relations from the Wordnet and a depth limit of 2 in the Wordnet taxonomy. Third, one can add domain specific knowledge bases. That is, instead of relying only on general lexical resources (Wordnet, VerbOcean, Wikipedia) one can convert domain ontologies (for climate change in our case) to a rule-based format required by the Biutee method.

Another research line to be pursued is detecting repetitive arguments, either in verbatim copies or in semantically equivalent rephrasing. Here, we considered only the number of arguments and semantic similarity between topics. To overcome this, multiple dimensions can be considered, like argument provenance or time of issue. Such direction can be integrated into the larger context of research on fake arguments, collusion of argument proponents, or on how arguments propagate in public arena or in specific communities.

**Acknowledgments.** This work was funded by the EEA and the Norwegian Financial Mechanisms 2009-2014 under the contract 73264/02.09.2016 ”Increasing understanding on climate change through public discourse analyse and stakeholders modeling” between Norwegian University of Science and Technology, Trondheim, Norway and Technical University of Cluj-Napoca, Romania.

## References

1. Facilitating climate change adaptation through communication: Insights from the development of a visualization tool. Energy Research & Social Science 10, 57 – 61 (2015)
2. Batanović, V., Bojić, D.: Using part-of-speech tags as deep-syntax indicators in determining short-text semantic similarity. Computer Science and Information Systems 12(1), 1–31 (2015)
3. Biehl, L.L., Zhao, L., Song, C.X., Panza, C.G.: Cyberinfrastructure for the collaborative development of u2u decision support tools. Climate Risk Management 15, 90–108 (2017)
4. Boussalis, C., Coan, T.G.: Text-mining the signals of climate change doubt. Global Environmental Change 36, 89–100 (2016)
5. de Bruin, W.B., Morgan, M.G.: Reflections on an interdisciplinary collaboration to inform public understanding of climate change, mitigation, and impacts. Proceedings of the National Academy of Sciences p. 201803726 (2019)
6. Chalaguine, L.A., Schulz, C.: Assessing convincingness of arguments in online debates with limited number of features. In: Proceedings of the SR Workshop at the 15th Conference of the European Chapter of the Association for Computational Linguistics. pp. 75–83 (2017)
7. Chaniotakis, E., Antoniou, C., Pereira, F.: Mapping social media for transportation studies. IEEE Intelligent Systems 31(6), 64–70 (2016)
8. Chklovski, T., Pantel, P.: Verbocean: Mining the web for fine-grained semantic verb relations. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (2004)
9. Groza, A., Ozturk, P., Slavescu, R.R., Marginean, A., Prasath, R.: Analysing climate change arguments using subjective logic. In: 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP). pp. 37–44. IEEE (2018)
10. Ha-Duong, M.: Hierarchical fusion of expert opinions in the transferable belief model, application to climate sensitivity. International Journal of Approximate Reasoning 49(3), 555 – 574 (2008)
11. Hamilton, L.C.: Did the arctic ice recover? demographics of true and false climate facts. Weather, Climate, and Society 4(4), 236–249 (2012)
12. Huang, Y., Huang, G., Hu, Z., Maqsood, I., Chakma, A.: Development of an expert system for tackling the public’s perception to climate-change impacts on petroleum industry. Expert Systems with Applications 29(4), 817 – 829 (2005), http://www.sciencedirect.com/science/article/pii/S0957417405001077
13. Jøsang, A.: A logic for uncertain probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 9(03), 279–311 (2001)
14. Kirilenko, A.P., Stepchenkova, S.O.: Public microblogging on climate change: One year of twitter worldwide. Global Environmental Change 26, 171–182 (2014)
15. Kraemer, H.C.: Kappa coefficient. Wiley StatsRef: Statistics Reference Online (1982)
16. Kwon, N., Zhou, L., Hovy, E., Shulman, S.W.: Identifying and classifying subjective claims. In: Proceedings of the 8th Int. Conf. on Digital government research: bridging disciplines & domains. pp. 76–81. Digital Government Society of North America (2007)

17. Letia, I.A., Groza, A.: Arguing with Justifications between Collaborating Agents, pp. 102–116. Springer Berlin Heidelberg, Berlin, Heidelberg (2012), `http://dx.doi.org/10.1007/978-3-642-33152-7_7`
18. Lioma, C., Larsen, B., Schütze, H., Ingwersen, P.: A subjective logic formalisation of the principle of polyrepresentation for information needs. In: Proceedings of the third symposium on Information interaction in context. pp. 125–134. ACM (2010)
19. Magnini, B., Zanoli, R., Dagan, I., Eichler, K., Neumann, G., Noh, T.G., Pado, S., Stern, A., Levy, O.: The excitement open platform for textual inferences. In: Association for Computational Linguistics (System Demonstrations). pp. 43–48 (2014)
20. Mamauag, S.S., Alino, P.M., Martinez, R.J.S., Muallil, R.N., Doctor, M.V.A., Dizon, E.C., Geronimo, R.C., Panga, F.M., Cabral, R.B.: A framework for vulnerability assessment of coastal fisheries ecosystems to climate change - tool for understanding resilience of fisheries (VA-TURF). Fisheries Research 147, 381 – 393 (2013), `//www.sciencedirect.com/science/article/pii/S0165783613001719`
21. Mayer, L.A., Loa, K., Cwik, B., Tuana, N., Keller, K., Gonnerman, C., Parker, A.M., Lempert, R.J.: Understanding scientists' computational modeling decisions about climate risk management strategies using values-informed mental models. Global Environmental Change 42, 107 – 116 (2017), `http://www.sciencedirect.com/science/article/pii/S0959378016306197`
22. Miller, G.: WordNet: An electronic lexical database. MIT press (1998)
23. Mohammad, S.M., Kiritchenko, S., Sobhani, P., Zhu, X., Cherry, C.: A dataset for detecting stance in tweets. In: Proceedings of 10th edition of the the Language Resources and Evaluation Conference (LREC), Portoroz, Slovenia (2016)
24. Padó, S., Noh, T.G., Stern, A., Wang, R., Zanoli, R.: Design and realization of a modular architecture for textual entailment. Natural Language Engineering 21(02), 167–200 (2015)
25. Pankratius, V., Li, J., Gowanlock, M., Blair, D.M., Rude, C., Herring, T., Lind, F., Erickson, P.J., Lonsdale, C.: Computer-aided discovery: Toward scientific insight generation with machine support. IEEE Intelligent Systems 31(4), 3–10 (July 2016)
26. Pührer, J.: Argueapply: A mobile app for argumentation. In: International Conference on Logic Programming and Nonmonotonic Reasoning. pp. 250–262. Springer (2017)
27. Qin, X.S., Huang, G.H., Chakma, A., Nie, X., Lin, Q.: A MCDM-based expert system for climate-change impact assessment and adaptation planning–a case study for the Georgia Basin, Canada. Expert Systems with Applications 34(3), 2164–2179 (2008)
28. Stern, A., Dagan, I.: The BIUTEE research platform for transformation-based textual entailment recognition. LiLT (Linguistic Issues in Language Technology) 9 (2014)
29. Stern, A., Stern, R., Dagan, I., Felner, A.: Efficient search for transformation-based inference. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. pp. 283–291. Association for Computational Linguistics (2012)
30. Svrcek, M., Kompan, M., Bielikova, M.: Towards understandable personalized recommendations: Hybrid explanations. Computer Science and Information Systems 16(1), 179–203 (2019)
31. Walton, D., Reed, C., Macagno, F.: Argumentation schemes. Cambridge Univ. Press (2008)
32. Washington, H.: Climate change denial: Heads in the sand. Routledge (2013)
33. Weiss, K.D., Vargas, C.R., Ho, O.A., Chuang, D.J., Weiss, J., Lee, B.T.: Readability analysis of online resources related to lung cancer. Journal of Surgical Research 206(1), 90–97 (2016)
34. Zhang, H., Alim, M.A., Li, X., Thai, M.T., Nguyen, H.T.: Misinformation in online social networks: Detect them all with a limited budget. ACM Trans. Inf. Syst. 34(3), 18:1–18:24 (Apr 2016), `http://doi.acm.org/10.1145/2885494`

**Adrian Groza** is assoc prof. at the Dept. of Computer Science at Technical University of Cluj-Napoca, Romania. His current research regards knowledge representation, argumentative agents, explainable AI.

**Pinar Ozturk** is assoc. prof. at the Dept. of Computer and Information Science at Norwegian University of Science and Technology (NTNU). Before joining NTNU, Ozturk was a senior research scientist at SINTEF. Ozturk's research area is artificial intelligence methods for decision making.

**Radu Razvan Slavescu** got his PhD from the Technical University of Cluj-Napoca, Romania with a thesis focused on trust modeling and its application in multiagent systems. Now, Radu is with the Intelligent Systems group at the Computer Science Department of the same University. His domains of interest include Artificial Intelligence, Multiagent Systems and trust modeling, and Natural Language Processing, in which he has authored 40+ papers.

**Anca Marginean** got her PhD fromthe Techical University of Cluj-Napoca, Romania in 2013. Her interests regard natural language processing and Semantic Web.

*Received: October 11, 2018; Accepted: September 17, 2019.*

Computer Science and Information Systems17(1):117–139 https://doi.org/10.2298/CSIS181118018P

# Architecting Business Process Maps

Geert Poels[1], Félix García[2], Francisco Ruiz[2], and Mario Piattini[2]

[1] Faculty of Economics and Business Administration,
Ghent University, Belgium
geert.poels@UGent.be
[2] Institute of Information Technology and Systems
University of Castilla-La Mancha, Spain
{felix.garcia, francisco.ruizg, mario.piattini}@uclm.es

**Abstract.** Process maps provide a high-level overview of an organization's business processes. While used for many years in different shapes and forms, there is little shared understanding of the concept and its relationship to business process architecture. In this paper, we position the concept of process map within the domain of architecture description. By 'architecting' the concept of business process map, we identify and clarify diverging views of this concept as found in the literature and set requirements for describing process maps. A meta-model for a process mapping language is produced as a result. The proposed meta-model allows investigating the suitability of EA modelling languages as a basis for defining a domain-specific language for process mapping along with the creation of a better understanding of business process architecture in relation to enterprise architecture, which can be beneficial for both BPM and EA professionals.

**Keywords:** Process map, Business process architecture, Enterprise architecture, Architecture description, Domain-specific modelling.

## 1. Introduction

Recently, business process architecture design has received attention in Business Process Management (BPM) research [1]. Business process architecture is commonly defined as the organised overview of the processes that exist within an organisation, including their relationships [2]. As the output of the BPM lifecycle process identification phase during which the organisation's business processes are designated and prioritised, the business process architecture provides the basis to single out the processes that will be subjected to further BPM lifecycle activities. Like modelling individual processes is a starting point for any BPM effort [2], modelling the architecture of an organisation's collection of business processes is required for any analysis, design or improvement effort that transcends the level of individual processes (e.g., multi-process analysis [3]). Process architecture has further been positioned as an important instrument for managing large collections of process models in organisations that have already invested heavily in BPM [4]. Process architecture is also essential for process portfolio management [5] and improvement initiatives that concern multiple processes like process standardisation efforts and the identification of shared services [6]. Research in Goal-Oriented Requirements Engineering (GORE) has resulted in a few methods for (re)designing business process architecture in alignment with business goals [7-9], though most GORE research on business processes relates to goal alignment or goal-driven design of individual business processes rather than entire process architectures [10]. In Enterprise Architecture (EA), the business process architecture is considered an integral component of the business layer of an organisation’s enterprise architecture, where processes are managed as assets that are vital to the organisation’s operations [11, 12]. Meanwhile, different kinds of models have been proposed for representing specific views on an organisation’s business process architecture, like business process co-operation models [13], process chain models [14], process landscape models [2] and process maps [6]. In particular, the concept of process map as a holistic and abstract representation of an organisation’s business processes, has only recently been investigated [15], while being in use for many years in different shapes and forms. In practice, there is little shared understanding of the concept, related to its contents, form, purpose, and relationship to business process architecture. According to [16], the current variety in process maps that can be observed might be due to the lack of modelling language dedicated to expressing process maps. The need for designing such language, preferably building upon a general-purpose modelling language, has been expressed by many researchers [1, 17, 18].

Although the modelling of business processes, their interrelationships, and their linkages with strategic, operational, informational or infrastructural business and information technology elements is part of several enterprise modelling approaches (e.g., EKD [19], ARIS [20], TBIM [21], MEMO [22], 4EM [23], PGA [24]), these types of models have not been positioned as general solution for articulating process maps as they are part of and make sense in the context of a specific enterprise modelling approach. Some researchers have proposed requirements for a general-purpose process mapping language [6, 18], while Malinova and Mendling [15] have proposed a meta-model for process maps that sets requirements for a process map representation language. Malinova and Mendling [15] further showed that BPMN is not ontologically expressive enough for meeting these requirements, and therefore a process mapping language needs to be designed. Apart from not having a process mapping language, there is lack of clarity in the conceptualisation of the process map in relation to business process architecture. Specifically, in the literature there are substantial differences in conceptualization of business process architecture, process map and their mutual relation. Further, only few works on modelling business process architecture hint at positioning an organization’s business process architecture within the broader enterprise architecture, though without elaborating the idea.

The goal of the research presented in this paper is to provide a conceptualization of process maps in the context of enterprise architecture by considering a process mapping language as an architecture description language. As a result, we conceptualize the process map as an enterprise architecture artefact and propose a meta-model for a business process architecture description language that can be used to represent process maps. By ‘architecting’ the concepts of business process architecture and process map we clarify diverging views of these concepts as found in the literature and set requirements for describing process maps. An integration of the current BPM research on process maps with EA thinking could lead to advancement in the field and increased knowledge sharing, and opens up new possibilities for research on the boundaries of BPM and EA [25]. It could also facilitate the harmonized use of general-purpose modelling languages from both fields (e.g., BPMN and ArchiMate).

The remainder of this paper is structured as follows: Section 2 presents the research methodology. Section 3 presents the background of the research, i.e., the ISO/IEC/IEEE 42010 standard for architecture description, and the related work. Section 4 describes the design of the meta-model based on our conceptualization using the defined requirements as design principles. Section 5 empirically evaluates the meta-model by instantiating it for known classifications of business process architecture descriptions and elaborate examples of process maps found in the literature. The meta-model is also formally evaluated by verifying general requirements for defining domain-specific languages. Finally, Section 6 presents the conclusion and future work.

## 2. Research Methodology

The development of a new enterprise modelling solution encompassing amongst others a language defined by a meta-model (i.e., abstract syntax and semantics of modelling constructs), a modelling notation (i.e., concrete syntax and notational conventions), and modelling guidelines and tool support (i.e., pragmatics of using the language and notation) – for a full set of requirements see e.g., [22] – can be undertaken as Design Science Research [26]. As several researchers have already noted the lack of a universal process mapping language and have motivated the need for its design, we engaged in an objective-centred initiation of a Design Science Research (DSR) project (Figure 1) [27], where the first research activity is the definition of the objectives of a solution for the identified problem. In this paper, we instantiate these solution objectives as a meta-model for a business process architecture description language that can be used to represent process maps, consisting of modelling constructs, their relationships and constraints – leaving other language requirements (e.g., notational, tool support) outside the scope of the current paper. Following [28], the design of a new meta-model initiates a new research cycle embedded in the overall DSR project.

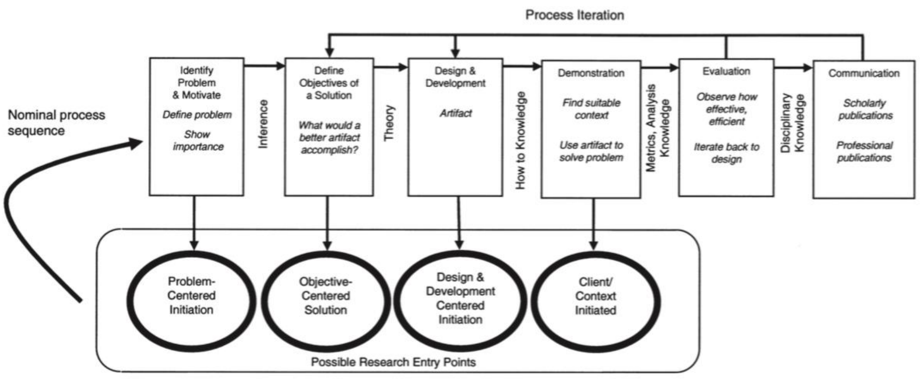

Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artifact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development Centered Initiation
Client/ Context Initiated
Possible Research Entry Points

**Fig. 1.** Design Science Research process [27]

In this embedded DSR project, the following research steps were taken:

**(1)** Literature review of research on process maps and modelling of business process architecture. Research presenting requirements for process mapping, informal solutions, and reviews of design approaches for business process architecture was analysed. The result was an inference of different perspectives on process maps in relation to business process architecture, indicating a lack of ‘architectural point of view’, which motivated our research (i.e., *identify problem and motivate* in Figure 1) (Section 3).

**(2)** Analysis of the reviewed process map and business process architecture concepts from an architectural point of view, using the ISO/IEC/IEEE 42010 software and systems engineering international standard for architecture description [29] as conceptual frame of reference. This standard provides a core ontology (i.e., ‘theory’ in Figure 1) for the description of architectures, that we used as a conceptual reference framework for clarifying the relationship between the business process architecture and process map concepts. Contextualising these concepts according to this architecture description standard clarifies their mutual relationship, reveals the diversity that is present in the inferred perspectives from the literature, and guides proper choices regarding assumptions and requirements for business process architecture description (i.e., *define objectives for a solution* in Figure 1) (Section 3).

**(3)** Guided by the developed contextualization within architecture description, we first recovered a conceptualization of process map as business process architecture model [30]. We critically assess existing proposals of process map conceptualizations, along with proposed requirements for a process mapping language and informal solutions that have been used in the absence of a standard process mapping language. We then developed a new meta-model for business process architecture description that can be used to guide the development of a general representation language for process maps (i.e., *design and development* in Figure 1) (Section 4).

**(4)** To demonstrate the meta-model’s ability to guide the design of a universal representation language for process maps, we instantiated it to represent a wide array of process maps and other business process architecture descriptions found in the literature which are currently only informally described or represented using different notations (i.e., *demonstration* in Figure 1) (Sub-Sections 5.1, 5.2, and 5.3).

**(5)** To evaluate the meta-model, we analysed the meta-model instantiations used in the demonstration for evaluating the meta-model’s ability to uniformly represent different kinds of process map (Sub-Sections 5.1, 5.2, and 5.3). In practice, this was an iterative process as representing those process maps allowed us to refine the design of the meta-model until all process maps were valid instantiations of the meta-model. Apart from that, we verified the satisfaction of core requirements for defining domain-specific modelling languages, as identified in the literature [31] (i.e., *evaluation* in Figure 1) (see Sub-Section 5.4).

**(6)** Presentation of the meta-model and its DSR research process (i.e., *communication* in Figure 1) (i.e., the purpose of this paper).

## 3. Background

In this Section the background of the presented proposal is described and the related work that we analysed to recover a conceptualization of process map consistent with this background is presented. As conceptual reference framework for imposing an architectural point of view on business process architecture and process maps as models of business process architecture we used the ISO/IEC/IEEE 42010 software and systems engineering international standard for architecture description [29]. This standard presents a conceptual model of architecture description that can be applied to any kind of architecture, including business process architecture. The standard also specifies desired properties for architecture descriptions, which result in requirements for architecture frameworks and architecture description languages such that desired properties are exhibited by the architecture descriptions that are developed using these frameworks and languages. The concepts and requirements provided by the standard can be used to guide the design of a business process architecture description language, which can be used to represent process maps. The mapping to business process architecture description of the standard's architecture description concepts is summarized in Table 1.

**Table 1.** Process map and process mapping language in terms of business process architecture description

| ISO/IES/IEEE 42010 standard- architecture description concept | Mapping to business process architecture description |
|---|---|
| System | (Collection of) business processes |
| Environment | Organization<br>Examples company, not-for-profit organization, university, business unit of a company |
| Stakeholder | Examples operational managers, process/domain architects, CPO, business managers, enterprise architects |
| System Concern (associates System and Stakeholder) | Examples consistency and completeness, dependencies between processes, responsibilities, performance, strategic fit |
| Purpose (is a System Concern) | Efficient organization of the work to be performed in the organization |
| Architecture | Business process architecture |
| Architecture Description | Business process architecture description |
| Architecture Viewpoint | Example E2E processes viewpoint |
| Architecture View | Example E2E processes view |
| Model kind | Example E2E processes model kind (e.g., meta-model) |
| Architecture Model | Example E2E processes model (i.e., *process map*) |
| Architecture Framework | Business process architecture framework |
| Architecture Description Language | Business process architecture description language (i.e., *process mapping language*) |

The standard also defines the concepts of architecture framework and architecture description language (ADL) as mechanisms that can be used for creating and employing architecture descriptions. TOGAF [32] for instance presents an architecture content framework identifying business processes as an architectural artefact, but refers to ArchiMate [13] as possible ADL to provide a notation for modelling business processes. Regarding the definition of business process architecture, we noticed in the literature two main perspectives:

1. The process architecture as the organization of the business processes in terms of their boundaries, dependencies, priorities, criticality, strategic importance, linkage with functional domains, etc. [1, 2]. In this perspective, the business process architecture is used to select processes that will be subjected to analysis and improvement actions (BPM) or to design or align the organization's system of business processes in relation with other organizational assets, goals and strategies (EA, GORE).
2. The process architecture as the organized collection of business process models and their relationships [4]. In this perspective, the business process architecture is used to categorize and manage process models [33] and to maintain the consistency between process models [18].

While these perspectives are not per se incompatible – the process architecture as a guide to initiate BPM and once the BPM initiative is ongoing as an organized overview of the resulting business process models – there are implications for the definition of the process map. In the first perspective, the process map is a model of the business process architecture, while in the second perspective it is part of the business process architecture, which would in that case more appropriately be called the business process models architecture. In the second perspective, the collection of business process models is usually hierarchically structured into several layers of modelling abstraction, resulting in a business process models decomposition tree, starting from the more abstract process models at the top to the more concrete process models at the bottom. While this decomposition can be organized in different ways, either or not ensuring consistency between models at different abstraction levels [34], the process map is typically seen as the entry-level model at the top of the hierarchy, providing a holistic and abstract overview of all or the main processes and their relationships [33].

In the first perspective, the business process architecture can also be hierarchically structured, but now according to increasing levels of granularity. In this context granularity refers to what is being considered as the atomic element of a business process architecture. In a flat (i.e., non-hierarchical) business process architecture, also called process landscape [14], there is only one atomic element and that is the business process. Business processes can be ordered, grouped, decomposed and specialized (whatever type of relation is recognized; see Section 4), but all process steps, process group members, sub-processes and process variants still qualify as business processes. On the contrary, in a hierarchical business process architecture, the atomic element considered at lower levels is more fine-grained than the atomic element considered at higher levels. For instance, the process architecture discussed in [2] defines three levels of granularity with respectively business processes, activities, and tasks as atomic elements. Sometimes the top level of a proposed hierarchy has a more coarse-grained atomic element than the business process. For instance, Van Nuffel and De Backer [18] consider the main business process, representing a process family, as atomic element for the top level, whereas the elementary business processes that are process variants in these process families are only considered at the second level.

Regardless whether the process map is defined as a model of the business process architecture (i.e., first perspective) or as entry-level model of the business process models architecture (i.e., second perspective), other differences surface. These differences emerge as a result of variations in the assumed structure of the business process architecture. Whereas in a flat architecture the scope of the process map is the entire process landscape [6], in a hierarchically structured business process architecture, the scope is typically limited to the top level. For instance, in [18] the process map describes the top level in a five-level process architecture and thus models the main business processes of an organization. On the other hand, in [2] the process map is positioned as a model of the second hierarchical level where it describes the main flow of process activities. Some literature also recognizes that a process map may only partially model the process architecture within its scope and is thus part of a view on the architecture [4]. A process map as part of a view on the business process architecture is an abstraction that serves some purpose. For instance, the requirements for process maps specified in [6] define an abstraction that is useful for identifying sub-processes that can be further investigated for being standardized. Few authors, however, explicate the intended use of process maps.

In summary, the current proposals of desiderata for process mapping are hard to compare and evaluate, unless an architecture viewpoint has been explicated. Purposes listed in the literature can be very general (e.g., representation) or very specific (e.g., identifying functional similarities). We found only one paper (i.e., [18]) that explicitly distinguishes process maps according to several different views, however, without defining the corresponding viewpoints. In general, there is a lack of explicit definition of viewpoints identifying stakeholders in the organization's system of business processes and the concerns of these stakeholders. Furthermore, it is clear that the different proposals for process map representation make their own (and generally implicit) assumptions about architectural viewpoints of the business process architecture description, which along with differing assumptions about the nature and structure of the business process architecture and its relation to the process map, result in lack of consensus on the requirements for and design of a general-purpose process mapping language.

## 4. Designing the Metamodel

Following [35] and [36] on the difference between ontology and meta-model, we move with the meta-model beyond the conceptualization of process map within the business process architecture description domain (i.e., ontology development), by supporting the computerized representation of business process architecture models (i.e., domain-specific modeling language development). Prior to the development of the metamodel a conceptual analysis of process maps was conducted. Despite the absence of explicitly defined business process architecture viewpoints in the related literature (see Section 3), there is one proposal that is similar to the design of an ADL for business process architecture description as it provides a model kind for process maps. The process map meta-model of Malinova and Mendling [15] is to the best of our knowledge unique in its kind. The proposed meta-model is positioned by its designers as a model of a process map conceptualization rather than a formal meta-model defining a process mapping language, which makes it a valuable starting point for our conceptual analysis. Its embedding in BPM research results, however, in a number of assumptions related to the use of process maps prior to BPM implementation (i.e., process identification) and during BPM implementation (i.e., process model management). Referring to the perspectives discussed in the previous Section, the meta-model conceptualizes the process map as an entry-level model of a hierarchically structured business process models architecture (i.e., the second perspective discussed in Section 3). We therefore complement the conceptual analysis with other related work that positions the process map as a model (i.e., abstraction) of the business process architecture (regardless what view is abstracted). The goal is to arrive at an elaborate conceptualization that covers not only the proposed meta-model but also other proposals even if only informally described or just based on a set of requirements or example notation. As a result of the conceptual analysis of process maps based on the reviewed literature, the following requirements were formulated:

**Req. 1**: The business process is the atomic element of the process map.

**Req. 2**: It should be possible to show on a process map the enterprise architecture elements that a business process (composite) is related to.

**Req. 3**: It should be possible to show on a process map composites of business processes that result from the application of different types of process relations.

- ***(Req. 3a)*** *Sequencing relations*: The execution of business processes may be ordered in time meaning that the execution of a first process is followed by the execution of a second process. Such ordering relations typically indicate that processes are steps in a process chain that serves a higher-level goal. For instance, the requisition process and the purchasing process are steps of the Purchase-to-Pay (P2P) end-to-end process where requisition is performed before purchasing. Identifying ordering relations is important as changes applied to a prior process may affect the design and execution of a subsequent process.
- ***(Req. 3b)*** *Decomposition relations*: A business process can be a sub-process of another business process, like the sales order data entry process that is a sub-process of the sales order handling process. The steps of a process chain are sub-processes of the process chain. Decomposition can also take the form of shared aggregation. For instance, a customer data verification process can be a sub-process of both a sales process and an after-sales service process. Decomposing business processes into sub-processes relates processes hierarchically which is important as BPM actions taken on sub-processes may affect their superordinate processes.
- ***(Req. 3c)*** *Grouping relations*: Business processes can be related through joint membership of a process group. From the moment business processes have something in common, a process group can be defined. For instance, a credit sales process and a cash sales process are members of the group of sales processes. Both processes share the goal of selling products or services to customers, but differ in the manner in which customers pay for their sales. In principle, any property of processes can be used to form process groups. Defining process groups allows abstracting from certain differences between

processes to see ‘the forest through the trees’, which can be important especially for organizations with large numbers of business processes.

- ***(Req. 3d)*** *Specialization relations*: A business process can be a specialized version of another business process, like the job student recruitment process that specializes the personnel recruitment process. A business process and its child processes form together a process family in which the child processes are process variants and the parent process becomes a standard process for these process variants. A process group, like the sales processes group, can be a process family, but is not necessarily so as there might be no standard sales process defined. Identifying specialization relations is important as changes applied to a parent process may have consequences for the child processes. Note that the implementation of specialization is not considered at the abstraction level of the business process architecture. One approach for instance is to define variation points in a standard process, which can be filled differently for the process variants [37].

The solution to these requirements which guided the design of the process maps metamodel can be summarized as follows (detailed explanations are included in [30]):

- We chose to restrict the use of process maps (as a business process architecture model) to *black-box modelling* of organization’s business processes (Req. 1).
- We generalize existing proposals of including process-related elements in a process map by means of an *Enterprise Architecture Element* that can be instantiated in process maps according to needs (depending on business process architecture viewpoint) (Req. 2).
- We recognize the need to represent in a flexible and extensible manner *Business Process Composites* where business processes can be aggregated to higher-level concepts reflecting different internal structures depending on the types of relation between the processes in the composite (Req. 3).

Therefore, our metamodel design philosophy was driven by the conceptualization of the process map as a business process abstraction that provides a black-box view on the organization’s business processes, the search for maximal integration with EA description assuring robustness and extensibility of the meta-model, and the recognition of different kinds of business process composites. In addition, to cater for expressing any business process architecture viewpoint, we need to allow maximum freedom for instantiating the meta-model according to process mapping needs, thereby limiting the number of constraints at the meta-model level. The result of our design is shown in Figure 2 as a UML class diagram.

As it can be observed in Figure 2, the core concept of the meta-model is the ***Business Process Architecture Element***, which is shown as an abstract class. We prefer the term business process architecture element to business process as a business process architecture does not necessarily include all organizational business processes [18], hence only the business processes and their composites that are part of the business process architecture are represented in process maps. Also, according to the architecture description standard, the system’s architecture is what is essential about the system considered by the system stakeholders and their concerns. For instance, while Rosemann and vom Brocke [38] include in their ‘enterprise process architecture’ all processes of an organization, [2] include only those processes that have been identified in the first phase of the BPM cycle [39]. As noted before, the process map conceptualization by Malinova and Mendling [15] is strongly based on the process landscape level of the three-level process architecture in [2], but then seen as entry-level model to the business process models architecture. To allow for different business process architecture viewpoints, we thus prefer the use of the term business process architecture element. The abstract business process architecture element is either an ***Elementary Business Process*** or a ***Business Process Composite***. According to Req. 1, a process map provides a black-box model of the elementary business processes in the business process architecture, meaning that the internal structure and operation of the elementary business processes is hidden [18]. An *Elementary Business Process* is thus an atomic business process architecture element [15]. The concept of business process composite is a new notion that we introduce because of Req. 3 and which is not present in the reviewed literature. We obtain it by applying the composition pattern [40]. A *Business Process Composite* can thus simply be defined as any business process architecture element that is not an elementary business process. By applying the composition pattern, a business process composite can be disaggregated into other business process composites or elementary business processes. A business process composite is thus a non-atomic business process architecture element. Even if non-atomic, a process map may show a business process composite without showing its parts. That is why there is no minimum cardinality constraint on the parts of a business process composite. Through the composition pattern we include the *decomposition* relation (Req. 3b) in the meta-model, in the form of unrestricted shared aggregation. Business process architecture decomposition structures can extend over multiple levels. The only constraint included in the meta-model is that elementary business processes cannot be disaggregated (as this would violate Req. 1). When instantiating the meta-model for developing process maps according to specific business process architecture viewpoints, additional constraints can be imposed. An example of such constraint could be that the leaves in a decomposition structure can only be elementary business processes.

The meta-model (Figure 2) shows three subclasses of business process composite: process group, process chain and process family. The specialization is partial meaning that there can be other business process composites than these three. We include these three composites as a specific type of relation between business process architecture elements defines each of them:

- ***Process Chain*** is an aggregate of business process architecture elements that are related through *sequencing* relations (Req. 3a), meaning that there is a sequential ordering amongst these elements. The roles of prior and subsequent process as steps in the process chain are extended to business process architecture elements to allow, for instance, a process chain to be composed of sequentially ordered ‘sub’ process chains or process families (as represented by their standard processes). Using our meta-model, end-to-end processes can be modelled as process chains.
- ***Process Group*** is an aggregate of business process architecture elements that become members of the same process group as defined by *grouping criteria* (Req. 3c). The members of a process group are related in the sense that they share one or more similar properties. They can, for instance, belong to a same process category, in which case the process group represents a category and the property of fulfilling a similar role in the organization or serving a similar

purpose is used as grouping criterion (see Sub-Section 4.5). The process group may also represent a phase meaning that its member processes are executed at the same time and time of execution is the property that serves as grouping criterion. Also, being related to a same enterprise architecture element is a property that can be used as grouping criterion to define a process group, for instance, all processes having the same business actor as process owner form a process group.

- ***Process Family*** is an aggregate of business process architecture elements based on *specialization* relations (Req. 3d) where a parent process assumes the role of standard process (called main process in [18]) and child processes are variants of the standard process. The specialization relation is defined for the abstract business process architecture elements to allow for maximum freedom when instantiating the meta-model (e.g., one process group specializing another process group, one process chain being a variant of another process chain). We follow Van Nuffel and De Backer [18] by having the process family represented through the standard process, implying that a standard process not only generalizes process variants but also aggregates these variants, i.e., the standard process is not an elementary business process but a business process composite. The meta-model does, however, not impose that all parent processes are standard processes that represent process families.

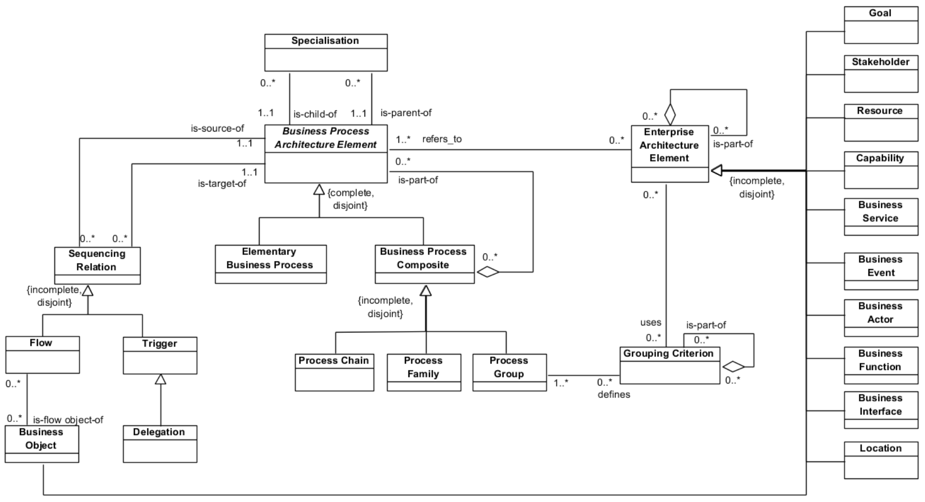


**Fig. 2.** Meta-model for business process architecture description.

The recognition of different types of business process composite implies that business process architecture elements can be related in different ways. The ***Sequencing Relation*** (Req. 3a) associates a source element to a target element implying a temporal ordering of *source* and *target*. The source and target association ends represent the roles that business process architecture elements fulfil in sequencing relations. In case of sequentially related elementary business processes, the source and target roles are equivalent to the prior and subsequent roles identified in our process map conceptualization [30]. The semantics of the temporal ordering of source and target are defined more precisely in the subclasses of the sequencing relation, yet the specialization is optional to allow for maximum freedom when instantiating the meta-model to describe business process architectures and create process maps. For instance, Dijkman et al. [1] present a process map example where processes are shown as temporally ordered without further specification of the exact nature of the sequential ordering (e.g., Should the prior process be completed before the subsequent process can start? Does the prior process passes information on to the subsequent process?). Scheer's Value Added Diagram [20] has been mentioned in related literature as a framework for designing process maps [41]. The sequential order of the prior and subsequent processes composing the process chain represented on such diagram might be indicated purely by means of secondary notation (i.e., using the chevron symbol for processes and placing sequentially ordered processes adjacently on the map). The only semantics attached to the sequential relation concept in our meta-model is that the business process architecture element that is the target of the relation follows up on something performed by the business process architecture element that is the source of the relation. 'Follows up on' means that it may depend on the type of sequential relation (if further specified) but also on the type of business process architecture elements that are sequentially related. Hence, when instantiating the meta-model, additional constraints might be imposed by the users (e.g., a constraint that process groups cannot be sequentially related). Two subclasses of sequential relation are specified in the meta-model: ***Trigger*** and ***Flow***, which we define based on the formalization in [42]. A Trigger is a sequencing relation in which the source business process architecture element causes the target business process architecture element to be instantiated and to start. Instantiation of business process composites is not further defined as its relevancy and semantics may depend on the chosen business process architecture viewpoint. For instance, the instantiation of a process chain could mean the instantiation of its first process step. For a process family, it could mean the instantiation of any variant of the standard process. For process groups, it probably has no relevancy. We further distinguish delegation as a subclass of trigger. A ***Delegation*** is a trigger in which the source is dependent on the outcome of the target. It is similar to the dependency relation in [18] and the uses relation in [1]. While with trigger it is not required that the source expects a response from the target (and so can end independently of the target), with delegation a response is expected and the source will not end before the target has performed some work whose outcome is needed to successfully complete the source. A ***Flow*** is a sequencing relation in which a business object flows from the source to the target. We define a ***Business Object*** as anything that flows between business process architecture elements and which is considered relevant according to some business process architecture viewpoint to be represented in a process map. This can be information, as in the meta-model of Malinova and Mendling [15], but also physical products or even persons (e.g., a patient). While conceptually every flow has at least one business object as flow object, the process map as an abstraction does not need to show this flow object. On the other hand, a business object can only be a flow object on a process map if it is linked to some flow. As shown in our meta-model, a business object is an enterprise architecture element. The meta-model allows flows between elementary business processes, but also between business process composites. More restrictive rules can be imposed if flows between certain kinds of composites are not meaningful, but for the sake of generality such constraints are not part of the meta-model. Trigger (or delegation) and flow can occur concurrently between a same source and target. Triggering is usually also accompanied by the passing of some information or signal. If meaningful to be shown on the process map, then both relations can be depicted, where the trigger has the meaning of starting the target and the flow has the meaning of passing a business object to the target.

Apart from decomposition and sequencing relations, the meta-model includes ***Specialisation*** which relates a child business process architecture element to a parent business process architecture element (Req. 3d). A parent can have many children while a child can have many parents. The meta-model allows business process composites to specialize other business process composites (e.g., a process group that specializes another process group), but again specialization of business process composites is not further defined as its relevancy and semantics depends on business process architecture viewpoint.

A special kind of relation is that between members of a process group. They are related in the sense that they share a common property. The property on the basis of which the grouping occurs is defined by a ***Grouping Criterion*** (Req. 3c). An example of a grouping criterion is ‘the process category is core’. This criterion then groups all the core processes of the organization. A grouping criterion can aggregate other grouping criteria, for instance ‘the process category is management and the location of process execution is the company headquarters’. Each of these aggregated criteria defines its own process group, while the aggregate criterion defines the intersection of these groups as a new process group. The type of aggregation shown in the meta-model is shared aggregation, allowing any kind of regular expression of logical operators to compose grouping criteria based on elementary criteria, again allowing maximum freedom when instantiating the meta-model. The meta-model shows that each grouping criterion defines at least one process group (possibly empty), but there can be many groups defined by the same criterion to account for evolution over time (i.e., the set of business process architecture elements that share some property is dynamic). The meta-model also allows that a process map shows process groups without identifying the defining grouping criteria.

The final element on the meta-model is that of ***Enterprise Architecture Element***, which is defined as an element that is part of the enterprise architecture and that is related to a business process architecture element (Req. 2). As discussed in Sub-Section 4.5, the reviewed literature offers a non-exhaustive set of concepts that can be shown in a process map as they relate to business processes or their composites. We believe that most of these can be described as enterprise architecture elements, though it might depend on the EA framework referred to whether they are recognized as such.

The meta-model in Figure 2 shows a number of subclasses of Enterprise Architecture Element. These subclasses are not an exhaustive enumeration of relevant types of enterprise architecture elements that can be included in process maps (as the specialization of Enterprise Architecture Element is partial). They are included in the meta-model for illustrative purposes only, being inspired by the concepts included in the motivation, strategy and business layers of the ArchiMate ADL [13]. We could also add concepts of the application, technology and physical layers of the ArchiMate ADL (e.g., application components, data objects), but we chose not to do so in order not to overload Figure 2. We consider a relationship with an enterprise architecture element as a property of the business process architecture element. Consequently, a primary use of enterprise architecture elements in a process map is to define process groups. Hence, enterprise architecture elements can be used by grouping criteria to define process groups. Enterprise architecture elements can aggregate other enterprise architecture elements. For instance, processes of an international company can be grouped by continent and by country, where a continent is a location aggregating countries as other locations.

The main feature of the meta-model is that through the use of the concept Enterprise Architecture Element, the business process architecture is integrated into the overall enterprise architecture and hence the process map, as a business process architecture model, can be linked to EA models.

## 5. Testing the Metamodel

To demonstrate and evaluate the meta-model's ability to serve as a conceptual foundation of a universal representation language for process maps, we instantiated it to represent a wide array of example process maps found in the literature. This was a repetitive process providing us with feedback on how to refine the meta-model until all our instantiations were valid. To this end we used three different published classifications of business process architecture description. Sub-Section 5.1 presents meta-model instantiations for the four 'archetypes' of process map used in industry after an extensive empirical study of how process mapping is performed in practice conducted in [4]. Sub-Section 5.2 shows how the meta-model can be instantiated for the example process maps described in [1] based on a systematic literature review of business process architecture design approaches. Next, Sub-Section 5.3 describes how the illustrative process maps that are part of different business process architecture views proposed by Van Nuffel and De Backer [18] can be represented using the meta-model that we designed. We show the instantiation of the meta-model by means of concept maps. To clarify the link between the concept map and the meta-model, concepts are stereotyped with the name of the meta-model class that is instantiated. The relationships between the concepts show the links between these class instances according to the associations and aggregation relationships of the meta-model. The meta-model evaluation against core requirements for domain-specific languages is presented in Section 5.4.

### 5.1. Representation of Archetypical Process Maps Used in Industry

With regard to the representation of archetypical process maps, in Malinova, Leopold [4] four types are distinguished. Table 2 shows these archetypical process maps (left column) together with their concept map representation using the proposed meta-model (right column). The model shown in the left column of the first row represents two adjacent modelling layers in a *hierarchical process model architecture*. Note that the process models in the layers labelled level 2 and 3 are not black-box models of business processes, showing for instance sequence flows and gateways for non-sequential flow. Following Req. 1, representing this model using our meta-model means abstracting from the internal details of these processes. The corresponding concept map thus shows Level 2 Process as a business process composite that has the two Level 3 Process as parts. These Level 3 Process are modelled as elementary business processes. The second row shows a process map of the *pipeline process architecture* archetype. This process map can be represented without compromise by instantiating the proposed meta-model. At each level a process chain is shown that is decomposed into sequentially related 'sub' process chains. The concept map on the right shows the mechanism of this recursive structure of sequentially related process chains. Although not shown in the concept map, for level 4 the business process architecture elements modelled would be elementary business processes. The third row contains the example process map of the *divisional process architecture* archetype. The process map distinguishes between management processes and core processes for which (business) units are responsible. The concept map at the right demonstrates the use of the grouping mechanism. The grouping of management processes is defined by the process category (i.e., management). We chose to explicitly model the grouping criterion, but the grouping can also be implicit and just based on the name of the process group. Core processes are grouped by means of a common property, i.e., their relation to a business unit. Such business units can be represented in an EA model as business actors (e.g., using ArchiMate). The reference to a common business actor thus defines the grouping of the core processes, which is only illustrated for Unit II in the concept map at the right. The plus signs in the process map at the left indicate that each core process is actually a business process composite, which can be further decomposed and specialized if needed, for instance as process chains like in the pipeline process architecture of the second row. Finally, the fourth row shows the example process map of the *service-oriented process architecture* archetype. The process map includes four distinct groups of processes, which we model as process groups, the criterion for grouping being the process category (i.e., management, service, support, measure & analyse). Each process in these groups is modelled as a business process composite (because of the plus sign in the process map). The concept map shows the mechanism of delegation that is exemplified in the process map. A business process (composite) in the service process group delegates part of the work to be performed to processes of other groups. In the example an elementary business process that is part of some business process composite of the support group and an elementary business process that is part of some business process composite that is part of the measure & analyse process group. Note that similar to what is seen in the first row, the right part of the process map (left column) is not a black-box model, hence the instantiation of the meta-model (right column) does not show gateways and sequence flows.

## 5.2. Representation of Process Maps Resulting from Different Business Process Architecture Design Approaches

Dijkman et al. [1] identified five different approaches for designing business process architectures in the literature. These approaches stand for the academic perspective on business process architecture. Each approach is exemplified by a different process map in an informal notation that is inspired by ArchiMate. Table 3 shows these example process maps (left column) along with their representation as instantiation of the meta-model proposed in Section 4 (right column). The example process maps in the left column of Table 3 differ from those of Table 2 in that they only show business process architecture elements and relationships between those, i.e., no process-related EA elements are shown. All the elements and relationships included in the example process maps can be represented by instantiating the proposed meta-model, in particular by means of business process composites, elementary business processes and is-part-of relationships. In the second row a process chain (Perform Project) is shown that consists of two elementary business processes (Make Project Plan and Approve Project Plan) that are related by a Trigger relation (i.e., Make Project Plan triggers Approve Project Plan). In the third row a process family is shown, which is represented by a standard process (Insurance Application) that generalizes two elementary business processes (Home Insurance Application and Car Insurance Application) that are part of the process family. An alternative representation is to show both processes as children of the parent process Insurance Application, however, the chosen representation emphasizes that Insurance Application represents an entire process family.

**Table 2.** Meta-model instantiations for archetypical industry process maps.

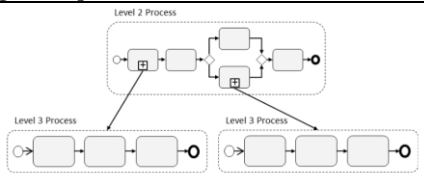
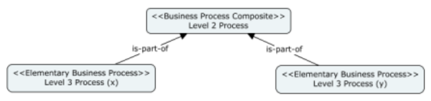
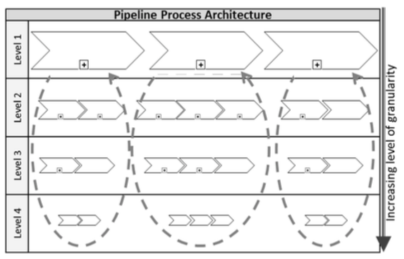
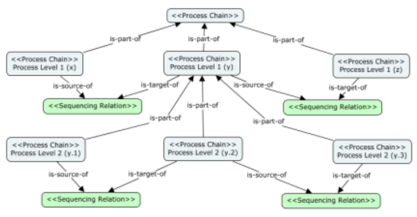
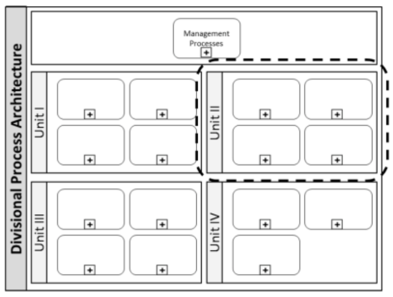
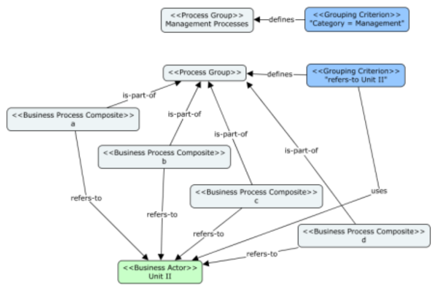
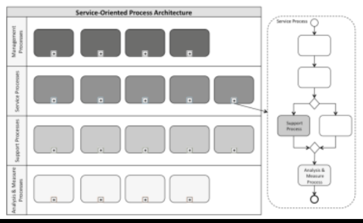
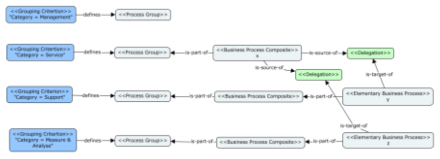
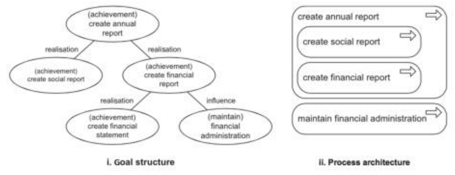
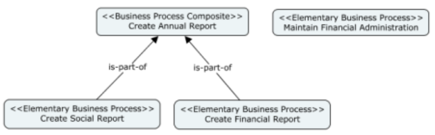
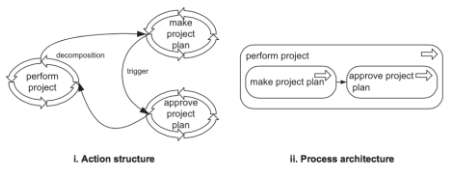
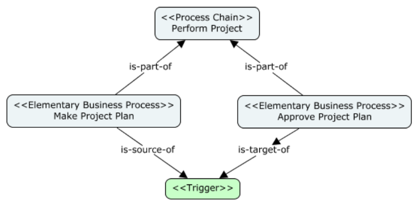
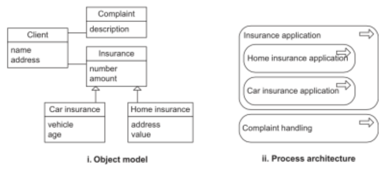
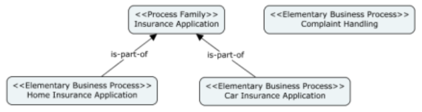
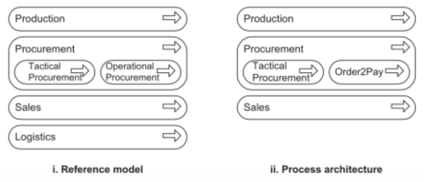
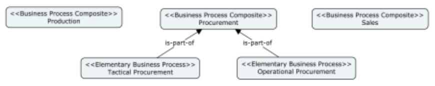
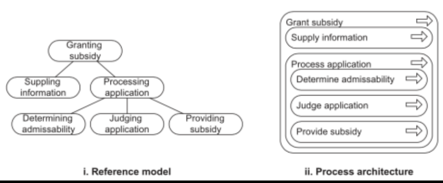
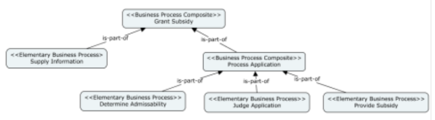
| Archetypical industry process map – original representation | Archetypical industry process map – meta-model instantiation as concept map |
|---|---|
|  |  |
|  |  |
|  |  |
|  |  |

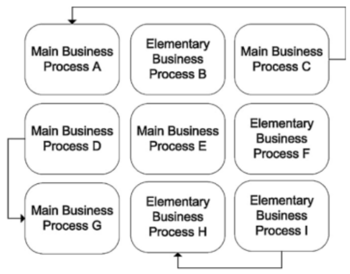
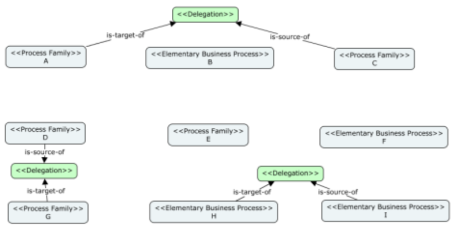
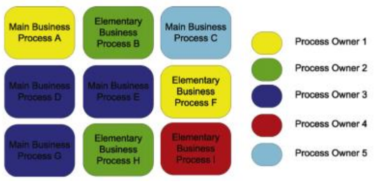
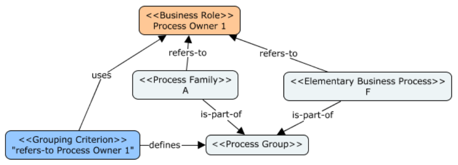
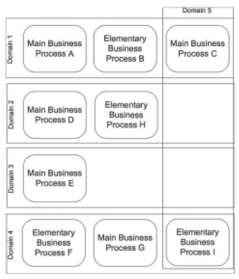
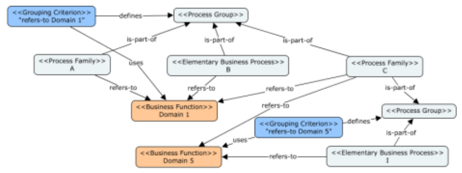
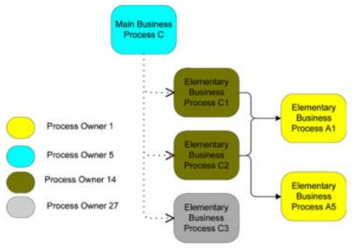
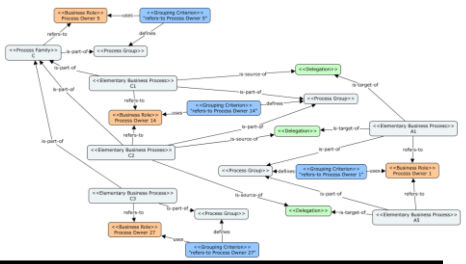
**Table 3.** Meta-model instantiations of business process architecture design approaches.

| Business process architecture design approach – original representation of resulting process map | Business process architecture design approach – meta-model instantiation as concept map |
|---|---|
|  (achievement) create annual report<br>realisation<br>realisation<br>(achievement) create social report<br>(achievement) create financial report<br>realisation<br>influence<br>(achievement) create financial statement<br>(maintain) financial administration<br>i. Goal structure<br>create annual report<br>create social report<br>create financial report<br>maintain financial administration<br>ii. Process architecture |  <<Business Process Composite>> Create Annual Report<br><<Elementary Business Process>> Maintain Financial Administration<br>is-part-of<br>is-part-of<br><<Elementary Business Process>> Create Social Report<br><<Elementary Business Process>> Create Financial Report |
|  make project plan<br>decomposition<br>trigger<br>perform project<br>approve project plan<br>i. Action structure<br>perform project<br>make project plan<br>approve project plan<br>ii. Process architecture |  <<Process Chain>> Perform Project<br>is-part-of<br>is-part-of<br><<Elementary Business Process>> Make Project Plan<br><<Elementary Business Process>> Approve Project Plan<br>is-source-of<br>is-target-of<br><<Trigger>> |
|  Complaint<br>description<br>Client<br>name<br>address<br>Insurance<br>number<br>amount<br>Car insurance<br>vehicle<br>age<br>Home insurance<br>address<br>value<br>i. Object model<br>Insurance application<br>Home insurance application<br>Car insurance application<br>Complaint handling<br>ii. Process architecture |  <<Process Family>> Insurance Application<br><<Elementary Business Process>> Complaint Handling<br>is-part-of<br>is-part-of<br><<Elementary Business Process>> Home Insurance Application<br><<Elementary Business Process>> Car Insurance Application |
|  Production<br>Procurement<br>Tactical Procurement<br>Operational Procurement<br>Sales<br>Logistics<br>i. Reference model<br>Production<br>Procurement<br>Tactical Procurement<br>Order2Pay<br>Sales<br>ii. Process architecture |  <<Business Process Composite>> Production<br><<Business Process Composite>> Procurement<br><<Business Process Composite>> Sales<br>is-part-of<br>is-part-of<br><<Elementary Business Process>> Tactical Procurement<br><<Elementary Business Process>> Operational Procurement |
|  Granting subsidy<br>Suppling information<br>Processing application<br>Determining admissability<br>Judging application<br>Providing subsidy<br>i. Reference model<br>Grant subsidy<br>Supply information<br>Process application<br>Determine admissability<br>Judge application<br>Provide subsidy<br>ii. Process architecture |  <<Business Process Composite>> Grant Subsidy<br>is-part-of<br>is-part-of<br><<Elementary Business Process> Supply Information<br><<Business Process Composite>> Process Application<br>is-part-of<br>is-part-of<br>is-part-of<br><<Elementary Business Process>> Determine Admissability<br><<Elementary Business Process>> Judge Application<br><<Elementary Business Process>> Provide Subsidy |

## 5.3. Representation of Process Maps that Model Different Business Process Architecture Views

Van Nuffel and De Backer [18] propose six views of business process architecture of which they illustrate three with an example process map in an informal self-crafted notation. Table 4 shows these examples (left column) and how they can be represented using the concepts of the proposed meta-model (right column). The fourth row in the table presents another example taken from [18] which is positioned at a lower level in their business process decomposition structure to show the process variants making up a process family. The first row contains a model that is part of a view that shows main business processes, elementary business processes and dependency relationships between processes. Using the proposed meta-model, these business process architecture elements and relationships are modelled as process families, elementary business processes and delegation relationships, as shown in the concept map (right column).

**Table 4.** Representation of process maps that model different business process architecture views.

| Business process architecture view – original representation | Business process architecture view – meta-model instantiation as concept map |
|---|---|
|  |  |
|  |  |
|  |  |
|  |  |

According to [18], a dependency relationship means that a ‘depender’ process depends on the result of a ‘dependee’ process to perform its task, which corresponds semantically to the delegation relation of our meta-model. The depender is the source of the relation, while the dependee is the target of the relation. The model in the second row uses colour coding to identify the process owner of each main and elementary business process in the process architecture. The concept map demonstrates the grouping mechanism by grouping process families and elementary business process owner that refer to the same process owner, which is modelled as a business role (i.e., specialization of the EA element class of the meta-model). The view portrayed in the third row shows that main and/or elementary business processes can belong to more than one process group. The concept map has process family C referring to functional domains 1 and 5 (modelled as business functions, which specialize the EA element class of the meta-model). Hence, process family C is part of two process groups. Finally, in the fourth row the model shows the process variants (C1, C2 and C3) of a main business process (C). The concept map on the right shows the full model where elementary business processes are part of process families and both kinds of business process architecture elements are further grouped based on common reference to a process owner, modelled as business role like in row two. Process dependencies are modelled using the delegation relation of the meta-model.

## 5.4. Evaluation of the Meta-Model

As demonstrated in Sub-Sections 5.1 to 5.3, a large variety of notations is used to articulate process maps, regardless whether they originate in practice or in academia. The metamodel was demonstrated to model this variety of situations along with the fulfilling of the stated solution requirements. Furthermore, the meta-model has been evaluated in order to have a first insight about its validity, by considering the core requirements for a domain-specific language (DSL) [31], given that the proposed meta-model is intended to serve as the domain definition meta-model of a potential DSL for process mapping. The meta-model has been tested against these core requirements [31], such as summarized in Table 5.

**Table 5.** Evaluation of the metamodel

| DSL req. | Justification |
|---|---|
| Conformity | The following general concepts were considered: (i) architecture; (ii) organizational context with regards to business processes; (iii) structure and relations between business processes.<br>Elements at the business level in EA are included (in accordance to the relevant related literature). **(Req. 2)**<br>The meta-model has been applied to represent a wide array of process maps and other business process architecture descriptions found in the literature (see Sub-Sections 5.1, 5.2 and 5.3). |
| Orthogonality | Each construct in the language was conceived to represent exactly one distinct concept in the domain. |
| Supportability | The suitability of enhancing EA modelling languages, in particular ArchiMate, will be considered, as we conceptualize the process map as an EA artefact along with the formal meta-model for a business process architecture description language. |
| Integrability | Conceptualization of the process map is driven as a business process abstraction that provides a black-box view on the organization's business processes **(Req. 1)**. This can easily be integrated with the |

| DSL req. | Justification |
|---|---|
| | white-box perspective, supported by BPMN.<br>A maximal integration with enterprise architecture description is obtained through the alignment with the ISO/IES/IEEE 42010 standard for architecture description.<br>Different kinds of business process composites are considered. **(Req. 3)**<br>Maximum freedom for instantiating the meta-model according to process mapping needs is provided, thereby limiting the number of constraints at the meta-model level. |
| Longevity | The meta-model was built to be aligned with the relevant standards about enterprise architecture and in the particular domain of business process architecture it generalizes existing proposals to reflect the mostly agreed upon concepts.<br>The usage of generalized concepts (Enterprise Architecture Element, Business Process Architecture Element, etc.) facilitates the extension of the meta-model to be adapted to future situations. |
| Simplicity | A set of minimal constructs and constraints were considered<br>Some complex mechanisms in process maps (e.g. aggregation) were simplified by applying design patterns (e.g. composition). |

## 6. Conclusion and Future Work

In this paper, a meta-model of a business process architecture description language for representing process maps was presented, based on a process map conceptualization in the context of enterprise architecture. With the aim of testing the meta-model's ability to serve as a foundation of a universal representation language for process maps, a wide sample of process maps were instantiated and it has been shown how the meta-model fulfils core requirements for a domain-specific language.

The contribution of this research is the creation through our meta-model of a better understanding of business process architecture in relation to enterprise architecture, which could promote major advancements in the field and can be beneficial for both BPM and EA professionals and enterprise modelling in general. As a process map is a model of the business process architecture, it is complementary to any model that describes a view of the enterprise architecture (e.g., a goal model, a capability map, an application landscape, an infrastructure landscape). The main novelty in our meta-model, compared to the related work and apart from the generic Business Process Composite concept obtained through applying the composition pattern, was the introduction of the Enterprise Architecture Element as a 'placeholder' for any kind of element in any kind of enterprise architecture model that is related to a Business Process Architecture Element (e.g., goals or capabilities realized by elements of the business process architecture, elements of the business process architecture served by applications that are hosted on IT infrastructure nodes). This way a process map can be effectively integrated into the overall enterprise architecture model of an organization.
Whether this integration is easy to perform is another issue, that will depend on the choice of languages for the different architecture models. For instance, models

expressed in ArchiMate do not allow users to zoom in on the details of specific types of business processes and relationships between them. For this reason, our proposal was designed to overcome this limitation by proposing a new type of model (i.e., the process map as a model of business process architecture) that is aligned with industrial EA standards such as TOGAF and ArchiMate.

The main future work will be the design of a concrete syntax for the meta-model, which considers the suitability of EA modelling languages as a basis for defining a domain-specific language for process mapping. A software tool will be developed and empirical studies will be conducted to test the usability and usefulness of the proposed metamodel and syntax. The resulting feedback can serve to improve the metamodel in a new research cycle by following DSR. In terms of tooling, our proposal will seek to support the navigation between different EA and BPM models. For example, the user could be viewing a model in ArchiMate and by clicking on a business process element, the software would show in another window the detailed process model with the workflow (BPMN). As a result, new possibilities can arise to harmonize the use of general-purpose modelling languages from both fields (e.g., BPMN and ArchiMate).

**Acknowledgements.** This work is partially supported by: **BIZDEVOPS-Global** (ref. RTI2018-098309-B-C31), "Ministerio de Economía, Industria y Competitividad" (MINECO) & "Fondo Europeo de Desarrollo Regional" (FEDER); and **G3Soft** (SBPLY/17/180501/000150), "Consejería de Educación y Ciencia, Junta de Comunidades de Castilla-La Mancha" and FEDER.

## References

1. Dijkman, R., I. Vanderfeesten, and H.A. Reijers, *Business process architectures: overview, comparison and framework.* Enterprise Information Systems, 2016. **10**(2): p. 129-158.
2. Dumas, M., et al., *Fundamentals of Business Process Management*. 2013, Berlin Heidelberg: Springer-Verlag.
3. Soffer, P. and Y. Wand, *Goal-driven multi-process analysis.* Journal of the Association for Information Systems, 2007. **8**(3): p. 175-203.
4. Malinova, M., H. Leopold, and J. Mendling, *An Empirical Investigation on the Design of Process Architectures*, in *11th International Conference on Wirtschaftsinformatik*. 2013: Leipzig.
5. Rosemann, M. *Process Portfolio Management*. BPTrends, 2006.
6. Heinrich, B., et al., *The process map as an instrument to standardize processes: design and application at a financial service provider.* Information Systems and E-Business Management, 2009. **7**(1): p. 81-102.
7. Lapouchian, A., E. Yu, and A. Sturm, *Design Dimensions for Business Process Architecture*, in *ER 2015, Lecture Notes in Computer Science*, P. Johannesson, Editor. 2015a, Springer International Publishing. p. 276-284.
8. Lapouchnian, A., E. Yu, and A. Sturm, *Re-Designing Process Architectures: Towards a Framework of Design Dimensions*, in *7th IEEE International Conference on Research Challenges in Information Science*. 2015b: Athens.
9. Odeh, Y., M. Odeh, and S. Green, *Aligning Riva-based Business Process Architectures with Business Goals Using the i* Framework*, in *3rd International Conference on Business Intelligence and Technology*. 2013: Valencia.
10. Poels, G., et al., *Investigating Goal-Oriented Requirements Engineering for Business Processes.* Journal of Database Management, 2013. **24**(2): p. 35-71.
11. Engelsman, W., et al., *Extending enterprise architecture modelling with business goals and*

*requirements.* Enterprise Information Systems, 2011. **5**(1): p. 9-36.
12. Nogueira, J., et al., *Leveraging the Zachman Framework Implementation Using Action – Research Methodology – A Case Study: Aligning the Enterprise Architecture and the Business Goals.* Enterprise Information Systems, 2013. **7**(1): p. 100-132.
13. The Open Group, *Open Group Standard. ArchiMate 3.0 Specification*. 2016.
14. Wierda, G., *Mastering ArchiMate. 2nd ed*. 2014.
15. Malinova, M. and J. Mendling, *Why is BPMN not Appropriate for Process Maps?*, in *36th International Conference on Information Systems*. 2015a: Fort Worth.
16. Malinova, M., H. Leopold, and J. Mendling, *An Explorative Study of Process Map Design*, in *CAiSE Forum 2014, Lecture Notes in Business Information Processing*, S. Nurcan and E. Pimenidis, Editors. 2015, Springer International Publishing.
17. Bider, I., et al., *A fractal enterprise model and its application for business development.* Software & Systems Modeling, 2016.
18. Van Nuffel, D. and M. De Backer, *Multi-abstraction layered business process modeling.* Computers in Industry, 2012. **63**(2): p. 131-147.
19. Rolland, C., S. Nurcan, and G. Grosz, *Enterprise knowledge development: the process view.* Information & Management, 1999. **36**(3): p. 165-184.
20. Scheer, A., *ARIS: Business Process Modeling*. 2000, Berlin Heidelberg: Springer-Verlag.
21. Francesconi, F., F. Dalpiaz, and J. Mylopoulos, *TBIM: A Language for Modeling and Reasoning about Business Plans*, in *ER 2013, Lecture Notes in Computer Science*, W. Ng, V. Storey, and J. Trujillo, Editors. 2013, Springer International Publishing. p. 33-46.
22. Frank, U., *Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges.* Software and Systems Modeling, 2014. **13**(3): p. 941-962.
23. Sandkuhl, K., et al., *Enterprise Modeling. Tackling Business Challenges with the 4EM Method*. 2014, Berlin Heidelberg: Springer-Verlag.
24. Roelens, B., W. Steenacker, and G. Poels, *Realizing strategic fit within the business architecture: the design of a Process-Goal Alignment modeling and analysis technique.* Software & Systems Modeling, 2017.
25. von Rosing, M., et al., *Combining BPM and EA in Complex IT Projects: A Business Architecture Discipline*, in *IEEE 13th Conference on Commerce and Enterprise Computing*. 2011. p. 271-278.
26. Hevner, A.R., et al., *Design science in Information Systems research.* Mis Quarterly, 2004. **28**(1): p. 75-105.
27. Peffers, K., et al., *A design science research methodology for Information Systems Research.* Journal of Management Information Systems, 2007. **24**(3): p. 45-77.
28. Wieringa, R., *Design Science as Nested Problem Solving*, in *4th International Conference on Design Science Research in Information Systems and Technology*. 2009: Philadelphia.
29. ISO/IEC/IEEE, *International Standard 42010. Systems and Software Engineering – Architecture Description*. 2011.
30. Poels, G., et al., *Conceptualizing Business Process Maps*. 2018: CoRR abs/1812.05395.
31. Kolovos, D., et al., *Requirements for Domain-Specific Languages*, in *1st ECOOP Workshop on Domain-Specific Program Development*. 2006: Nantes.
32. The Open Group, *Open Group Standard. TOGAF 9.1 Specification*. 2009.
33. Malinova, M., *A Language for Process Map Design*, in *BPM 2014 Workshops, Lecture Notes in Business Information Processing*, F. Fournier and J. Mendling, Editors. 2015, Springer International Publishing. p. 567-572.
34. Milani, F., et al., *Criteria and Heuristics for Business Process Model Decomposition Review and Comparative Evaluation.* Business & Information Systems Engineering, 2016. **58**(1): p. 7-17.
35. Aßmann, U., S. Zschaler, and G. Wagner, *Ontologies, Meta-models, and the Model-Driven Paradigm*, in *Ontologies for Software Engineering and Software Technology*, C. Calero, F. Ruiz, and M. Piattini, Editors. 2006, Springer-Verlag. p. 249-273.
36. Saeki, M. and H. Kaiya, *On Relationships Among Models, Meta Models, and Ontologies*, in

*6th OOPSLA Workshop on Domain-Specific Modeling*. 2006: Portland.

37. La Rosa, M., M. Dumas, and A. ter Hofstede, *Modeling Business Process Variability for Design-Time Configuration*, in *Handbook of Research in Business Process Modeling*, J. Cardoso and W. van der Aalst, Editors. 2009, IGI Global. p. 204-228.
38. Rosemann, M. and J. vom Brocke, *The Six Core Elements of Business Process Management*, in *Handbook on Business Process Management 1*, J. vom Brocke and M. Rosemann, Editors. 2015, Springer-Verlag: Berlin Heidelberg. p. 105-122.
39. Malinova, M., B. Hribar, and J. Mendling, *A Framework for Assessing BPM Success*, in *22nd European Conference on Information Systems*. 2014: Tel Aviv.
40. Gamma, E., et al., *Design Patterns: Elements of Reusable Object-Oriented Software*. 1995, Boston: Addison-Wesley Longman Publishing.
41. Malinova, M. and J. Mendling, *Leveraging Innovation Based on Effective Process Map Design: Insights from the Case of a European Insurance Company*, in *BPM – Driving Innovation in a Digital World*, J. vom Brocke and T. Schmiedel, Editors. 2015b, Springer International Publishing: Bern. p. 215-227.
42. Eid-Sabbagh, R.-H., R. Dijkman, and M. Weske, *Business Process Architecture: Use and Correctness*, in *BPM 2012, Lecture Notes in Computer Science*, A. Barros, A. Gal, and E. Kindler, Editors. 2012, Springer International Publishing. p. 65-81.

**Geert Poels** is head of the Management Information Systems research group at Ghent University (Belgium). He is full professor at the Faculty of Economics and Business Administration, Ghent University where he teaches Computer Science, Information Systems and Management of Information Technology subjects to Business Engineering and Business Administration students. He is member of the Ghent University Research Council. He also teaches, consults and directs master dissertation research at the IC Institute, which is part of the INNOCOM company (Beersel, Belgium). His areas of research are business process management, enterprise modelling, business ontology, digital transformation, agile requirements engineering, and IT governance (as co-developer of COBIT 2019). Mid 2019, he is promoter of 13 completed PhD research projects (11 at UGent and 2 at KU Leuven) and has 118 publications listed in Web of Science. His Google Scholar h-index is 30 with over 3000 citations recorded.

**Félix García** is Full Professor at the University of Castilla La-Mancha (UCLM), where he received his MSc (2001) and PhD (2004) degrees in Computer Science. He is member of the Alarcos Research Group and his research interests include business process management, software processes, software measurement and agile methods. http://orcid.org/0000-0001-6460-0353.

**Francisco Ruiz** is a full professor at the University of Castilla-La Mancha (UCLM), Spain. His current interests include enterprise architecture, business process management, information systems, and socio-demographic data analysis. Ruiz received a Master in chemistry-physics from Complutense University of Madrid, and a PhD in computer science from UCLM. Contact him at francisco.ruizg@uclm.es.

**Mario Piattini** is the director of the Alarcos Research Group and a full professor at the University of Castilla-La Mancha, Spain. His research interests include software and data quality, information-systems audit and security, and IT governance. Piattini received a Ph.D. in computer science from Madrid Technical University, Spain.

*Received: November 18, 2018; Accepted: July 30, 2019*

Computer Science and Information Systems 17(1):141–160 https://doi.org/10.2298/CSIS181115017M

# Correctness of the Chord Protocol

Bojan Marinković[1], Zoran Ognjanović[1], Paola Glavan[2], Anton Kos[3], and Anton Umek[3]

[1] Mathematical Institute of the Serbian Academy of Sciences and Arts
Beograd, Serbia
[bojanm,zorano]@mi.sanu.ac.rs

[2] Faculty of Mechanical Engineering and Naval Architecture
University of Zagreb, Zagreb, Croatia
pglavan@fsb.hr

[3] Faculty of Electrical Engineering
University of Ljubljana, Ljubljana, Slovenia
[anton.kos, anton.umek]@fe.uni-lj.si

**Abstract.** Internet of Things (IoT) can be seen as a cooperation of various devices with limited performances that participate in the same system. IoT devices compose a distributed architecture system. The core of every IoT system is its discovery and control services. To realize such services, some authors used the developed solutions from the different domains. One such solution is the Chord protocol, one of the first, the simplest and the most popular distributed protocols. Unfortunately, the application of the Chord protocol was realized using the correctness of the Chord protocol for granted, or by the very hard assumptions. In this paper we prove the correctness of the Chord protocol using the logic of time and knowledge with the respect to the set of possible executions, called regular runs. We provide the deterministic description of the correctness of the Chord protocol and consider Chord actions that maintain ring topology while the nodes can freely join or leave.

**Keywords:** IoT, DHT, Chord, correctness, temporal logic, epistemic logic

## 1. Introduction

Internet of Things (IoT) paradigm can be defined as: "The pervasive presence around us of a variety of things or objects which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals" [1].

In this framework the smart objects, which are connected by a network structure, are able to communicate and exchange information and to enable new forms of interaction among things and people [2]. The core of every IoT system consists of its discovery and control services. It is common that various homogeneous and heterogeneous devices participate in the same IoT system. Usually, these devices are highly distributed, therefore it can be seen as they participate in a distributed Peer-to-Peer (P2P) system.

In a case of homogeneous decentralized distributed P2P system, nodes (peers) run the same application, and share the same properties in terms of computation and storage capacities and network connectivity [3]. Without any centralized control processes are dynamically distributed to nodes that can join or leave the system at any time. Thus, P2P systems are highly scalable, as they have no inherent bottlenecks. Also, such systems are resilient to failures, attacks, etc., since there is no single node or a group of nodes that implement a critical functionality, which would render the system unusable, if disrupted. P2P systems are used for file sharing, redundant storage, and real-time media streaming.

While the underlying network actually connects devices, P2P systems are frequently implemented in a form of overlay networks, structures which organize system resources in an independent logical topology [4]. Overlay networks can be realized in the form of Distributed Hash Tables (DHTs). A DHT provides a lookup service similar to a hash table. Pairs of the form $\langle key, value\rangle$ are stored in a DHT, while nodes participating in the network can efficiently retrieve the value associated with a given key. The functionality of maintaining the mapping from keys to values is implemented by nodes in a distributed manner and changes in the set of participating nodes cause only minimal amount of disruption.

The Chord protocol [5,6,7] is one of the first, the simplest and the most popular implementations of DHTs. Nodes in a network executing the Chord protocol are organized in a ring. Because of the simplicity and popularity of the Chord protocol, it was used for the realization of the discovery and/or control service of IoT systems described in [2,8,9,10,11]. Although Chord is one of the simplest protocols, since it is distributed, it is subject to faults and bugs, so the protocol verification is extremely important.

In this paper we use a temporal epistemic logic to prove the correctness of the Chord protocol with respect to the Chord actions that maintain ring topology. We consider the case when the nodes are allowed to leave or join the network with respect to the set of possible executions, called regular runs. It means that a network executing a regular run will be always brought from an arbitrary state to a state in which links between nodes form a ring.

Up to our knowledge there were of only a few papers related to formal verification of DHTs, and particularly Chord [12,13,14,15,16]. They are considered them in Section 2 and compared with our approach.

The rest of the paper is organized in the following way: in Section 2 we consider other approaches for proving the correctness of the Chord protocol and clearly present the contributions of this paper; Section 3 presents a short description of the Chord protocol; in Section 4 we present a logical framework which is used to prove the correctness of the maintenance of the ring topology of the Chord protocol with the respect to the regular runs; the proof is given in Section 5; we conclude with Section 6. In Appendix A we provide detailed proofs of the main lemmas and theorems from the paper.

## 2. Related Work and Contributions

### 2.1. Related Work

The Chord protocol is introduced in [5,6,7]. The detailed examination of the protocol's performance and robustness is given under the assumption that nodes and keys are randomly chosen. The provided approach is probabilistic, e.g., statements are of the form "With high probability, the number of nodes that must be contacted to find a successor in a $N$-node network is $O(\log N)$".

The only deterministic given result is [5,6,7, Theorem IV.3] which partially corresponds to our Lemma 6. Theorem IV.3 proves that inconsistent states produced by executing several concurrent joins of the new nodes are transient, i.e., that after the last node joins the network will form a cycle. A more general sequences of concurrent joining and leaving of nodes is presented in [15]. The paper gives a lower bound of the rate at which nodes need to maintain the system such that it works correctly with high probability. In this paper we consider the case when the nodes are allowed to leave the network with respect to the set of possible executions, called regular runs.

While it is hard to directly compare these two approaches (deterministic and probabilistic), they are complementary and may be used together to improve our understanding of the Chord protocol. One can argue that the probabilistic approach is useful to study robustness of protocols. On the other hand, it will be also useful to describe sequences of actions that lead to (un)stable states of Chord networks, in order to be able to analyze properties of systems incorporating Chord and assuming its correctness, as it is the case with the discovery and/or control service of an IoT system.

The paper [14] uses the theory of stochastic processes to give estimations of the probability that a Chord network is in a particular state. The correctness of the Chord's stabilization algorithm is proved in the framework of $\pi$-calculus by showing the equivalence of the corresponding specification and implementation in [12,13], but assuming that nodes cannot leave a network. The correctness of the pure join model is also proved using the Alloy formal language in [16], and some counterexamples to correctness of Chord ring-maintenance in the general case are presented.

As we mentioned in the Introduction, using DHT/Chord in IoT domain is not a novelty [2,8,9,10,11]. In all papers regular discovery (and/or control) service, to improve performances, is replaced by DHT protocol. In [8] authors proposed distributed control plane. They consider the problem how to deliver control messages to the devices that are in sleeping mode most of the time. The proposed DHT algorithm, to realize the *control-plane*, is Chord. The system consists from two type of nodes - peers and clients. Peers are those nodes which participate in the Chord network, and by the assumption they are **stable**. Clients are "normal" nodes that use peers as proxies to connect to the Chord network. Paper [2] introduces scalable, self-configuring and automated service and resource discovery mechanism based on structured DHT architecture to provide support of more complex search queries. Article [9] presents the overview and comparison of the discovery service mechanisms in IoT domain, both traditional and distributed approaches. In [10] authors give the description of a novel discovery service for IoT. In this approach the information repositories are organized in a DHT network to enable multidimensional search procedure. Authors of [11] presented discovery service, that improves scalability, load balance and reliability, for objects carrying RFID tags based on double Chord ring . In all these articles, the **correctness** of the Chord protocol was accepted **for granted**, or by an assumption of stable DHT network.

In [17] a joint frame for reasoning about knowledge and linear time is presented, and the proof of weak completeness for a logic which combines expressions about knowledge with linear time is provided. Related strongly complete logics that concern linear and branching time are presented in [19,20,21]. We use this framework, as a starting point for our logic of time and knowledge. We describe the Chord protocol using the formal language of that temporal epistemic logic and prove properties of the Chord protocol.

## 2.2. Contributions

The main contributions of this paper are:

- a description of the Chord protocol using a temporal epistemic logic;
- a proof of the correctness of the maintenance of the ring topology of the Chord protocol with the respect to the set of possible executions called regular runs.

This work was motivated by the importance of the discovery and control service of an IoT system and the obvious fact that errors in concurrent systems are difficult to reproduce and find merely by program testing. This proof could be, also, the foundation for the formal proof created using a formal proof assistant (like, Coq or Isabelle/HOL).

## 3. Chord Protocol

The Chord protocol was introduced in [5,6,7] where its specification in C++-like pseudocode was given. Here we provide a short description of the Chord protocol that is needed to understand the rest of the paper.

Nodes that run the Chord protocol construct a network that is ring-shaped. The main operations supported by the Chord protocol are:

- adding a node to network,
- removing a node from a network, and
- mapping the given key onto a node using consistent hashing.

The *primitive actions* on which the procedure of maintaining the ring topology in the Chord protocol is based are:

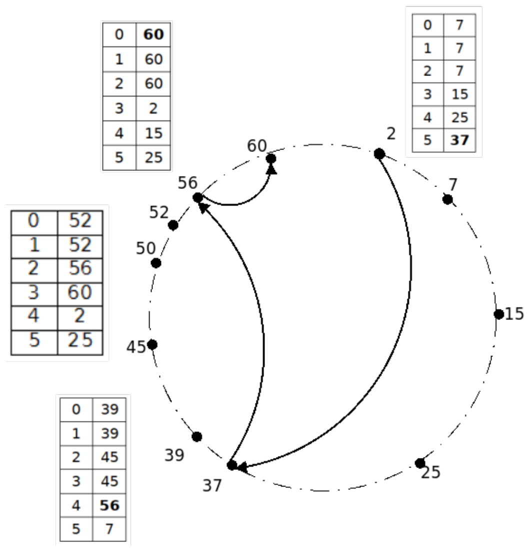
- a node starts a network,
- a new node joins a network,
- a node leaves a network,
- a node updates its successor, and
- a periodic check of the predecessor.

The consistent hashing is used because it provides load-balancing, i.e., every node receives roughly the same number of keys, and when nodes join or leave the network only a few keys are required to be moved [18]. Since Chord networks are overlay systems, each node in a network, which consists of $N$-nodes, needs "routing" information about only $O(\log N)$ other nodes, and resolves all lookups via $O(\log N)$ messages to other nodes.

Nodes are assigned random identifiers and objects are stored across these nodes using consistent hashing. The identifier for a node (a key), $hash(node)$ ($hash(key)$) is obtained by hashing the node's IP address, or the value of the key, respectively. The length of identifiers, e.g., $m$ bits, guarantees that the probability that two objects of the same type are assigned the same identifiers is negligible. Identifiers are ordered in an identifier circle modulo $2^m$ (see Figure 1), while all arithmetic is preformed modulo $2^m$. A key is assigned to a node if their $hash$-values are equal. Otherwise, if there is no node such that $hash(node) = hash(key)$, the key is assigned to the first node in the ring such that $hash(node) > hash(key)$.

Every node stores its current successor and predecessor nodes in the identifier circle. To improve performance of the lookup procedure, every node organizes routing information in the *Finger Table* with up to $m$ entries. The $i^{th}$ entry in the table which belongs to the node $n$ contains the identifier of the first node $s$ such that $s = successor(n + 2^{i-1})$, where $1 \leqslant i \leqslant m$. In other words, the node $s$ succeeds the node $n$ by at least $2^{i-1}$ in the identifier circle. Figure 1 presents Finger tables of nodes $n_2$, $n_{37}$, $n_{50}$ and $n_{56}$.

A node can be aware of only a few other nodes in the system. For example, node $n_2$ from Figure 1 knows about the existence of only 4 other nodes. Other nodes can have different node identifiers in almost every entry in its Finger table, like the node $n_{50}$ from Figure 1.



| | |
|---|---|
| 0 | **60** |
| 1 | 60 |
| 2 | 60 |
| 3 | 2 |
| 4 | 15 |
| 5 | 25 |

| | |
|---|---|
| 0 | 7 |
| 1 | 7 |
| 2 | 7 |
| 3 | 15 |
| 4 | 25 |
| 5 | **37** |

| | |
|---|---|
| 0 | 52 |
| 1 | 52 |
| 2 | 56 |
| 3 | 60 |
| 4 | 2 |
| 5 | 25 |

| | |
|---|---|
| 0 | 39 |
| 1 | 39 |
| 2 | 45 |
| 3 | 45 |
| 4 | **56** |
| 5 | 7 |

**Fig. 1.** Chord lookup procedure. Node $n_2$ is looking for the node responsible for the key with the identifier 57.

During the lookup procedure, a node forwards a query to the largest element of the Finger table smaller than the key used in the query, in respect to the used arithmetic modulo $2^m$. In the example illustrated by Figure 1, if $n_2$ is looking for the responsible node for the key with identifier $57$, it will forward this query to node $n_{37}$, the closest predeceasing node from its finger table to the identifier 57, the closes node from its finger table. After that, this query will be forwarded to $n_{56}$ (again as the closest predeceasing node from its finger table to the identifier 57), until it finally ends at $n_{60}$, as the successor of the node $n_{56}$. The answer if $n_{60}$ contains the key and respected value with identifier 57 will be returned to node that started query, in this case $n_2$.

When a node $n$ joins an existing network, certain keys previously assigned to $n$'s successor now become assigned to $n$. When a node $n$ leaves the network regularly, it notifies its predecessor and successor and reassigns all of its keys to the successor.

The stabilization procedure implemented by Chord runs periodically in the background at each node and must guarantee that each node's finger table, predecessor and successor pointers are up to date.

Figure 2 illustrates the process of stabilization after joining of a new node $n_5$ between nodes $n_2$ and $n_7$. Figure 2a shows a pair of stable nodes $n_2$ and $n_7$ that are a part of the

Chord ring. When a new node $n_5$ joins the system, it sets its successor to $n_7$ as presented in Figure 2b. Node $n_7$ will then set its predecessor to $n_5$ as seen in Figure 2c, and node $n_2$ will later set its successor to $n_5$ as shown in Figure 2d. Finally, node $n_5$ will set its predecessor to $n_2$ as seen in Figure 2e.

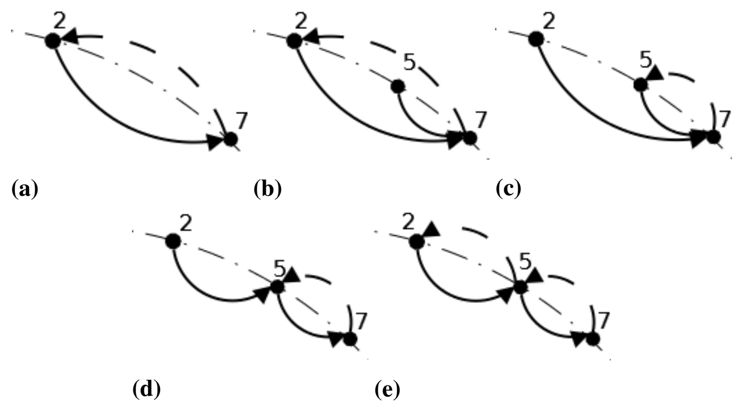


**Fig. 2.** Stabilization process during the joining of a new node: (a) a stable pair of nodes $n_2$ and $n_7$; (b) node $n_5$ joins and sets its successor to $n_7$; (c) node $n_7$ sets its predecessor to $n_5$; (d) node $n_2$ sets its successor to $n_5$; (e) node $n_5$ sets its predecessor to $n_2$.

Figure 3 illustrates the process of stabilization after leaving of the node $n_5$. Figure 3a shows two pairs of stable nodes $n_2$ and $n_5$, and $n_5$ and $n_7$ that are a part of the Chord ring. When the node $n_5$ leaves the system, nodes $n_2$ and $n_7$ point to nothing, as presented in Figure 3b. Then, node $n_2$ will set its successor to $n_7$ as shown in Figure 3c. Finally, node $n_7$ will set its predecessor to $n_2$ as seen in Figure 3d.

## 4. Logic of Time and Knowledge

As we mentioned in the previous Section, a system which runs the Chord protocol is a dynamic multi-agent system, where every node has it own partial view of the surrounding environment. To be able to reason about such systems, we need to introduce a framework for formal description of changes of the knowledge of a node during the time, and which allows nodes to share their knowledge. In this section we present the corresponding temporal-epistemic logic.

### 4.1. Syntax

Let $\mathbb{N}$ be the set of non-negative integers. We denote $\mathbf{Nodes} = \{n_0, \dots n_{m-1}\}$, where $m \in \mathbb{N}$, and then let $\mathbf{N_1} = \mathbf{Nodes} \cup \{u\}$ be the set of propositional variables. The elements from the set $\mathbf{Nodes}$ will represent nodes of the Chord network, while $u$ is the special letter used in the situation when some value is undefined.

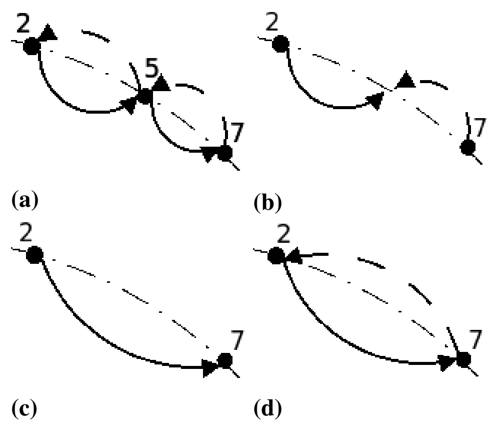
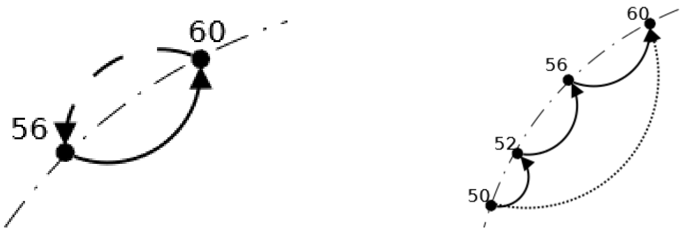


**Fig. 3.** Stabilization process after leaving of a Chord ring node: (a) two stable pairs of nodes $n_2$ and $n_5$, and $n_5$ and $n_7$; (b) node $n_5$ leaves the Chord ring; (c) node $n_2$ sets its successor to $n_7$; (d) node $n_7$ sets its predecessor to $n_2$;.

The set $For$ of all formulas is the smallest superset of $\mathbf{N_1}$ which is closed under the following formation rules:

- $\langle \phi, \psi \rangle \mapsto \phi * \psi$ where $* \in \{\succ, \prec\}$ and $\phi, \psi \in \mathbf{N_1}$,
- $\langle \phi, \psi, \varphi \rangle \mapsto \phi \mathtt{M} \langle \psi, \varphi \rangle$ where $\phi, \psi, \varphi \in \mathbf{Nodes}$,
- $\psi \mapsto * \psi$ where $* \in \{\neg, \bigcirc, \bullet, \mathtt{K}_i\}$,
- $\langle \phi, \psi \rangle \mapsto \phi * \psi$ where $* \in \{\wedge, \mathtt{G}, \mathtt{H}\}$.

The operators $\succ$ and $\prec$ represent the relations successor and predecessor of a node, respectively. The tip of the “arrow” is pointing to the node with “greater” identifier, with respect to the ordering determined by the ring shaped Chord network. An abbreviation $n_i \succ^2 n_k$ will be used for $n_i, n_k \in N$ iff there is an $n_j \in N$ such that $n_i \succ n_j$ and $n_j \succ n_k$, and $n_k \prec^2 n_i$ for $n_i, n_k \in N$ iff there is an $n_j \in N$ such that $n_k \prec n_j$ and $n_j \prec n_i$. Similarly, we can define $n_j \succ^i n_k$, as well as $n_j \prec^i n_k$ for $n_j, n_k \in N$ and $0 < i < m$. Figure 4 illustrates the relations $\succ$, $\prec$ (Figure 4a) and $\succ^i$ (Figure 4b).



**(a).** $n_{56} \succ n_{60}$ and $n_{60} \prec n_{56}$ **(b).** $n_{50} \succ^3 n_{60}$

**Fig. 4.** Examples of $\succ$, $\prec$ and $\succ^i$

The operators $\neg$ and $\wedge$ are logical negation and conjunction. The operators $\bigcirc$, $\bullet$, $\mathtt{G}$ and $\mathtt{H}$ are standard temporal operators *next*, *previous*, *always in the future* and *always in the past*. The operator $\mathtt{K}_i$ represents the knowledge of the node $i$.

The remaining logical $\vee$, $\rightarrow$, $\leftrightarrow$, and temporal $\mathtt{F}$ (*eventually in the future*), $\mathtt{P}$ (*eventually in the past*) connectives, $\mathtt{G}$, $\mathtt{H}$ are defined in the usual way:

- $\phi \vee \psi =_{def} \neg(\neg\phi \wedge \neg\psi)$,
- $\phi \rightarrow \psi =_{def} \neg\phi \vee \psi$,
- $\phi \leftrightarrow \psi =_{def} (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$,
- $\mathtt{F}\psi =_{def} \neg\mathtt{G}\neg\psi$,
- $\mathtt{P}\psi =_{def} \neg\mathtt{H}\neg\psi$,
- $\bigcirc^0\psi =_{def} \psi; \bigcirc^{n+1}\psi = \bigcirc\bigcirc^n \psi, n \geqslant 0$,
- $\bullet^0\psi =_{def} \psi; \bullet^{n+1}\psi = \bullet\bullet^n\psi, n \geqslant 0$.

## 4.2. Semantics

In this paper we will consider time flow which is isomorphic to the set $\mathbb{N}$. We take into account both future and past.

We will defined models as Kripke's structures, where the central notion is that of run. A run is an ordered list (a sequence) of consecutive states of the system and corresponds to a possible execution which starts in an initial state.

**Definition 1.** *A model $\mathcal{M}$ is any tuple $\langle R, \pi, \mathcal{A}, \mathcal{K}\rangle$ such that*

- *$R$ is the set of runs, where:*
  - *every run $r = \langle (x_0^t, \ldots, x_{m-1}^t) | t = 0, 1, 2 \ldots \rangle$, is a countably infinite sequence, where $x_i^t \in \{\top, \bot\}$, and*
  - *a state (or a possible world) of a run $r$ is $\langle r, t\rangle = (x_0^t, \ldots, x_{m-1}^t)$,*
- *$\pi : R \times \mathbb{N} \times \mathbf{N_1} \rightarrow \{\top, \bot\}$ is the set of valuations:*
  - *$\pi(r, t, n_l) = x_l^t$, associates truth values to propositional letters of the possible world $\langle r, t\rangle$,*
- *$\mathcal{A}$ associates sets of active nodes to possible worlds, and*
- *$\mathcal{K} = \{\mathcal{K}_a : a \in \mathbf{A}\}$ is the set of transitive and symmetric accessibility relations for nodes, such that:*
  - *if $a \notin \mathcal{A}(\langle r, t\rangle)$, then $\langle r, t\rangle \mathcal{K}_a \langle r', t'\rangle$ is false for all $r' \in R$ and all $t' \in \mathbb{N}$.*

Actually, a state, or a possible world $\langle r, t\rangle$, in a run $r$ represents the state of the system in the corresponding time instant $t$, and we will alternatively use these notions in the rest of the paper.

Figure 5 illustrates a Kripke model which contains the runs $r^1, r^2, r^3, r^4$, where $r^1$ is the sequence of $\langle r^1, 0\rangle$, $\langle r^1, 1\rangle$, $\langle r^1, 2\rangle$, etc. and similarly for other runs. In this model, for example $\langle r^2, 1\rangle \mathcal{K}_1 \langle r^2, 2\rangle$, etc.

An $n_i \in \mathbf{Nodes}$ is $true$ in the time instant $t$ in the run $r$ ($x_i^t = \top$) if the Chord network node $i$ is active in the corresponding realization of the network. For $n_i, n_j, n_k \in \mathbf{Nodes}$ we define the relation $\mathtt{M}$ which represents the fact that $n_i$ is the member of the ring interval $(n_j, n_k]$ as: $n_i\mathtt{M}\langle n_j, n_k\rangle$ is $true$ iff

- $j = k$, or
- $j < k$ and $j < i \leqslant k$, or
- $k < j$ and $\neg(k < i \leqslant j)$.

Note that the relation $\mathtt{M}$ is defined for all members of the set $\mathbf{Nodes}$, regardless the fact if the members are active or not.

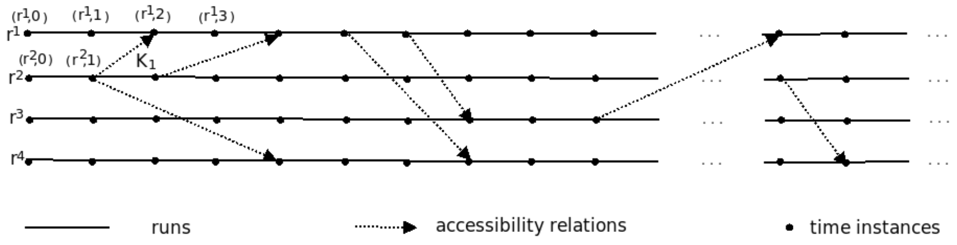


**Fig. 5.** Kripke model

## 4.3. Satisfiability relation

A formula is satisfiable if there is a possible world in a model which makes that formula true.

**Definition 2.** *Let $\mathcal{M} = \langle R, \pi, \mathcal{A}, \mathcal{K} \rangle$ be any model. The satisfiability relation $\models$ (formula $\alpha$ is satisfied in a time instant $t$ of a run $r$) is defined recursively as follows:*

1. $\langle r,t \rangle \models n$ *iff* $\pi(r,t,n) = true$, $n \in \mathbf{N_1}$
2. $\langle r,t \rangle \models \alpha \wedge \beta$ *iff* $\langle r,t \rangle \models \alpha$ *and* $\langle r,t \rangle \models \beta$
3. $\langle r,t \rangle \models \neg\alpha$ *iff not* $\langle r,t \rangle \models \alpha$ ( $\langle r,t \rangle \not\models \alpha$)
4. $\langle r,t \rangle \models \bigcirc\alpha$ *iff* $\langle r,t+1 \rangle \models \alpha$
5. $\langle r,t+1 \rangle \models \bullet\alpha$ *iff* $\langle r,t \rangle \models \alpha$
6. $\langle r,0 \rangle \models \bullet\alpha$
7. $\langle r,t \rangle \models \mathtt{G}\alpha$ *iff for all* $i \geqslant 0$ *holds* $\langle r,t+i \rangle \models \alpha$
8. $\langle r,t \rangle \models \mathtt{H}\alpha$ *iff for all* $0 \leqslant i \leqslant t$ *holds* $\langle r,t-i \rangle \models \alpha$
9. $\langle r,t \rangle \models \mathtt{K}_i\alpha$ *iff* $\langle r',t' \rangle \models \alpha$ *for all* $\langle r',t' \rangle \in \mathcal{K}_i(\langle r,t \rangle)$
10. $\langle r,t \rangle \models n_i \succ n_j$ *iff*
    (a) $i = j$ *and* $\langle r,t \rangle \models n_i \wedge \mathtt{K}_i(\bigwedge_{n_j \in \mathbf{Nodes} \setminus \{n_i\}} \neg n_j)$
    (b) $i < j \leqslant m$ *and* $\langle r,t \rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=i+1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (c) $j < i < m$ *and* $\langle r,t \rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=i+1}^{m} \neg n_k) \wedge \mathtt{K}_i(\bigwedge_{k=1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (d) $j < i$ *and* $i = m$ *and* $\langle r,t \rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
11. $\langle r,t \rangle \models n_j \prec n_i$ *iff*
    (a) $i = j$, $t \neq 0$ *and* $\langle r,t \rangle \models n_i \wedge \mathtt{K}_i(\bigwedge_{n_k \in \mathbf{Nodes} \setminus \{n_i\}} \neg n_k)$
    (b) $i < j \leqslant m$ *and* $\langle r,t \rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=i+1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (c) $j < i < m$ *and* $\langle r,t \rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=i+1}^{m} \neg n_k) \wedge \mathtt{K}_i(\bigwedge_{k=1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (d) $j < i$ *and* $i = m$ *and* $\langle r,t \rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (e) $n_i = u$ *and* $\langle r,t \rangle \models \neg n_j \vee (n_j \wedge (\bullet(\neg \mathtt{K}_k(n_k \succ n_j))))$

Now, we can explain the intuitive meaning of the operators. For example, the formula $\bigcirc\alpha$ is true in a state $\langle r,t \rangle$ of a run $r$ iff $\alpha$ is true in the next state $\langle r,t+1 \rangle$ of the same run. Next, $\mathtt{K}_i\alpha$ is true in a state $\langle r,t \rangle$ iff $\alpha$ is true in all states $\langle r',t' \rangle$ accessible from $\langle r,t \rangle$ by the relation $\mathcal{K}_i(\langle r,t \rangle)$. On the other hand, the formula $n_i \succ n_j$ is true if there is only one active node in the Chord network (case (a)), or there is not other active nodes between $n_i$ and $n_j$ and node $n_i$ is aware of activity of $n_j$ (cases (b)-(d)). The cases (b)-(d) take in account the ring structure of the Chord network, i.e. $[n_i, n_j]$ is one ring interval. Similarly for $n_j \prec n_i$.

By Definition 2 it is trivial to prove that the following lemma holds:

**Lemma 1.** **TP:** $\langle r,t\rangle \models (\bigcirc\alpha \wedge \bigcirc\beta) \leftrightarrow \bigcirc(\alpha \wedge \beta)$.

We use this property in the proofs provided in Appendix A.

## 5. Proof of Correctness

In this Section we analyze correctness of the Chord protocol. More precisely, we prove that any execution of a regular run maintains the ring topology of the corresponding network. It means that the network will be brought, after a finite number of steps, from a state in which links between nodes do not form a ring to a stable state. First, we need to introduce the following definitions:

**Definition 3 (Stable pair).** *The pair of active nodes $\langle n_k, n_l\rangle$ is stable in a time instant $t$ of a run $r$ (i.e., in the state $\langle r,t\rangle$), denoted with $n_k \Cap n_l$, iff*

$$\langle r,t\rangle \models (n_l \succ^{m_1} n_k) \wedge (\bigwedge_{j=1}^{m_1} \mathtt{K}_{i_j}(n_{i_j} \succ n_{i_{j+1}})) \wedge (n_l \prec^{m_1} n_k) \wedge (\bigwedge_{j=1}^{m_1} \mathtt{K}_{i_{j+1}}(n_{i_{j+1}} \prec n_{i_j})),$$

*where $n_k, n_l, n_{i_j} \in \mathcal{A}(\langle r,t\rangle)$ for $1 \leqslant j \leqslant m_1$.*

Intuitively, a pair $\langle n_k, n_l\rangle$ is stable if:

- the nodes $n_k$ and $n_l$ are active, and
- every active node $n_j$ such that $n_j\mathtt{M}\langle n_k, n_l\rangle$ knows its successor and predecessor.

It means that there is no node $n_i$ between $n_k$ and $n_l$ which tries to join or leave the network in $\langle r,t\rangle$.

**Definition 4 (Stable network).** *A Chord network is stable (we denote it with $\circledcirc$) at $\langle r,t\rangle$ iff $n_k \Cap n_k$ for all $n_k \in \mathcal{A}(\langle r,t\rangle)$.*

Intuitively, the whole network is stable if all successor and predecessor pointers are sorted.

**Definition 5.** *A node $n_i$ is a member of a stable pair $\langle n_k, n_l\rangle$ iff $n_i\mathtt{M}\langle n_k, n_l\rangle$ and $n_k \Cap n_l$.*

**Definition 6 (Regular runs).** *A run is regular if in every state of the run a node can leave the network only if it is a member of a stable pair of nodes.*

Definition 6 is essential in producing a deterministic proof of the correctness of the Chord protocol, since if it is allowed that nodes can leave the network when they are members of unstable pairs, there are counter examples in which those nodes can be left isolated or the network can be divided into more then one separate subnetworks, as it is shown in [16].

We assume the following form of the fairness condition: there is a constant $f \in \mathbb{N}$ which denotes a uniform limit from above for duration of all primitive actions of the Chord protocol (listed in Section 3).

The basic properties of $\succ$ and $\prec$ relations can be described with:

AS1: $\langle r,t\rangle \models n_i \succ n_j \to \bigwedge_{n_k \in \mathbf{N_1}\setminus\{n_j\}} \neg(n_i \succ n_k), n_i, n_j \in \mathbf{Nodes}$
AS2: $\langle r,t\rangle \models n_i \prec n_j \to \bigwedge_{n_k \in \mathbf{N_1}\setminus\{n_j\}} \neg(n_i \prec n_k), n_i, n_j \in \mathbf{Nodes}$
AS3: $\langle r,t\rangle \models n_i \prec n_j \to \bigwedge_{n_k \in \mathbf{N_1}\setminus\{n_i\}} \neg(n_k \prec n_j), n_i, n_j \in \mathbf{Nodes}$
AS4: $\langle r,t\rangle \models n_i \prec n_j \to n_j \succ n_i, n_i, n_j \in \mathbf{Nodes}$
AS5: $\langle r,t\rangle \models n_i \succ n_j \to \mathtt{K}_i(n_i \succ n_j), n_i, n_j \in \mathbf{Nodes}$
AS6: $\langle r,t\rangle \models ((n_i \succ n_j) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle \wedge \bigcirc(\neg\mathtt{K}_i n_k)) \to \bigcirc(n_i \succ n_j)$, $n_i, n_j, n_k \in \mathbf{Nodes}$
AS7: $\langle r,t\rangle \models ((n_i \prec n_j) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle \wedge \bigcirc(\neg\mathtt{K}_i n_k)) \to \bigcirc(n_i \prec n_j)$, $n_i, n_j, n_k \in \mathbf{Nodes}$

[AS1] says that a node can have only one successor. [AS2] says that a node can be predecessor of only one node. [AS3] says that a node can have only one predecessor. [AS4] says that if a node is predecessor of some other node, that other node has to be its successor. [AS5] says that if a node $n_i$ has the successor $n_j$, then $n_i$ knows that $n_j$ is its successor. [AS6] says that the current successor ($n_j$) of a node ($n_i$) will be successor of the node in the next time instant ($\bigcirc(n_i \succ n_j)$), assuming that $n_i$ does not know ($\bigcirc(\neg\mathtt{K}_i n_k)$) that a new node $n_k$ which belongs to the ring interval $[n_i, n_j)$ (i.e., $n_k\mathtt{M}\langle n_i, n_j\rangle$) joins the network. Similarly, [AS7] says when the current predecessor will be the predecessor in the next time instant, if there are no new nodes.

The primitive actions of the Chord network can be describe in the following way:

$\rho_S$: $\langle r,t\rangle \models \mathtt{H}(\bigwedge_{n_j \in \mathbf{Nodes}} \neg n_j) \wedge n_i \wedge (\bigwedge_{n_j \in \mathbf{Nodes}\setminus\{n_i\}} \neg n_j) \wedge \mathtt{K}_i(n_i \succ n_i) \wedge \mathtt{K}_i(n_i \prec u)$ for one $n_i \in \mathbf{Nodes}$,
$\rho_{J,i}$: $\langle r,t\rangle \models \bullet(\neg n_i) \wedge n_i \wedge \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_i(n_i \prec u), n_j \in \mathcal{A}(\langle r,t\rangle)$, $n_i \in \mathbf{Nodes}, i \neq j$,
$\rho_{L,i}$: $\langle r,t\rangle \models \bullet(n_i \wedge n_j \Cap n_k) \wedge n_i\mathtt{M}\langle n_j, n_k\rangle \wedge \neg n_i, n_i \in \mathbf{Nodes}\; n_k, n_j \in \mathcal{A}(\langle r,t\rangle)$
$\rho_{S1,i,j}$: $\langle r,t\rangle \models (\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec u)) \vee (\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_i\mathtt{M}\langle n_k, n_j\rangle) \to \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_j(n_j \prec n_i), n_i, n_k, n_j \in \mathcal{A}(\langle r,t\rangle)$,
$\rho_{S2,i,j}$: $\langle r,t\rangle \models \mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle \to \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_k)$, $n_i, n_k, n_j \in \mathcal{A}(\langle r,t\rangle)$,
$\rho_{S3,i}$: $\langle r,t\rangle \models \mathtt{K}_i(n_i \succ n_j) \wedge \neg n_j \to \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_k) \wedge \bigvee_{l \in \mathcal{A}(\langle r,t\rangle)} \neg n_l\mathtt{M}\langle n_i, n_k\rangle$, $n_i, n_k \in \mathcal{A}(\langle r,t\rangle), n_j \in \mathbf{Nodes}$
$\rho_{S4,i}$: $\langle r,t\rangle \models \mathtt{K}_i(n_j \prec n_i) \wedge \neg n_j \to \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(u \prec n_i), n_i \in \mathcal{A}(\langle r,t\rangle)$, $n_j \in \mathbf{Nodes}$

[$\rho_S$] describes the start of the new Chord network, i.e., there was no active node in the past ($\mathtt{H}(\bigwedge_{n_j \in \mathbf{Nodes}} \neg n_j)$), in the current time instant only $n_i$ becomes active ($n_i \wedge (\bigwedge_{n_j \in \mathbf{Nodes}\setminus\{n_i\}} \neg n_j)$), and $n_i$ knows that it is its own successor ($\mathtt{K}_i(n_i \succ n_i)$), while its predecessor is still undefined ($\mathtt{K}_i(n_i \prec u)$). Similarly, for the other formulas: [$\rho_{J,i}$] represents the situation when a new node $n_i$ joins the existing Chord network, while [$\rho_{L,i}$] represents the situation when a new node $n_i$ leaves the existing Chord network in a regular run. [$\rho_{S1,i,j}$] - [$\rho_{S4,i}$] characterize stabilization processes (that make predecessor and successor pointers up to date).

To be able to describe periodicity of the stabilization process, we introduce the following formulas:

ACF1: $\langle r,t\rangle \models n_i \wedge \rho_S \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S1,i,j}, n_i \in \mathcal{A}(\langle r,t\rangle)$,

ACF2: $\langle r,t\rangle \models n_i \wedge \rho_S \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S2,i,j}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF3: $\langle r,t\rangle \models n_i \wedge \rho_{J,i} \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S1,i,j}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF4: $\langle r,t\rangle \models n_i \wedge \rho_{J,i} \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S2,i,j}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF5: $\langle r,t\rangle \models n_i \wedge \rho_{S1,i,k} \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S1,i,j}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
$k \in \{0, m-1\}$,
ACF6: $\langle r,t\rangle \models n_i \wedge \rho_{S2,i,k} \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S2,i,j}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
$k \in \{0, m-1\}$,
ACF7: $\langle r,t\rangle \models n_i \wedge \rho_S \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S3,i}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF8: $\langle r,t\rangle \models n_i \wedge \rho_S \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S4,i}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF9: $\langle r,t\rangle \models n_i \wedge \rho_{J,i} \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S3,i}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF10: $\langle r,t\rangle \models n_i \wedge \rho_{J,i} \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S4,i}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF11: $\langle r,t\rangle \models n_i \wedge \rho_{S3,i} \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S3,i}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF12: $\langle r,t\rangle \models n_i \wedge \rho_{S4,i} \to \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S4,i}$, $n_i \in \mathcal{A}(\langle r,t\rangle)$.

The correctness of the Chord protocol can be proved by compounding simpler cases. The statements 2-6 guarantee that the successor and predecessor pointers for each node will be eventually sorted after one or more nodes join existing or start a new network. Theorem 6 expresses the correctness of the model of executions without failures of nodes and corresponds to Theorem IV.3 from [5]. The statements 7-9 consider possible leaving of a node. Lemma 10 says that a stable pair of nodes in a Chord network eventually becomes stable after adding/removing of a node between them with the respect to the regular runs, while Theorem 4 shows the same, but for a stable network.

**Lemma 2.** *Let a node start a new Chord network. Then, there is a finite period of time after which the network will be stable, if no other nodes are trying to join in the meanwhile.*

**Lemma 3.** *Let a new node join a stable Chord network which consists of only one node. Then, there is a finite period of time after which the network will be stable again, if no other nodes are trying to join in the meanwhile.*

Proofs of Lemmas 2 and 3 are similar to the proof of Lemma 4 (that can be found in Appendix A).

**Lemma 4.** *Let a node join a Chord network, between two nodes which constitute a stable pair, such that the second node is the successor of the first node. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join the network between the nodes that constitute the particular stable pair in the meanwhile.*

**Lemma 5.** *Let a node join a Chord network, between two nodes which constitute a stable pair. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join in the meanwhile.*

*Proof.* Since one new node is joining the network between a stable pair, we can choose two nodes which are each others successor and predecessor and the new node is joining between them, so we can apply Lemma 4.

**Lemma 6.** *Let a Chord network contain a stable pair. If a sequence of nodes join between the nodes that constitute this stable pair, then there is a finite period of time after which the starting pair will be stable again.*

*Proof.* If we assume that all nodes that want to join the network have different successors, by Lemma 5 the statement holds.

If this is not the case, we can assume that $n_i \Cap n_k$ and that the set of nodes $n_{j_1}, n_{j_2}, \ldots$, such that $i \leqslant \ldots \leqslant j_2 \leqslant j_1 \leqslant k$, are joining this stable pair. Then, we can apply Lemma 5 on the tuples $\langle n_i, n_{j_1}, n_k \rangle$, $\langle n_i, n_{j_2}, n_{j_1} \rangle, \ldots$. This process will produce the stable pair $n_i \Cap n_k$, again.

**Lemma 7.** *Let a Chord network contain a stable pair and let a node between them leave the network. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join in the meanwhile.*

Proofs of Lemmas 8 and 9 are similar to the proof of Lemma 7 (that can be found in Appendix A).

**Lemma 8.** *Let a Chord network contain a stable pair $n_i \Cap n_k$. Let a node which is between those nodes leave the network followed by several nodes which want to join between $\langle n_i, n_k \rangle$.Then, there is a finite period of time after which $\langle n_i, n_k \rangle$ will be stable again.*

**Lemma 9.** *Let a Chord network contain a stable pair. Let a node, which is in between those nodes, leave the network. Then, there is a finite period of time after which the starting pair will be stable again.*

**Lemma 10.** *Let a finite initial segment of a run of a Chord network produce the state $\langle r, t \rangle$, and $\langle n, n' \rangle \in \mathcal{A}(\langle r, t \rangle)$ be nodes that do not leave the network. Then, there is a finite period of time after which $\langle n, n' \rangle$ will be stable, i.e., $n \Cap n'$.*

*Proof.* First note that, if the pair is stable at $\langle r, t \rangle$, then the statement trivially holds. Otherwise, since we consider only regular runs, only joining of the new nodes between $\langle n, n' \rangle$ is possible. So, this is similar to Lemma 9.

**Theorem 1.** *Let a finite initial segment of a run produce the state $\langle r, t \rangle$ of a Chord network. Then, there is a finite period of time after which the network will be stable again:*

$$\langle r, t \rangle \models \neg \circledcirc \rightarrow \mathtt{F} \circledcirc$$

*Proof.* We assume that at least one node does not leave the network. Note that an unstable state can be reached under following conditions:

– if a node starts the Chord network, which is considered in Lemma 2,
– if a node joins the Chord network, which is considered in Lemma 3, 4, 5 and 6,
– if a node leaves the Chord network, which is considered in Lemma 7, and
– if some nodes are leaving and some nodes are joining the Chord network, which is considered in Lemma 8, 9 and 10.

So, as an unstable state can be reached only in some of the cases that are already considered, this theorem is the corollary of the lemmas 2 – 10.

## 6. Conclusion

Discovery and control services are the core part of every IoT system. To improve the performances of them, we found several examples where the regular approach was replaced by DHT, i.e. the Chord protocol. All these examples do not consider correctness of the Chord protocol and take it for granted, or consider it by very strong assumptions. To be used in a proper manner, it is necessary to describe in a deterministic scenario for all the situation when the Chord protocol manifest incorrect behavior. For that purpose, we define the notion of regular runs and prove the correctness of the maintenance of the ring topology of the Chord protocol with the respect of them, using the framework of time and knowledge.

One of the possible directions for further work is to apply the similar technique to describe other DHT protocols and other cloud processes. Another challenge could be to verify the given proof in one of the formal proof assistants (e.g., Coq, Isabelle/HOL). It might also produce a certified program implementation from the proof of correctness.

**Acknowledgments.** The work presented here was supported by Serbian Ministry of Education, Science and Technology Development (the projects ON174026 and III44006), through Matematički institut SANU, and Croatian Ministry of Science and Education. This work was supported in part by the Slovenian Research Agency within the research program Algorithms and Optimization Methods in Telecommunications.

## References

1. L. Atzori, A. Iera, G. Morabito. *The Internet of things: A survey*. In *Computer Networks*, 54.15, 2787–2805, 2010.
2. S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, L. Veltri. *A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things*. In *IEEE Internet of Things Journal*, Vol. 1, No. 5, 508–521, 2014.
3. R. Rodrigues, P. Druschel. *Peer-to-Peer Systems* In *Communications of the ACM*, Vol. 53 Issue 10, pages 72–82, October 2010.
4. I. Taylor. *From P2P to Web Services and Grids*. Springer-Verlag, 2005.
5. I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan. *Chord: A Scalable Peer-to-Peer Lookup service for Internet Applications*. In *ACM SIGCOMM*, pages 149–160, 2001.
6. I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, H. Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*. MIT Technical report, TR-819, 2001.
7. I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, H. Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*. In *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, 17 – 32, 2003.
8. J. J. Bolonio, M. Urueña, G. Camarillo. *A Distributed Control Plane for the Internet of Things Based on a Distributed Hash Table*. In *Mobile Networks and Management*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 125, 108–121, 2013.
9. S. Evdokimov, B. Fabian, S. Kunz, N. Schoenemann. *Comparison of Discovery Service Architectures for the Internet of Things*. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, 2010.
10. F. Paganelli, D. Parlanti. *A DHT-Based Discovery Service for the Internet of Things*. In *Journal of Computer Networks and Communications*, doi:10.1155/2012/107041, 2012.

11. D. Xu, Z. Wu, Z. Wu, Q. Zhang, L. Qin, J. Zhou. *Internet of Things: Hotspot-based Discovery Service Architecture with Security Mechanism.* In *International Journal of Network Security*, Vol. 17, No. 2, 208–216, 2015.
12. R. Bakhshi, D. Gurov. *Verification of Peer-to-peer Algorithms: A Case Study*. Technical report, ICT, 2006.
13. R. Bakhshi, D. Gurov. *Verification of Peer-to-peer Algorithms: A Case Study*. In *Electronic Notes in Theoretical Computer Science* (ENTCS), Volume 181, 35–47, 2007.
14. S. Krishnamurthy, S. El-Ansary, E. Aurell, S. Haridi. *A Statistical Theory of Chord Under Churn*. In *4th International Workshop on Peer-To-Peer Systems*, pages 93–103, 2005.
15. D. Liben-Nowell, H. Balakrishnan, D. R. Karger. *Analysis of the Evolution of Peer-to-Peer Systems*. In *Proc.* $21^{st}$ *ACM Symp. Principles of Distributed Computing (PODC)*, pages 233–242, 2002.
16. P. Zave. *Using Lightweight Modeling to Understand Chord.* In ACM SIGCOMM Computer Communication Review, Vol. 42, Issue 2, pages 50–57, April 2012.
17. R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press, Cambridge, Massachusetts, 1995.
18. D. R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, D. Lewin. *Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web*. In *Proceedings of STOC'97*, pages 654–663, 1997.
19. B. Marinkovic, Z. Ognjanovic, D. Doder, A. Perovic. *A Propositional Linear Time logic with Time Flow Isomorphic to* $\omega^2$. In *Journal of Applied Logic*, 12(2), 208 – 229, 2014.
20. Z. Ognjanovic. *Discrete Linear-time Probabilistic Logics: Completeness, Decidability and Complexity*. In *Journal of Logic Computation*, Vol. 16, No. 2, 257–285, 2006.
21. Z. Ognjanovic, D. Doder, Z. Markovic. *A Branching Time Logic with Two Types of Probability Operators*. In *Fifth International Conference on Scalable Uncertainty Management SUM-2011*, Springer LNCS 6929, 219–232, 2011.

## A. Proofs

Recall that $f$ denotes the maximum of durations of all primitive actions of the Chord protocol.

**Lemma 4.** *Let a node join a Chord network, between two nodes which constitute a stable pair, such that the second node is the successor of the first node. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join the network between the nodes that constitute the particular stable pair in the meanwhile.*

*Proof.* We will show that 5 rounds of the stabilization process (i.e., $5f$ steps of an execution of the Chord protocol) will be enough to guarantee that the considered pair of nodes becomes stable.

Let us assume that $n_i, n_j \in \mathcal{A}(\langle r, t\rangle)$ and $n_i \Cap n_j$, i.e. $(n_i \succ n_j) \wedge (n_j \prec n_i)$ and that $n_k$ tries to join that stable pair. Let us denote

$$\alpha = \bullet(n_i \Cap n_j) \wedge \rho_{J,k} \bigwedge_{n_l \in I} \bigwedge_{t=0}^{5f} \bigcirc^t \neg n_l, I = \{n_l | n_l \mathtt{M}\langle n_i, n_j\rangle, n_k \neq n_l, n_j \neq n_l\}.$$

The formula $\alpha$ says that the pair $\langle n_i, n_j\rangle$ is stable in the previous time instant ($\bullet(n_i \Cap n_j)$), $n_k$ is joining ($\rho_{J,k}$) and during the next $5f$ time instants no other node

($\bigcirc^t \neg n_l$) is trying to join the network between the nodes that constitute initial stable pair ($n_l \mathtt{M}\langle n_i, n_j\rangle$).

By MP we denote the standard inference rule *modus ponens*:

from $\alpha$ and $\alpha \to \beta$ conclude $\beta$.

We have,

$$\langle r,t\rangle \models \alpha \quad (0)$$

$$\langle r,t\rangle \models \mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_i) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle \wedge n_k \quad \text{(by AS6) (1)}$$

$$\langle r,t\rangle \models \mathtt{K}_i(n_i \succ n_j) \quad \text{(by definition of } \wedge \text{ and 1) (2a)}$$

$$\langle r,t\rangle \models \mathtt{K}_j(n_j \prec n_i) \quad \text{(by definition of } \wedge \text{ and 1) (2b)}$$

$$\langle r,t\rangle \models n_k\mathtt{M}\langle n_i, n_j\rangle \quad \text{(by definition of } \wedge \text{ and 1) (2c)}$$

$$\langle r,t\rangle \models n_k \quad \text{(by definition of } \wedge \text{ and 1) (2d)}$$

$$\langle r,t\rangle \models \rho_{J,k} \quad \text{(by definition of } \alpha\text{) (2e)}$$

$$\langle r,t\rangle \models n_k \wedge \rho_{J,k} \quad \text{(by 2d, 2e) (2f)}$$

$$\langle r,t\rangle \models \rho_{J,k} \to \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_k(n_k \succ n_j) \quad \text{(by definition of } \rho_{J,k}\text{) (3)}$$

$$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_k(n_k \succ n_j) \quad \text{(by MP, 2e, 3) (4)}$$

$$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_k(n_k \succ n_j) \to \bigcirc^f \mathtt{K}_k(n_k \succ n_j), \quad \text{(by definition of AS6,4) (5)}$$

$$\langle r,t\rangle \models \bigcirc^f \mathtt{K}_k(n_k \succ n_j) \quad \text{(by MP, 4, 5) (6)}$$

$$\langle r,t\rangle \models n_k \wedge \rho_{J,k} \to \bigvee_{l=0}^{f} \bigcirc^l \rho_{S1,k,j} \quad \text{[ACF3] (7)}$$

$$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc^l \rho_{S1,k,j} \quad \text{(by MP, 2f,7) (8a)}$$

$$\langle r,t\rangle \models \bigcirc^f \rho_{S1,k,j} \quad \text{(by AS6) (8b)}$$

$$\langle r,t\rangle \models \bigcirc^f ((\mathtt{K}_k(n_k \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_i) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle) \to \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_j(n_j \prec n_k)) \quad \text{(by 8b) (9a)}$$

$$\langle r,t\rangle \models \bigcirc^f ((\mathtt{K}_k(n_k \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_i) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle)) \to \bigcirc^f (\bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_j(n_j \prec n_k)) \quad \text{(by AT2, 9a) (9b)}$$

$$\langle r,t\rangle \models \bigcirc^f \mathtt{K}_k(n_k \succ n_j) \quad \text{(by AS6, 6) (10a)}$$

$$\langle r,t\rangle \models \bigcirc^f \mathtt{K}_k(n_j \prec n_i) \quad \text{(by AS6, 2b) (10b)}$$

$$\langle r,t\rangle \models \bigcirc^f (n_k\mathtt{M}\langle n_i, n_j\rangle) \quad \text{(by AS6, 2c) (10c)}$$

$\langle r,t\rangle \models \bigcirc^f(\mathtt{K}_k(n_k \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_i) \wedge n_k\mathtt{M}\langle n_i,n_j\rangle)$ (by TP, 10a, 10b, 10c) (11)

$\langle r,t\rangle \models \bigcirc^f(\bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_j(n_j \prec n_k))$ (by MP, 9,11) (12)

$\langle r,t\rangle \models \bigcirc^{2f}\mathtt{K}_j(n_j \prec n_k)$ (by definition of $\bigcirc$, AS6, 12) (13)

$\langle r,t\rangle \models \bigvee_{l=f}^{2f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S2,i,j}$ (by $n_i \in \mathcal{A}(\langle r,t\rangle)$ and ACF2 or ACF4) (14)

$\langle r,t\rangle \models \bigvee_{l=f}^{2f} \bigcirc^l \rho_{S2,i,k}$ (by definition of $\vee$,14) (15a)

$\langle r,t\rangle \models \bigcirc^{2f}\rho_{S2,i,k}$ (by definition AS6,15a) (15b)

$\langle r,t\rangle \models \bigcirc^{2f}(\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_k\mathtt{M}\langle n_i,n_j\rangle) \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_k))$

(by 15b) (16a)

$\langle r,t\rangle \models \bigcirc^{2f}(\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_k\mathtt{M}\langle n_i,n_j\rangle)) \rightarrow \bigcirc^{2f}(\bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_k))$

(by AT2, 16a) (16b)

$\langle r,t\rangle \models \bigcirc^{2f}\mathtt{K}_i(n_i \succ n_j)$ (by AS6, 2a) (17a)

$\langle r,t\rangle \models \bigcirc^{2f}\mathtt{K}_j(n_j \prec n_k)$ (by AS6, 13) (17b)

$\langle r,t\rangle \models \bigcirc^{2f}(n_k\mathtt{M}\langle n_i,n_j\rangle)$ (by AS6, 2c) (17c)

$\langle r,t\rangle \models \bigcirc^{2f}(\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_k\mathtt{M}\langle n_i,n_j\rangle)$ (by TP, 17a, 17b, 17c) (18)

$\langle r,t\rangle \models \bigcirc^{2f}(\bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_k))$ (by MP, 16b,18) (19)

$\langle r,t\rangle \models \bigcirc^{3f}\mathtt{K}_i(n_i \succ n_k),$ (by definition of $\bigcirc$, AS6,19) (20)

$\langle r,t\rangle \models \bigvee_{l=3f}^{4f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S1,i,j}$ (by $n_i \in \mathcal{A}(\langle r,t\rangle)$ and ACF1 or ACF3) (21)

$\langle r,t\rangle \models \bigvee_{l=3f}^{4f} \bigcirc^l \rho_{S1,i,k}$ (by definition of $\vee$,21) (22a)

$\langle r,t\rangle \models \bigcirc^{4f}\rho_{S1,i,k}$ (by definition AS6, 22a) (22b)

$\langle r,t\rangle \models \bigcirc^{4f}(\mathtt{K}_i(n_i \succ n_k) \wedge \mathtt{K}_k(n_k \prec u) \wedge n_k\mathtt{M}\langle n_i,n_j\rangle) \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_k(n_k \prec n_i))$

(by 22b) (23a)

$\langle r,t\rangle \models \bigcirc^{4f}(\mathtt{K}_i(n_i \succ n_k) \wedge \mathtt{K}_k(n_k \prec u) \wedge n_k\mathtt{M}\langle n_i,n_j\rangle)) \rightarrow \bigcirc^{4f}(\bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_k(n_k \prec n_i))$

(by definition of AT2, 23a) (23b)

$$\langle r,t\rangle \models \bigcirc^{4f}\mathtt{K}_i(n_i \succ n_k) \qquad \text{(by AS6, 29) (24a)}$$
$$\langle r,t\rangle \models \bigcirc^{4f}\mathtt{K}_k(n_k \prec u) \qquad \text{(by AS6, 2e) (24b)}$$
$$\langle r,t\rangle \models \bigcirc^{4f}(n_k\mathtt{M}\langle n_i, n_j\rangle) \qquad \text{(by AS6, 2c) (24c)}$$
$$\langle r,t\rangle \models \bigcirc^{4f}(\mathtt{K}_i(n_i \succ n_k) \wedge \mathtt{K}_k(n_k \prec u) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle) \qquad \text{(by TP, 24a, 24b, 24c) (25)}$$
$$\langle r,t\rangle \models \bigcirc^{4f}(\bigvee_{l=0}^{f} \bigcirc^{l}\mathtt{K}_k(n_k \prec n_i)) \qquad \text{(by MP, 23b,25) (26)}$$
$$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_k(n_k \prec n_i) \qquad \text{(by definition of } \bigcirc\text{, AS6,26) (27)}$$
$$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_k(n_k \succ n_j) \qquad \text{(by AS6, 6) (28)}$$
$$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_j(n_j \prec n_k) \qquad \text{(by AS6, 13) (29)}$$
$$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_i(n_i \succ n_k) \qquad \text{(by AS6, 20) (30)}$$
$$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_k(n_k \prec n_i) \qquad \text{(by AS6, 27) (31)}$$
$$\langle r,t\rangle \models \bigcirc^{5f}(\mathtt{K}_k(n_k \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge \mathtt{K}_i(n_i \succ n_k) \wedge \mathtt{K}_k(n_k \prec n_i))$$
$$\text{(by TP, 28, 29, 30, 31) (32)}$$
$$\langle r,t\rangle \models \bigcirc^{5f}(n_i \Cap n_j) \qquad \text{(by definition of } \Cap\text{) (33)}$$

The last formula represents the statement of this Lemma.

**Lemma 7.** *Let a Chord network contain a stable pair and let a node between them leave the network. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join in the meanwhile.*

*Proof.* We will show that 2 rounds of the stabilization process (i.e., $2f$ steps of an execution of the Chord protocol) will be enough to guarantee that the considered pair of nodes becomes stable.

Let us assume that $n_i, n_j, n_k \in \mathcal{A}(\langle r,t\rangle)$ and $n_i \Cap n_k$, i.e. $(n_i \succ n_j) \wedge (n_j \prec n_i) \wedge (n_j \succ n_k) \wedge (n_j \prec n_k)$ and that $n_j$ tries to leave that stable pair. Let us denote

$$\alpha = \bullet(n_i \Cap n_k) \wedge \rho_{L,j} \bigwedge_{n_l \in I} \bigwedge_{t=0}^{2f} \bigcirc^{t}\neg n_l, I = \{n_l | n_l\mathtt{M}\langle n_i, n_j\rangle, n_k \neq n_l, n_j \neq n_l\}.$$

$\rho_{L,j}$: $\bullet(n_j \wedge n_i \Cap n_k) \wedge n_j\mathtt{M}\langle n_i, n_k\rangle \wedge \neg n_j$
We have,

$$\langle r,t\rangle \models \alpha \qquad (0)$$
$$\langle r,t\rangle \models \mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_k(n_j \prec n_k) \wedge \neg n_j \qquad \text{(by simplification } \alpha\text{) (1)}$$
$$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc\mathtt{K}_i(n_i \succ n_k) \qquad \text{(by 1, } \rho_{S3,i}\text{) (2)}$$
$$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc\mathtt{K}_k(u \prec n_k) \qquad \text{(by 1, } \rho_{S4,k}\text{) (3)}$$
$$\langle r,t\rangle \models \bigcirc^{f}\mathtt{K}_i(n_i \succ n_k) \qquad \text{(by AS6,2) (4)}$$

$$\langle r,t\rangle \models \bigcirc^f \mathtt{K}_k(u \prec n_k) \qquad \text{(by AS7, 3) (5)}$$

$$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc^f \mathtt{K}_k(n_i \prec n_k) \qquad \text{(by MP, 4,5, } \rho_{S2,i,k}\text{) (6)}$$

$$\langle r,t\rangle \models \bigcirc^{2f} \mathtt{K}_k(n_i \prec n_k) \qquad \text{(by AS7, 6) (7)}$$

$$\langle r,t\rangle \models \bigcirc^{2f} \mathtt{K}_i(n_i \succ n_k) \qquad \text{(by AS6, 4) (8)}$$

$$\langle r,t\rangle \models \bigcirc^{2f} (\mathtt{K}_k(n_i \prec n_k) \wedge \mathtt{K}_i(n_i \succ n_k)) \qquad \text{(by TP, 7,8) (9)}$$

$$\langle r,t\rangle \models \circ^{2f} (n_i \Cap n_k) \qquad \text{(by definition of } \Cap\text{) (10)}$$

The last formula represents the statement of this Lemma.

**Bojan Marinković** is Research Assistant Professor at Mathematical Institute of the Serbian Academy of Sciences and Arts. He received his PhD student at The Faculty of Techical Sciences University of Novi Sad, Serbia in 2014. During 2009, he spent three months as a visiting researcher at INRIA Sophia Antipolis, France. His research interests concern: distributed systems, applications of mathematical logic in computer science, automated theorem proving and digitization of cultural and scientific heritage.

**Zoran Ognjanović** is a research professor at the Mathematical Institute of the Serbian Academy of Sciences and Arts. He received his PhD degree in mathematical logic from University of Kragujevac, Serbia, in 1999. He has authored or coauthored over 70 (chapters of) monographs and technical papers in major international journals and conferences. His research interests concern: applications of mathematical logic in computer science, artificial intelligence and uncertain reasoning, automated theorem proving, applications of heuristics to satisfiability problem and digitization of cultural and scientific heritage. He is a recipient of the Serbian Academy of Sciences and Arts Award in the field of mathematics and related sciences for 2013 and the annual award of Serbian Ministry of Science for results in fundamental research in 2004.

**Paola Glavan**, PhD, is currently research and teaching assistant at the University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture-FSB. Her main scientific interests include using logic and logical methods in analyzing distributed processes and protocols. In particular, she is interested in Abstract State Machines and their application to semantics of programming languages and distributed protocols and the use of temporal epistemic logic in describing and verifying distributed protocols.

**Anton Kos** received his Ph.D. in electrical engineering from University of Ljubljana, Slovenia, in 2006. He is an assistant professor at the Faculty of Electrical Engineering, University of Ljubljana. He is a member of the Laboratory of Information Technologies at the Department of Communication and Information Technologies. His teaching and research work includes communication networks and protocols, quality of service, dataflow computing and applications, usage of inertial sensors in biofeedback systems and applications, signal processing, and information systems. He is the (co)author of more than thirty papers appeared in the international engineering journals and of more than fifty papers presented at international conferences.

**Anton Umek** received his Ph.D. in electrical engineering from University of Ljubljana, Slovenia, in 1999. He is currently an assistant professor at the Faculty of Electrical Engineering, University of Ljubljana. He is a member of the Laboratory of Information Technologies at the Department of Communication and Information Technologies. He is a member of the research program Algorithms and optimization methods in telecommunications that was two years in a row the best research program financed by the Slovenian research agency. Since last year he is the leader of industrial research and development projects in designing of sensor based smart sport equipment and sensor based forestry machinery. His teaching and research work includes signal processing, digital communication, secure communications , access network technologies and design of sensor supported sport training systems. He is the (co)author of eight papers appeared in the international engineering journals and of more than thirty papers presented at international conferences. Anton Umek is a member of IEEE and between 2015 and 2018 he was the Slovenian section ComSOC chapter chair.

*Received: November 15, 2018; Accepted: August 8, 2019.*

Computer Science and Information Systems 17(1):161–179 https://doi.org/10.2298/CSIS181011020K

# Distance Transform and Template Matching Based Methods for Localization of Barcodes and QR Codes

Melinda Katona, Péter Bodnár and László G. Nyúl

Department of Image Processing and Computer Graphics
University of Szeged
Árpád tér 2., H-6720 Szeged, Hungary
{mkatona, bodnaar, nyul}@inf.u-szeged.hu

**Abstract.** Visual codes play an important role in automatic identification, which became an inseparable part of industrial processes. Thanks to the revolution of smartphones and telecommunication, it also becomes more and more popular in everyday life, containing embedded web addresses or other small informative texts. While barcode reading is straightforward in images having optimal parameters (focus, illumination, code orientation, and position), localization of code regions is still challenging in many scenarios. Every setup has its own characteristics, therefore many approaches are justifiable. Industrial applications are likely to have more fixed parameters like illumination, camera type and code size, and processing speed and accuracy are the most important requirements. In everyday use, like with smartphone cameras, a wide variety of code types, sizes, noise levels and blurring can be observed, but the processing speed is often not crucial, and the image acquisition process can be repeated in order for successful detection.
In this paper, we address this problem with two novel methods for localization of 1D barcodes based on template matching and distance transformation, and a third method to detect QR codes. Our proposed approaches can simultaneously localize several different types of codes. We compare the effectiveness of the proposed methods with several approaches from the literature using public databases and a large set of synthetic images as a benchmark. The evaluation shows that the proposed methods are efficient, having 84.3 % Jaccard accuracy, superior to other approaches. One of the presented approaches is an improvement on our previous work. Our template matching based method is computationally more complex, however, it can be adapted to specific code types providing high accuracy. The other method uses distance transformation, which is fast and gives rough regions of interests that can contain valid visual code candidates.

**Keywords:** barcode localization, QR code localization, feature extraction, distance transform, template matching.

## 1. Introduction

Item identification using visual codes is popular in our everyday life, and there are several methods available for the process to be fast and reliable. The retrieval of the embedded data takes place in two steps. First, we have to find the visual code object within the acquired sensor data or image (localization step), then we have to use the symbology of the code and recognize the embedded data (decoding step). Decoding is widely studied, so we can use many approaches from the literature [8,10,19,23,27], or public APIs like the ZBar library[1].

It should be emphasized that decoding is far more straightforward, while the issue of localization is similar to object recognition and is still not fully solved. For localization of the code object, most algorithms use segmentation techniques with different features. Several applications simply ignore the localization step by adding a fast rotating laser that scans in many directions. Also, false positives are not acceptable, but the checksum digit (barcode) and the error correction (QR code) make false positives very unlikely in practice.

Visual codes are not meant to be readable for humans, they are decoded by specific devices. The most popular 1D barcode subtypes are the EAN-13 and UPC standards. These are widely used in commerce, like on wrapping of products, and they help quickly obtain the information on e.g. the producing country, types of entities of products. The flow of information is greatly boosted using visual codes, which provide decoding of the embedded data by electronic devices. Some types include features that also help their localization. The traditional 1D barcode structure is simple: a sequence of parallel light and dark bars of varying thickness represent information. The literature sometimes refers to 2D codes as "barcodes", however, they do not necessarily consist of "bars". They carry the embedded data along two axes, and their most popular types are QR code and Data Matrix. Some 1D and 2D codes are presented in Fig. 1. In addition, these classical visual codes can also be produced in a way that they become unique and thus can be used to validate originality or authenticity. For example, in our previous work [14], we focused on automatic localization of glitters used as a certain kind of Natural Feature Identifier (NFI).

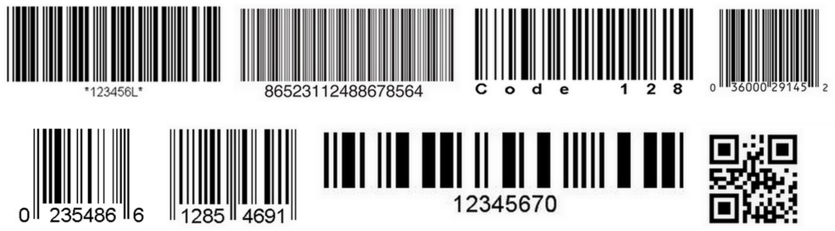


**Fig. 1.** Popular barcode types (from left to right). Top row (1D codes): Code39, Codabar, Code128, UPC-A; Bottom row (1D codes, 2D code): UPC-E, EAN-8, I2of5, QR code.

The use of visual codes has a reputation of more than 50 years, however, in the past, the localization process required many conditions to fulfill. The first barcodes were in a fixed position on railway trucks and were read by a fixed sensor gate. As technology progressed, PoS terminals appeared, still requiring human intervention to perform code reading. In the '70s, new algorithms have been developed that could localize codes having various orientation and position within the image. The first approaches were very simplistic, they imitated the laser scanners of the barcode reading device. From the '90s, machine

[1] publicly available at http://zbar.sourceforge.net/

learning provided some more sophisticated solutions for the issue. Methods providing automatic code localization are usually slower, but more accurate than their predecessors. Accuracy and processing speed are conditions that can hardly be fulfilled simultaneously, and most approaches aim to find a balance between these. Some machine learning algorithms make an exception, and they are capable of a quick evaluation after a significantly slower learning process, provided that the features can be computed efficiently and there is sufficient amount of training data available.

In industrial applications, accuracy is more crucial, since missed codes may lead to loss of profit. In those cases, speed is a second desired attribute, while in smartphone applications accuracy is not as critical, because the user handles the device interactively, and repetition of the image acquisition is possible and relatively easy.

There are numerous methods for the localization of visual codes in digital images, some imitating the classical laser scanner. Adelmann et al. [1] introduced a barcode recognition and information system to detect and read EAN-13 barcodes. This system works as a mobile application and traditional and widely used. After some preprocessing steps, Ohbuchi et al. [18] used a scanline based procedure to detect QR and EAN codes.

The toolkit of mathematical morphology has been used in many approaches in the literature. Bodnár et al. explained that using simple detectors [4] such as combination of different morphological operators and distance map to detect barcodes efficiently. Furthermore, texture analysis [3] can also achieve great efficiency. Similarly, Katona et al. [12] showed a method that relies on simple morphological bottom-hat filtering after a pre-processing step that highlights the bars of the barcode. Later in their work [13], they used simple features to localize a barcode areas. A distance map based approach was used as an extension of this process to merge split regions, which improves accuracy. Those regions, for example, arise from bad illumination, or flaws of the barcode material. Lin et al. [17] demonstrated a fast and effective method that can simultaneously detect 1D and 2D barcodes. Their method is based on a modified run length smearing algorithm. Kong [15] defines regions of interests that may contain QR codes in synthetic images with the mix of Harris corner detector and convex hull. In addition, they recommended a solution to correct for geometric distortion. Belussi et al. [2] introduced a machine learning method that is based on the locator pattern of the QR code. They proposed a cascade of weak classifiers using features from the Haar wavelet family. Although it is fast, it provides a noticeable amount of false positive code candidates. Bodnár et al. [6] proposed an improvement on that, using LBP and HoG features as an extension of the training step on the full code object. Sörös et al. [20] aim to localize 1D and 2D code using edge and corner maps, even considering the saturation channel in HSV images. Their algorithm is optimized on images suffering from heavy directional smoothing [21]. The method has high accuracy, in cases however, where the code object is surrounded by text, their approach provides oversized bounding boxes. Text filtering can help get rid of this problem, considering the surrounding text as a priori information. Szentandrási et al. [22] also work with edge and corner maps and HoG features. Their method works locally on square image cells, similarly to convolution. This approach enables parallel execution and it is also highly accurate.

Yun et al. [28] introduced an orientation histogram-based method. They used a histogram to the principal orientation components from the entire image and calculate the local entropy of the orientation to generate a saliency map. Bodnár et al. [5] presented a method based on distance transformation. The algorithm also considers local image blocks and evaluates the distance map of the edge map. It takes into account the mean and standard deviation of the distance values within each block, then makes a binary decision whether or not the block contains a barcode part. While this feature can be computed efficiently, it has weak classification power, therefore it is not sufficient for use alone for the localization step. In their work, the authors tried to overcome this attribute using morphological operations.

Many recent papers use machine learning methods to solve various image processing problems. Hansen et al. [11] used a deep learning object detection algorithm, namely You Look Only Once (YOLO) model. Their network is based on a pre-trained Darknet19 model with 6000 epochs. The most common architecture for semantic segmentation is the U-net that has different variants for each task. Ventsov et al. [26] divided the input image into $128\times128$ blocks, extracted statistical characteristics for each block and trained a convolution neural network. A Region-based Convolutional Neural Network (R-CNN) model was proposed by Ban et al. [25] for detecting diversified barcodes under complex scenes. They used for experiments two pre-trained model, ImageNet and VGG16.

In this paper, in Sec. 2.2, we present an improvement to this latter approach, giving a feature that can also be computed using the distance map and it also considers direction information. Also, instead of using only statistical values, we propose to use the whole distribution vector and make the final decision with SVM. The feature is computed locally, and in the final step, the accuracy is further improved by processing the feature matrix. This shows good performance for 1D codes, but on 2D codes, it is not sufficiently accurate. We also present two algorithms based on template matching, one suitable for efficiently localizing 1D barcodes (Sec. 2.1), and the other suitable for QR code detection (Sec. 2.3).

## 2. Methods

Although the imaging quality of recent digital cameras is high, lower quality images may be acquired as well due to various circumstances, such as dust, humidity, shaking of the camera in low-light situations. Due to this, preprocessing of the input images is usually necessary before code localization. In this section, we present three barcode detection algorithms. They use different classical operations to find the barcode in the image.

### 2.1. 1D barcode localization using pattern matching

In this section, we present a novel method for the localization of 1D barcodes based on pattern matching. The overview of the algorithm is presented in Fig. 2, while particular steps are illustrated in Fig. 3 and described below.

During processing, the input images can be of different sizes. We reduce the height of the image to a fixed size of 500 pixels in order to make them more uniform, easier to handle, and make processing faster. Empirical experience has shown that this is the smallest image size where smaller area code regions can still be localized. We did not use color information during the process, so the input RGB image was converted to grayscale. Input images are often blurry, therefore we use sharpening (Fig. 3(b)).

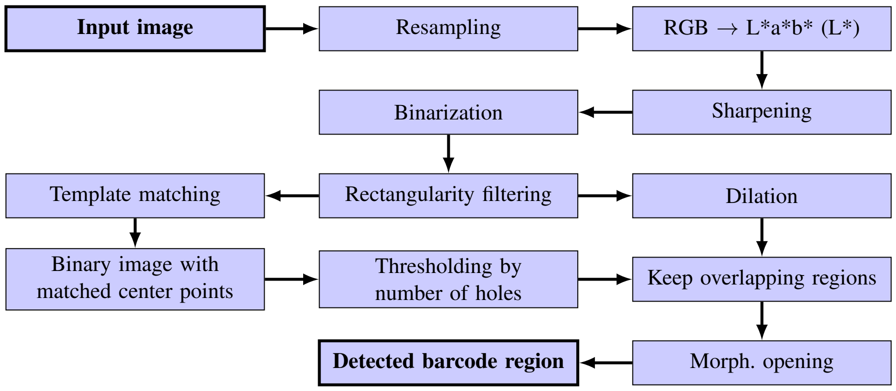


**Fig. 2.** The barcode localization process using template matching

The detection process is based on binary images, so the image is binarized using a global threshold. In our case, this value was 4% of the maximum intensity (Fig. 3(c)). This threshold was chosen empirically, based on the observation that white parts of the code more often fall into the gray intensity range because of dust, cheap quality labels or shapes being present because the packaging does not necessarily use white color for the bright parts of the code. This is not a robust solution, but selecting valid regions from false regions is easier than finding a missing part during a post-processing process. Obviously, a filter step is needed to reduce the number of false regions where different noise, etc., can occur. The shape of the bars of a barcode are rectangular, so we examine the shape of each object. If the shape of the object is not approximately rectangular, we do not consider it as a candidate region. The examined barcodes have a specific structure, so template matching is a possible way to detect the bars of the barcode. The input for pattern matching is illustrated in Fig. 3(d). As barcodes consist of parallel "bars", the template consists of two parallel lines. Traditional barcodes consist of a plurality of parallel lines, so a similar part of the image may be suitable for template matching. We also know the maximum and minimum distance between bars for each type of code. The template image was selected based on this information.

The process of template matching occurs in the frequency domain using Fast Fourier Transformation. The complexity of template matching in Fourier domain is $\mathcal{O}(n \log n * n_2)$, where $n$ is the data size. We rotate the template image in every $10°$, up to $170°$, and compute the sum of the pointwise multiplication of the frequency representation of the original image and the rotated template images. $10°$ step was empirically found to be sufficient because the efficiency of this method had its maximum at around $15°$. Thanks to the symmetric nature of the matched template, it is sufficient to examine only the aforementioned rotations. The summarized feature image is then thresholded with the mean of the summarized value.

Next, we use only the center of the objects that are being obtained. Pixels belonging to a specific cluster are well-separable like the bars of the barcode that are close to each other. We used the well-known kNN clustering method to separate connected objects from each other. The set of points from template matching can be well separated, so we have

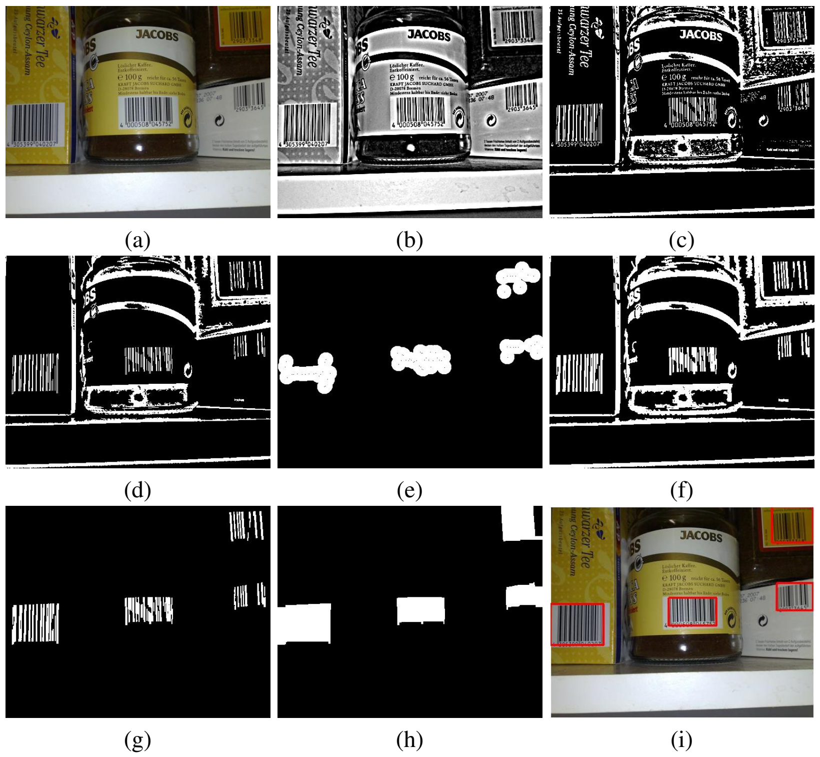


**Fig. 3.** The proposed method for 1D barcode localization. (a) input image, (b) deblurring, (c) binarization, (d) filtering by rectangularity, (e) thresholding using the number of cluster points, (f) result of (d) after dilation, (g) matching objects with (e) on image (f), (h) opening, (i) code candidate boxes overlaid onto (a) in red

chosen $k = 3$. To keep valid regions, we use a priori information that a barcode consists of at least 8 bars, so only those clusters are kept that have at least 8 points (3(e)). The complexity of kNN is $\mathcal{O}(ndk)n$, where $n$ denotes the number of training points, $d$ is the number of dimensions and $k$ is the number of iterations.

We apply morphological dilation on the binary version of the original image, using a 3×3 structuring element (Fig. 3(f)). We investigate objects between the dilated image and filtered cluster points and we only keep overlapping regions (Fig. 3(g)). In order to determine the whole barcode region, we use morphological opening with a square-shaped structuring element. The size is defined based on the maximal distance between the stripes, which provides that every barcode will have its own connected region (Fig. 3(h)). The complexity of morphological operations depends on the used operator.

## 2.2. Barcode localization using distance transformation

The proposed algorithm is based on a feature derived from distance transformation. First, the Canny edge map is produced, then distance transformation is performed, where every point gets the distance value from its closest edge point. During this computation, we propose to also register the angle of each corresponding edge point, which reinforces the feature.

A priori information is needed regarding the element size of the visual code. “Bar thickness” of the barcodes may vary with respect to the distance of the code object and the camera, however, a range can be given for the expected thickness of the bars. Typically, the thinnest bar of a barcode is 1–3 px, while the thickest is 5–15 px wide in a case where the data is reasonably retrievable. With QR codes, typical element size is 5–30 px, regarding public code databases.

We propose to work on the brightness channel of the color space (V in HSV or L* in L*a*b*). As a preprocessing step, contrast stretching is performed and Gauss smoothing is applied, with a kernel size depending on the image resolution and expected bar thickness. For the aforementioned case, a $3\times3$ or $5\times5$ kernel is appropriate. After smoothing, the Canny edge map is produced as the hysteresis thresholding of the Sobel’s $x$ and $y$ gradients. The method greatly helps localization because it produces thin, connected edges. Those edge points are the marked points for the distance transformation. We also calculate the direction to the closest corresponding point. This approach produces similar “zones” like the Voronoi diagram (Fig. 4).

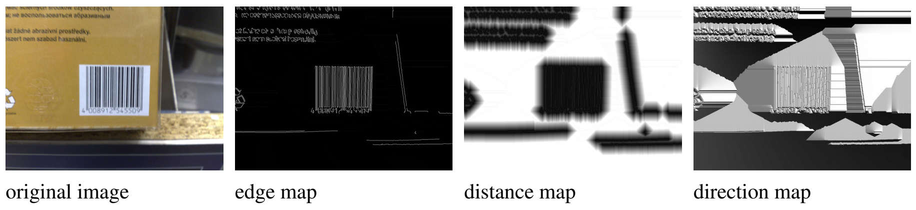


**Fig. 4.** Sample from the Muenster data set and its corresponding feature images

The feature image is divided into disjoint square blocks as the next step, then we calculate the distribution of the distance and angle values, aggregated in a predefined number of bins. The two vectors are then concatenated and fed into an SVM that learns a binary classification. Although we could give the raw pixel data to the SVM, it is less efficient than the aforementioned features that take advantage of spatial information. The prediction of the SVM will give an answer to the question of whether or not an image block contains part of a barcode. Such binary value is assigned to each block forming a feature matrix.

In the next step, connected components of the feature image are determined. Components are filtered by size, as barcode bar width gives a range for expected minimum and maximum code size. Those code objects appear as connected components in the feature matrix. Compactness can be calculated as the proportion of the perimeter and area of a blob. The tolerance should be set according to the expected visual code type (bar width and the width-to-height ratio of the specific code). Fig. 5 shows an example for the feature image, its thresholded connected components and filtering by size and compactness.

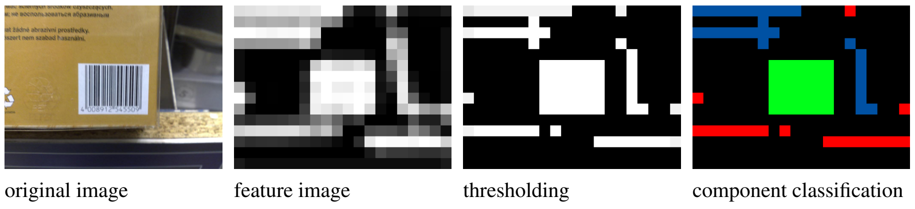


**Fig. 5.** Post-processing steps of the feature image

A rotated bounding rectangle is given for the components that meet the aforementioned conditions. After that, we look for a homography with a properly oriented rectangle having the expected code size and proportions. The decoder gets the rectangular area from the image with the inverse homography applied.

In a previous work [5] the distance transformation was performed block-wise, which means the closest marked point was only searched for within the block. It is more appropriate to do the distance transformation before the tiling because we can also find the closest corresponding points in neighboring blocks and this helps the training process of the SVM. Block size is not relevant for the distance feature itself, however, it should be selected so that most bins of the distribution contain a sufficient number of samples. The rule of thumb for binning declares that $n$ is a proper choice for the number of bins if we have at least $n^2$ samples. According to that, a distribution of 16 bins defines a lower bound of $16\times16$ px block size. The upper bound for the block size is related to the code size. In order to successfully detect a code object, at least 15–20 blocks are needed in the feature matrix for a code candidate. Fewer blocks would mean block length being bigger than 25–35 % of the code length along its dimensions, which decreases the occurrence of blocks that are full with code pattern only.

Blocks can be overlapping, but overlapping does not significantly improve the variability (and the learnability) of the distribution, and approximating the process of convolution only increases running time. Summary of the steps can be observed in Fig. 6.

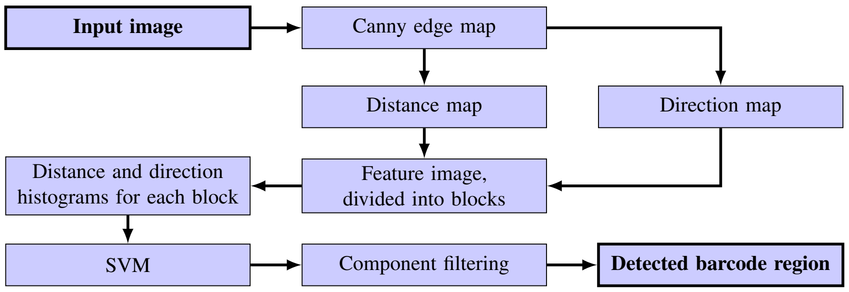


**Fig. 6.** Steps of the Distance Transform approach

### 2.3. QR code localization with template matching

In this section, we present a new method for localization of QR codes. The overview is given in Fig. 7, while the particular steps are illustrated in Fig. 8 and described below.

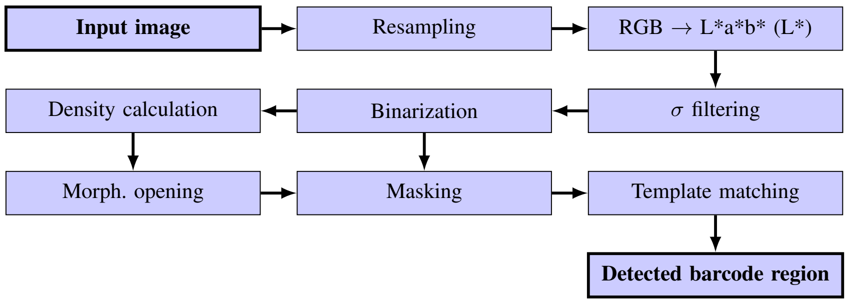


**Fig. 7.** The QR code localization process

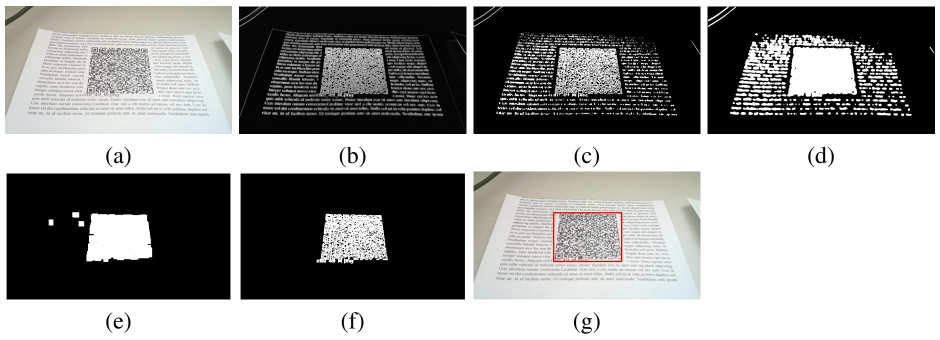


**Fig. 8.** Proposed method for QR code localization. (a) input image, (b) $\sigma$ filtering, (c) binarization, (d) density calculation, (e) morphological opening, (f) post-processing, (g) detected QR code

Similarly to the procedure described in Sec. 2.2, we also work with images with specific size during the localization and convert the images to grayscale. In order to highlight the barcode areas, we use standard deviation based adaptive filtering method with $3\times3$ neighborhood [9]. The resulting image is heterogeneous, so we used density calculation with a fixed $7\times7$ kernel. We calculated the number of object points for every kernel and removed from candidate barcode regions where this value was under the half of the kernel size.

In order to remove false small regions and then merge the connected regions, we apply a morphological opening. The shape of the QR code is ideally a square, so we use a square shaped structuring element for the morphological operation. Based on empirical observations, we binarize the image obtained in the previous step with the threshold value of 7/8th of the maximum intensity. Since global thresholding is not an overly robust operation, but the bars of the barcode have low-intensity as usual (we supposed that barcode not colorful), so we can eliminate numerous false segments with a low-intensity value when determining global thresholding. In the last step, we validate the code segments by pattern matching. For this, we used a region from the inner box of a QR code as a sample. We do the template matching on the original image and we investigate overlapping with the opened binary image similarly as described in Sec. 2.2. Valid QR code regions are available after the validation step.

## 3. Evaluation and results

In this section, the proposed algorithms are compared against some effective ones from the literature. Several research groups [8,7,12,20,24,29,28,11,26] evaluated their algorithms on the WWU Muenster data set, therefore we also decided to use that set for evaluation, while the fine-tuning of parameters and some of our other tests were performed on our custom, synthetic image set[2].

### 3.1. Test suite and implementation

An artificial test set is created from the barcode examples presented in Fig. 1. One example is selected from each code type, and various distortions and levels of noise are applied. The generated images were rotated from $0°$ to $180°$ by $15°$. Gaussian smoothing is applied with $3{\times}3$ kernel and 6 different $\sigma$ values. Also, Gaussian noise was added from $0\,\%$ to $50\,\%$ with the step of $10\,\%$. In total, we created 12 orientations from 8 types of barcodes, using 6 different smoothing and 6 different noise levels, with perspective distortions, counting as cca. 15 000 images. Fig. 9 illustrates some examples from our artificial data set. Furthermore, we used 1056 images of real barcodes from the WWU Muenster data set.

For the test set of QR codes, we used a database consisting of 1400 real images [20], and 10 000 synthetic test images. The latter set is generated similarly to the 1D barcode set. Fig. 10 shows some samples from that data set. The second public database by Dubská et al. contained two similar sets of QR code images, surrounded with text in a scene having low saturation in general. The first set has 410 high-resolution ($2560 \times 1440$ px) images with uneven lighting conditions, high grades of distortion and minor blur. The second test set has 400 low-resolution ($604 \times 402$ px) images with smaller grades of distortion and more even illumination, but having less light in general, thus producing darker images.

### 3.2. Figures of merit

To measure the efficiency of the algorithms, we compared the overlap of our segmentation output and the ground truth with the Jaccard similarity measure, defined as

$$J = \frac{TP}{TP + FP + FN},$$

[2] `http://www.inf.u-szeged.hu/~bodnaar/barcode_database/`

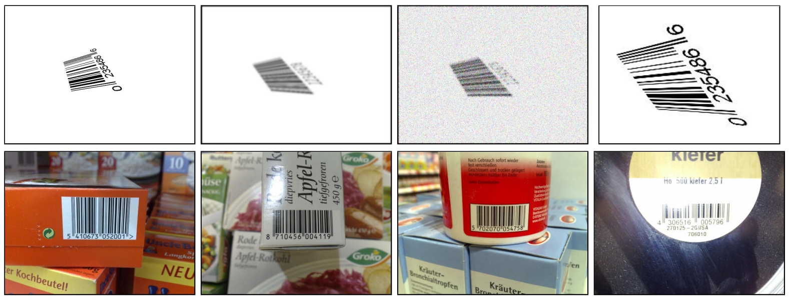


**Fig. 9.** 1D samples with different distortions, synthetic (first row) and real images (second row).

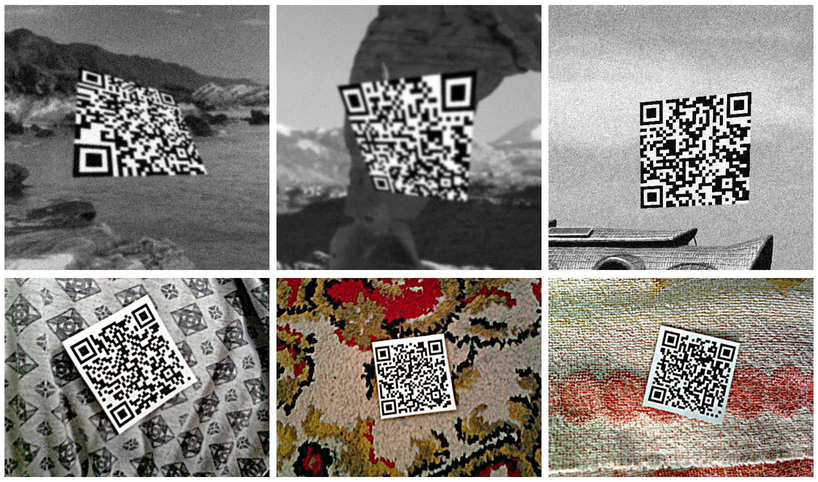


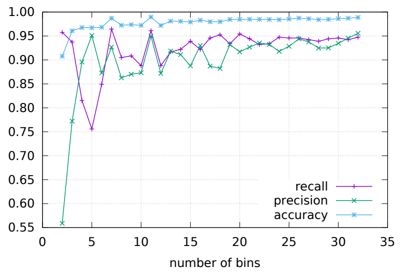
**Fig. 10.** Synthetic and real images with QR code

where $TP$ denotes the correctly detected codes, $FP$ is the number of not valid code regions and $FN$ is the number of not localized codes. Note that, ground truth regions have a tight fitting bounding polygon showing the code object without numbers and "quiet zones" as a border. A successful detection is where $J > 0.5$, according to the work of Szentandrási et al. [22].

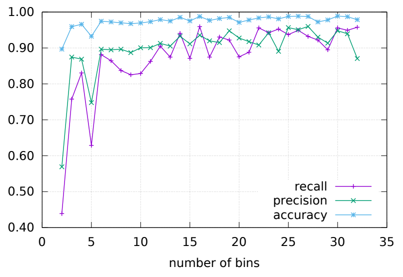
## 3.3. Parameters for the distance transformation approach

The fine-tuning of the SVM parameters were performed using a subset of the Muenster database since with 1D cases, the visual structure of the code and the distribution of the angles is more prominent.

As the first step of the evaluation, we examined how SVM accuracy is influenced by the number of bins. We performed separate trainings using only the distributions of

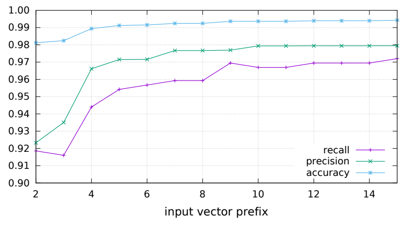


(a)

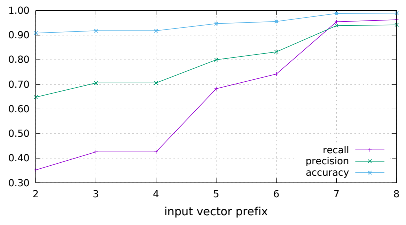


(b)

**Fig. 11.** Efficiency of the Distance Transformation method w.r.t. the number of distance (a) and direction (b) bins



(a)



(b)

**Fig. 12.** SVM efficiency w.r.t. the used prefix of the distance (a) and direction (b) histograms

distance values (Fig. 11(a)) and angles (Fig. 11(b)). We can conclude that more than 16 distance classes do not significantly improve accuracy. With angles, 8 classes show the highest classification power.

Also, we examined feeding only a prefix of a distribution of distances or angles, which shows how important the individual bins are. Intuitively, we shall expect that for the distance values, the first few bins are important, because the edge points are close to each other in barcodes, and this characteristic contributes the most to the classification power. Our results confirm this assumption (Fig. 12(a)).

For the directions we shall expect that the elements of the distribution are equally important, so the number of bins is linearly correlated to accuracy (Fig. 12(b)).

We examined the accuracy considering only distance values, only angles, and both distance and angle simultaneously. Results are shown in Table 1. The highest accuracy is obtained when both directions and angles are used for the training.

We also evaluated the training accuracy using a small number of training samples. Results are shown in Fig. 13. The whole Muenster database with a given block size of 50×50 px contains cca. 300 000 input vectors, about 10 % of them labeled as positive. The SVM classes should be weighted according to that proportion. We used 10-fold cross-

**Table 1.** SVM training performance for various features

| input | recall | precision | accuracy |
|---|---|---|---|
| directions only | 0.9625 | 0.9416 | 0.9893 |
| distances only | 0.8169 | 0.8978 | 0.9698 |
| both dir. & dist. | 0.9597 | 0.9871 | 0.9942 |

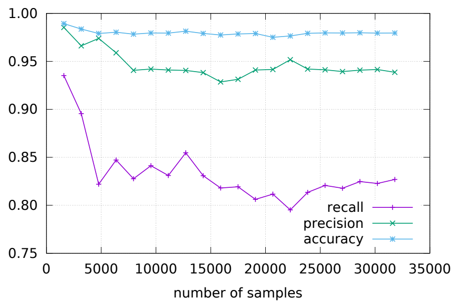


**Fig. 13.** SVM efficiency w.r.t. the number of used training samples

validation for validation purposes. The decrease in accuracy at the beginning of the graph indicates that a small number of samples are more separable.

We implemented the algorithm using OpenCV, with the automatic SVM optimizer option. Optimal parameter set means that the error is minimal during the cross-validation. Additionally, it shall be noted that considering the directions along with the distances does not increase running time significantly, because, during the 2-pass calculation of distance values, the directions can also be recorded. Regarding memory usage, the space needed to store the angles is similar to that for the original image.

## 3.4. Comparison

Our approaches were compared against other algorithms of the literature using the Muenster database as a benchmark. Results are shown in Table 2. Considering the proposed method with distance transformation (PROP-DT, Sec. 2.2), it shall be noted that the earlier method based only on distance values can only be used as a weak classifier, but this improvement with the directions and the SVM trained on the whole distribution makes it a usable state-of-the-art solution. Only the algorithm of Tekin el al. [24] has better mean value, however, with higher variance. For the comparison, we selected algorithms based on various features, like edge and corner maps [20], or deformable templates [8]. Zamberletti et al. [29] work with the popular localization method that is also implemented in ZXing barcode reading framework, while Creusot et al. [7] have an approach that uses MSER. Our template matching based approach (PROP-TM, Sec. 2.1) is even more specific to the barcode localization issue, therefore it shows even better performance than PROP-DT. Hansen et al. [11] reported 0.87 Jaccard value to result in their article, but we did not use this result in our comparison, because there is some missing information about training parameters, so we were unable to re-implement and evaluate the procedure.

**Table 2.** Comparison of various localization algorithms on the Muenster data set. (Mean and standard deviation of Jaccard index.) Best performing method is typeset in bold.

| Algorithm | $\overline{J}$ | st.dev. |
|---|---|---|
| Zamberletti et al. [29] | 0.6950 | N/A |
| Creusot et al. [7] | 0.7990 | N/A |
| Gallo et al. [8] | 0.7089 | 0.3542 |
| Tekin et al. [24] | 0.8122 | 0.2562 |
| Katona et al. [12] | 0.5200 | 0.2967 |
| Sörös et al. [20] | 0.6647 | 0.2277 |
| Yun et al. [28] | 0.4716 | 0.2240 |
| PROP-DT (Sec. 2.2) | 0.8104 | 0.1944 |
| **PROP-TM** (Sec. 2.1) | **0.8430** | **0.1876** |

Most of the presented methods have the main goal as to highlight the bars of the barcode (gradiens calculation, bottom-hat filtering, etc.). The algorithms that are based on simple image operations and logic are examined on the synthetic image database under different conditions. We investigated and found that the procedures are not sensitive to rotation. We also examined the results with different noise and blur levels. In both cases the generated images had 6 different levels for those parameters, modifying the $\sigma$ value of Gaussian smoothing and a $\gamma$ weight for weighted addition of uniform noise to the original image. We present the obtained Jaccard indexes in Table 3. It shows that the efficiency of the procedures is hardly reduced by increasing the level of noise and smoothing. However, in all cases, the methods do not behave significantly differently on ideal cases. We used 8 different codes to generate the images. The procedures were less successful in localization task for these three types (Code-39, I2of5, UPC-E) as shown in Table 4. We also present some qualitative results of the aforementioned approaches on challenging images. See 14 for details.

In the 2D case, we evaluated PROP-DT on the two data sets of Dubská et al. [22], and obtained $\overline{J}_{set2} = 0.7588$ and $\overline{J}_{set1} = 0.5592$ as shown in Table 5. This means that distance transformation cannot be used for 2D code localization because of the low level of separability of distance and angle values. However, our template matching approach (PROP-TMQR, Sec. 2.3) with 2D QR codes performed well on the Dubská data sets, with $\overline{J}_{set1} = 0.8315$ and $\overline{J}_{set2} = 0.8102$. Hansen et al. [11] published 0.73 average Jaccard value on Dubská sets.

We also compared the results on synthetic images along different blur and noise levels (Table 6).

**Table 3.** Accuracy of the methods for different blur ($\sigma$ value) and noise levels (in percent) on 1D synthetic images (mean Jaccard index).

| Blur | Noise | Gallo [8] | Yun [28] | Sörös [20] | PROP-DT | PROP-TM |
|---|---|---|---|---|---|---|
| | 0 | 0.74 | 0.64 | 0.75 | 0.83 | 0.82 |
| 0 | 30 | 0.75 | 0.66 | 0.76 | 0.83 | 0.82 |
| | 50 | 0.75 | 0.65 | 0.77 | 0.83 | 0.81 |
| | 0 | 0.77 | 0.64 | 0.74 | 0.75 | 0.82 |
| 3 | 30 | 0.75 | 0.67 | 0.76 | 0.83 | 0.81 |
| | 50 | 0.73 | 0.63 | 0.76 | 0.77 | 0.78 |
| | 0 | 0.76 | 0.65 | 0.74 | 0.73 | 0.82 |
| 5 | 30 | 0.74 | 0.66 | 0.76 | 0.83 | 0.81 |
| | 50 | 0.73 | 0.63 | 0.75 | 0.77 | 0.78 |

**Table 4.** Accuracy of the algorithms for various types of code on 1D synthetic images (mean Jaccard index).

| | Gallo [8] | Yun [28] | Sörös [20] | PROP-DT | PROP-TM |
|---|---|---|---|---|---|
| Codabar | 0.81 | 0.71 | 0.8 | 0.73 | 0.87 |
| Code-128 | 0.82 | 0.73 | 0.83 | 0.78 | 0.92 |
| Code-39 | 0.57 | 0.52 | 0.59 | 0.84 | 0.72 |
| Ean-13 | 0.82 | 0.69 | 0.81 | 0.85 | 0.83 |
| EAN-8 | 0.81 | 0.70 | 0.81 | 0.91 | 0.83 |
| I2of5 | 0.69 | 0.6 | 0.69 | 0.77 | 0.72 |
| UPC-A | 0.78 | 0.66 | 0.78 | 0.84 | 0.80 |
| UPC-E | 0.69 | 0.64 | 0.74 | 0.74 | 0.73 |

**Table 5.** Comparison of various localization algorithms on the QR synthetic database (mean Jaccard index).

| input | Ohbuchi [18] | Lin [16] | PROP-DT | PROP-TMQR |
|---|---|---|---|---|
| Dubska set1 | 0.79 | 0.81 | 0.56 | 0.83 |
| Dubska set2 | 0.77 | 0.79 | 0.76 | 0.81 |

**Table 6.** Accuracy of the methods for different blur ($\sigma$ value) and noise levels (in percent) on QR synthetic images (mean Jaccard index).

| Blur | Noise | Ohbuchi [18] | Lin [16] | PROP-DT | PROP-TM |
|---|---|---|---|---|---|
| | 0 | 0.96 | 0.50 | 0.77 | 0.99 |
| 0 | 50 | 0.96 | 0.50 | 0.77 | 0.99 |
| | 100 | 0.96 | 0.50 | 0.77 | 0.99 |
| | 0 | 0.57 | 0.51 | 0.67 | 0.77 |
| 1.5 | 50 | 0.52 | 0.49 | 0.67 | 0.78 |
| | 100 | 0.49 | 0.40 | 0.70 | 0.75 |
| | 0 | 0.47 | 0.42 | 0.65 | 0.83 |
| 3 | 50 | 0.48 | 0.45 | 0.65 | 0.80 |
| | 100 | 0.45 | 0.37 | 0.64 | 0.76 |

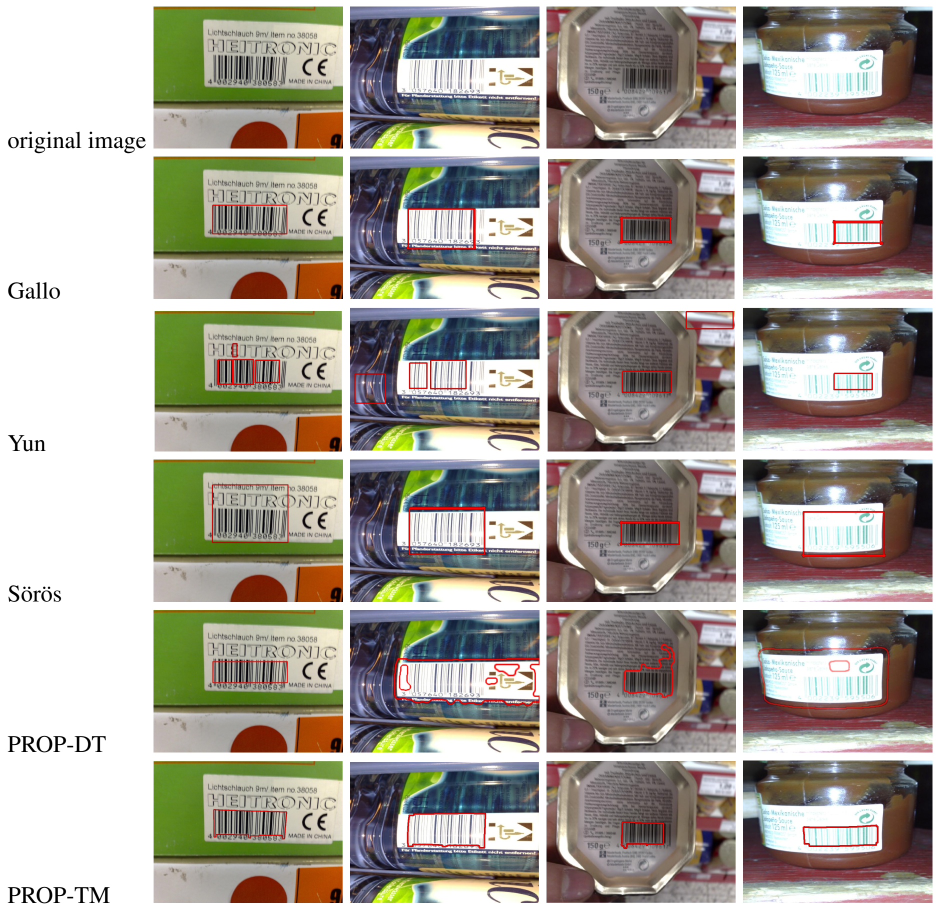


**Fig. 14.** Qualitative results for the compared methods on some challenging 1D images.

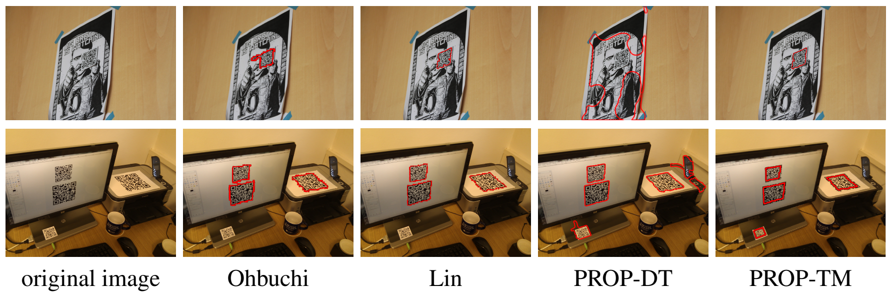


**Fig. 15.** Qualitative results for the compared methods on some challenging QR images.

## 4. Conclusion

Three novel approaches were presented, two for 1D barcode localization, and one for 2D QR codes. They are compared to algorithms from the literature, some of them being universal (working on both 1D and 2D codes), others specialized for either 1D or 2D. For the evaluation of efficiency, we generated data sets containing a large number of synthetic images. Results indicate that the proposed algorithms are efficient, even in cases where the visual codes suffer from perspective distortion.

Distance transformation was used for barcode localization in [5], however, it only could be considered as a weak classifier. Adding angle information to the feature, accuracy improves significantly. Distance transformation can be used as a standalone feature for the problem, with various code types. The approach also has a disadvantage, namely, it cannot be tuned to consider sophisticated structural details of different code types, like template matching.

In specific cases, when we can define more assumptions on the expected code type, size, or orientation, template matching can outperform general purpose solutions. However, in most applications, we cannot make very specific assumptions. Nevertheless, for some industrial applications, where the code properties fall in narrow ranges, the concept of template matching can be useful. In more general cases, the method based on distance transformation performs well, and it only contains computationally simple steps. Also, template matching might need different templates depending on the code type to find.

The proposed methods can be implemented to run in real time. The methods that use SVM and distance transformation take cca. 250 ms for an image of size 1024x768 px. The template matching based method's running time is 680 ms for an image in the case of 1D barcodes, and 419 ms for QR codes.

**Acknowledgments.**

This work was supported by the project “Integrated program for training new generation of scientists in the fields of computer science”, No. EFOP-3.6.3-VEKOP-16-2017-0002. The project has been supported by the European Union and co-funded by the European Social Fund. Melinda Katona was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP 4.2.4.A/2-11-1-2012-0001 ’National Excellence Program’. The authors would like to thank Dr. Attila Tanács for his detailed and helpful comments and suggestions.

## References

1. Adelmann, R., Langheinrich, M., Flörkemeier, C.: A toolkit for bar-code-recognition and -resolving on camera phones – jump starting the internet of things. In: In Workshop Mobile and Embedded Interactive Systems (MEIS'06) at Informatik (2006)
2. Belussi, L.F.F., Hirata, N.S.T.: Fast QR Code Detection in Arbitrarily Acquired Images. In: Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on. pp. 281–288 (2011)
3. Bodnár, P., Nyúl, L.G.: Barcode Detection with Morphological Operations and Clustering. In: Signal Processing, Pattern Recognition, and Applications, Proceedings of the Ninth IASTED International Conference on. pp. 51–57 (2012)
4. Bodnár, P., Nyúl, L.G.: Improving Barcode Detection with Combination of Simple Detectors. In: 8th International Conference on Signal Image Technology and Internet Based Systems, SITIS 2012. pp. 300–306 (2012)
5. Bodnár, P., Nyúl, L.G.: Barcode detection with uniform partitioning and distance transformation. IASTED International Conference on Computer Graphics and Imaging pp. 48–53 (2013)
6. Bodnár, P., Nyúl, L.G.: QR Code Localization Using Boosted Cascade of Weak Classifiers. In: Image Analysis and Recognition, pp. 338–345. Springer International Publishing (2014)
7. Creusot, C., Munawar, A.: Real-Time Barcode Detection in the Wild. In: 2015 IEEE Winter Conference on Applications of Computer Vision. pp. 239–245 (Jan 2015)
8. Gallo, O., Manduchi, R.: Reading 1D Barcodes with Mobile Phones Using Deformable Templates. IEEE Trans. Pattern Anal. Mach. Intell. 33(9), 1834–1843 (2011)
9. Gonzalez, R.C., Woods, R.E., Eddins, S.L.: Digital Image Processing Using MATLAB. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2003)
10. Ha, J.E.: A new method for detecting data matrix under similarity transform for machine vision applications. International Journal of Control, Automation and Systems 9, 737–741 (2011)
11. Hansen, D.K., Nasrollahi, K., Rasmussen, C.B., Moeslund, T.B.: Real-Time Barcode Detection and Classification using Deep Learning. In: IJCCI. pp. 321–327 (2017)
12. Katona, M., Nyúl, L.G.: A Novel Method for Accurate and Efficient Barcode Detection with Morphological Operations. In: Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on. pp. 307–314 (2012)
13. Katona, M., Nyúl, L.G.: Efficient 1D and 2D Barcode Detection Using Mathematical Morphology. In: Mathematical Morphology and Its Applications to Signal and Image Processing, Lecture Notes in Computer Science, vol. 7883, pp. 464–475. Springer Berlin Heidelberg (2013)
14. Katona, M., Nyúl, L.G.: Fast Recognition of Natural Feature Identifiers by a Mobile Phone. Acta Cybernetica 22(1), 101–116 (2015)
15. Kong, S.: QR Code Image Correction based on Corner Detection and Convex Hull Algorithm. Journal of Multimedia 8, 662–668 (2013)
16. Lin, D.T., Lin, C.L.: Multi-symbology and Multiple 1D/2D Barcodes Extraction Framework. In: Proceedings of the 17th International Conference on Advances in Multimedia Modeling - Volume Part II. pp. 401–410 (2011)
17. Lin, D.T., Lin, C.L.: Automatic location for multi-symbology and multiple 1D and 2D barcodes. Journal of Marine Science and Technology 21, 663–668 (2013)
18. Ohbuchi, E., Hanaizumi, H., Hock, L.A.: Barcode Readers Using the Camera Device in Mobile Phones. In: Proceedings of the 2004 International Conference on Cyberworlds. pp. 260–265. CW '04 (2004)
19. Shams, R., Sadeghi, P.: Bar Code Recognition in Highly Distorted and Low Resolution Images. In: Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on. vol. 1, pp. I–737 –I–740 (2007)
20. Sörös, G., Flörkemeier, C.: Blur-resistant Joint 1D and 2D Barcode Localization for Smartphones. In: Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia. pp. 11:1 – 11:8 (2013)

21. Sörös, G., Semmler, S., Humair, L., Hilliges, O.: Fast Blur Removal for Wearable QR Code Scanners. In: Proceedings of the 2015 ACM International Symposium on Wearable Computers. pp. 117–124. ISWC '15, ACM (2015)
22. Szentandrási, I., Herout, A., Dubská, M.: Fast Detection and Recognition of QR Codes in High-resolution Images. In: Proceedings of the 28th Spring Conference on Computer Graphics. pp. 129–136. SCCG '12 (2013)
23. Tekin, E., Coughlan, J.: A Bayesian Algorithm for Reading 1D Barcodes. In: Proceedings of the 2009 Canadian Conference on Computer and Robot Vision. pp. 61–67. CRV '09 (2009)
24. Tekin, E., Coughlan, J.: BLaDE: Barcode localization and decoding engine. Tech. rep., Technical Report 2012-RERC (2012)
25. Tian, Y., Che, Z., Zhai, G., Gao, Z.: BAN, A Barcode Accurate Detection Network. In: 2018 IEEE Visual Communications and Image Processing (VCIP). pp. 1–5 (2018)
26. Ventsov, N.N., Podkolzina, L.A.: Localization of Barcodes Using Artificial Neural Network. 2018 IEEE East-West Design & Test Symposium (EWDTS) pp. 1–6 (2018)
27. Wang, K., Zou, Y., Wang, H.: Bar code reading from images captured by camera phones. In: 2005 2nd International Conference on Mobile Technology, Applications and Systems. p. 6 (2005)
28. Yun, I., Kim, J.: Vision-based 1D barcode localization method for scale and rotation invariant. In: TENCON 2017 - 2017 IEEE Region 10 Conference. pp. 2204–2208 (2017)
29. Zamberletti, A., Gallo, I., Carullo, M., Binaghi, E.: Neural Image Restoration for Decoding 1-D Barcodes using Common Camera Phones. In: Computer Vision, Imaging and Computer Graphics. Theory and Applications. pp. 5–11 (2010)

**Melinda Katona** received the M.S. degree in Computer Science in 2014 at University of Szeged, Hungary. He is currently an PhD candidate at Institute of Informatics, University of Szeged. Her research interests include pattern recognition and medical imaging.

**Péter Bodnár** graduated at University of Szeged in 2011 as technical informatician. After that, he was admitted the PhD fellowship of the Doctoral School in Informatics. After the three years of PhD studies, he was awarded the predoctoral fellowship in 2014. He wrote his thesis about localization of visual codes. He participates in various projects of the Department of Image Processing and Computer Graphics at University of Szeged since 2011. Besides that, he has participated in the teaching activity of the Institute of Informatics.

**László G. Nyúl**, received the M.S. degree in Computer Science in 1994, and the Ph.D degree in 2003 from the University of Szeged, Hungary. He has been a visiting research associate at University College London, University of Pennsylvania, Universität Erlangen-Nürnberg, Medical University Graz, and Uppsala Universitet. He is currently an associate professor and the Head of Department of Image Processing and Computer Graphics, and the Head of the Institute of Informatics at the University of Szeged, Hungary. His research interests include image processing, medical imaging, and pattern recognition.

*Received: August 29, 2018; Accepted: September 10, 2019.*

Computer Science and Information Systems 17(1):181–203 https://doi.org/10.2298/CSIS181012035B

# Comparison of systematically derived software metrics thresholds for object-oriented programming languages

Tina Beranič[1] and Marjan Heričko[1]

University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
{tina.beranic, marjan.hericko}@um.si

**Abstract.** Without reliable software metrics threshold values, the efficient quality evaluation of software could not be done. In order to derive reliable thresholds, we have to address several challenges, which impact the final result. For instance, software metrics implementations vary in various software metrics tools, including varying threshold values that result from different threshold derivation approaches. In addition, the programming language is also another important aspect. In this paper, we present the results of an empirical study aimed at comparing systematically obtained threshold values for nine software metrics in four object-oriented programming languages (i.e., Java, C++, C#, and Python). We addressed challenges in the threshold derivation domain within introduced adjustments of the benchmark-based threshold derivation approach. The data set was selected in a uniform way, allowing derivation repeatability, while input values were collected using a single software metric tool, enabling the comparison of derived thresholds among the chosen object-oriented programming languages. Within the performed empirical study, the comparison reveals that threshold values differ between different programming languages.

**Keywords:** software metrics, threshold values, reference values, object-oriented, benchmark data, programming language interdependence, reliable derivation, repeatability, replication

## 1. Introduction

Measurement is a key component for good software engineering, important for understanding, control and improvement [9]. It is performed with software metrics that facilitate the monitoring of the achieved quality level [19]. As defined, the software metric is a quantitative measurement of the degree to which an evaluated entity possesses a specific attribute [17]. Many different object-oriented software metrics have been introduced [18,29,6,3,24]. However, their use in practice, especially within software quality evaluation, is limited, since reliable threshold values have not been proposed, [19,10,1].

Evaluating software quality with software metrics thresholds is a known approach [29]. When software metric values of the assessed software entity exceed the threshold values of the evaluated software metrics, this indicates potential problems in the form of code deficiencies or smells, non-optimal source code, or different structural problems. In the study by Beranič et al. [5], identification of deficient code was done using the combination of software metrics and corresponding threshold values. The performed expert judgment confirmed the efficiency of identification based on derived thresholds, since, by using the highest threshold, the proposed evaluation resulted in the detection of truly deficient software classes.

Since efficient software quality evaluation can be done only with reliable threshold values, the process of threshold derivation is very important. Different approaches for deriving threshold values are proposed in the literature [4], including approaches based on benchmark data, like [19,10,1,30,13,21]. They use software metric values as an input, and provide concrete threshold values for selected software metrics. Resulting thresholds vary between studies, and besides, generally applicable and accepted threshold values cannot be found within related work. This may be due to the different challenges which exist in the domain of software metrics threshold derivation.

When deriving threshold values, it is crucial that the input data sets are transparent, and that they are gathered in a systematic and uniform way. With this, the reliability of the results is increased and repeatability of calculations is achieved. Different software metric tools are available that enable the collection of software metric values. However, the implementation of the same software metric often varies within different tools [31,10,13,44,22,14,33], resulting in different values for the same software metric using the same input data. In various software metric tools, a set of supported metrics differs and, additionally, new tool-specific software metrics can be detected. Available software metric tools rarely enable the collection of software metric values for more than one programming language, wherein the broadest support is available in the Java programming language [4]. The above-mentioned challenges affect the derivation of software metric threshold values directly, especially when using benchmark-based approaches, where the gathered metric values present an input data set into the threshold calculation step.

Also, different approaches for software metric threshold derivation based on benchmark data are available in existing literature. Each has its own characteristics, that are reflected in differing threshold values, though the input data set is the same. In our preliminary research, we compared threshold values derived by using approaches by Ferreira et al. [10], Oliveira et al. [30] and Alves et al. [1], and confirmed that the derived thresholds vary. This was also confirmed by Yamashita et al. [44], where they observed that the derived thresholds would differ if a different derivation approach were to be used. Hence, to provide comparable results, it is important that only a single derivation approach is used.

The prevalence of the Java programming language within software metric tools is also detectable between existing threshold derivation approaches, where most of the threshold values are derived for Java, while other object-oriented programming languages are not covered. The only exceptions, using the benchmark based threshold derivation approach, are the studies by Alves et al. [1], which derived threshold values for C#, and by Lanza and Marinescu [19], who derived threshold values for the C++ programming language. Though some individual examples of derived thresholds exist, the systematic analysis of threshold values between programming languages was not detected.

Based on the presented background, our research pursued the following research question: *Do software metric threshold values differ between different object-oriented programming languages?* We analyzed if the programming language has an influence on the derived threshold values. The presented research work describes a systematic threshold derivation, which enables a reliable analysis and comparison of software metric threshold values. In the paper, software metric threshold values are derived for four object-oriented programming languages, namely Java, C++, C# and Python. Considering the above-mentioned challenges, the derivation was conducted using a single threshold derivation approach within all of the selected programming languages. Thresholds were derived for nine class level software metrics. Metric *CountLineCode*, that counts the number of lines of code in a class, *AvgLineCode*, expressing the average size of methods in a class, metric *SumCyclomatic*, presenting the sum of cyclomatic complexities of all the methods in a class, *AvgCyclomatic*, expressing the average value of cyclomatic complexity in the methods of a class, metric *MaxNesting*, measuring the maximal nesting level in a class, *CountClassCoupled*, counting the classes to which a class is coupled, software metric *PercentLackOfCohesion*, expressing the lack of cohesion in a class, metric *CountDeclMethodAll*, counting the number of methods in a class, and metric *MaxInheritanceTree*, expressing the maximum depth of a class in the inheritance tree. Benchmark data was collected systematically, and input values were collected using a single software metric tool. With this, the comparison of derived software metric threshold values between programming languages was enabled, and a replication of the performed derivation approach was provided.

The structure of the paper consists of the following parts. Chapter 2 presents related work, followed by Chapter 3, presenting a threshold derivation approach that is based on a statistical distribution of benchmark data. In Chapter 3, adjustments to the approach are proposed, and the tools and data sets used within the empirical study are presented. Chapter 3.3 describes the calculation of thresholds, covering the distribution analysis of input values, resulting in concrete threshold values. Later, an analysis and comparison of derived threshold values are presented in Chapter 4. Limitations and threats to validity are presented at the end.

## 2. Related work

Software quality evaluation with software metrics can be done only when reliable threshold values are defined. Different approaches for deriving threshold values are available in the literature [4]. Fontana et al. [13] categorizes derivation approaches into (1) approaches based on observations, (2) error-based approaches, (3) approaches using machine learning and, (4) approaches that derive thresholds based on a statistical analysis of benchmark data. In the presented research, we focus on the latter.

Table 1 lists derivation approaches based on benchmark data. The approach proposed by Lanza and Marinescu [19] derives threshold values using the mean and standard deviation, but the distribution of input data sets is not considered. On the other hand, the majority of studies consider the assumption that software metrics values usually follow a power law distribution [10,30,38,1,7], since distribution has a significant impact on software metric interpretation [38]. The distribution is also considered by Lavazza and Morasca [20]. Although they use the mean and the standard deviation, an improvement is proposed that enables the use of data that does not follow a normal distribution. Approaches by Ferreira et al. [10], Oliveira et al. [30], Alves et al. [1], Lima et al. [21], Vale and Figueiredo [41] and Vale et al. [42] consider the fact that software metric values usually do not follow a normal distribution, leading to the inapplicability of methods connected to normal distribution [1,38]. Ferreira et al. [10] identify threshold values for six

| Study | Programming language(s) |
|---|---|
| Alves et al. [1] | Java, C# |
| Ferreira et al. [10] | Java |
| Filo et al. [12] | Java |
| Fontana et al. [13] | Java |
| Lanza and Marinescu [19] | Java, C++ |
| Lavazza and Morasca [20] | Java |
| Lima et al. [21] | Java |
| Oliveira et al. [30] | Java |
| Vale and Figueiredo [41] | SPL benchmark (FH-Java, AHEAD, FH-JML) |
| Vale et al. [42] | SPL benchmark (FH-Java, AHEAD, FH-JML), Java |

**Table 1.** Literature proposing software metric threshold derivation approaches

object-oriented software metrics reflecting a common practice. Filo et al. [12] introduced two improvements to the approach presented by Ferreira et al. [10], the modification of the ranges names, and the use of two percentiles, dividing the values into three areas. Alves et al. [1] propose an approach using weighting according to the size of the entities, wherein the approach was an inspiration for the work by Fontana et al. [13]. Fontana et al. [13] define thresholds for the metrics used in the code smell detection rules. Lima et al. [21] addressed the area of threshold derivation for annotations in the Java programming language, and Vale and Figueiredo [41] present a threshold derivation approach in the software product lines context. In the study [42], Vale et al. generalize the approach by using a benchmark composed of 103 Java open source projects. The major difference according to other available studies is that Vale et al. [42] also extract the lower bond thresholds, namely the 3$^{rd}$ and 15$^{th}$ percentiles. Oliveira et al. [30] introduce the concept of the relative threshold, implemented in an RTTool [31]. Another threshold derivation tool, a TDTool, is presented by Veado et al. [43].

As seen in Table 1, the threshold values are mainly derived for the Java programming language. In rare cases, thresholds are derived for two or more languages, as in [1,19,42]. However, the empirical comparison of threshold values between programming languages was not found within the related work.

The studies were done in order to define the impact of different contextual factors on derived threshold values, or on the distribution of software metric values. Ferreira et al. [10], in an experiment, analyzed the impact of thematic domains in gathered benchmark data. As they observed, there is only a slight difference between the thresholds derived using a full data set and thresholds derived within each of the thematic domains [10]. Even more, the results show that software metric values follow the same probability distribution regardless of the application domain. Mori et al. [28] also analyzed the impact of domains on derived thresholds. In contrast to the study described above, they found evidence that software metrics thresholds are sensitive to the software domain, but we can still find domains that have similar thresholds for some of the analyzed software metrics [28].

On the other hand, Dósea et al. [8] conducted an empirical study regarding design decisions influencing the distribution of software metrics. As they conclude, the design roles affect the distribution of metric values, wherein design roles include architectural roles and classes with application-specific responsibility that are not connected to any specific reference architecture [8]. The impact of programming language on the distribution of software metrics values was included in the study by Zhang et al. [45] and a study by Gil and Lalouche [15]. Zhang et al. [45] present a study about the impact of different contextual factors on maintainability metrics. They found out that application domain, programming language, and the number of changes, are the most influential factors regarding the distribution of software metric values [45]. They use the data from 320 software projects, selected randomly from SourceForge, but they were not represented equally for each of the used programming languages, which can have a great impact on the validity of results. A similar study was done by Gil and Lalouche [15], where they found out that every project is different, therefore, measurement in one project could not be used for making predictions in another software project.

Namely, the challenges connected to benchmark data present one of the biggest challenges when using benchmark based threshold derivation approaches. Considering the above-mentioned challenges, we collected the data in a systematic way, where each programming language is represented equally and selected software projects correspond to the whole population of open-source software available on the chosen online repository. Moreover, to avoid the impact of a specific software project on a final result, we gathered a large set of benchmark projects, since Lochmann [23] found out that, with large numbers of data, the diversity of areas and the variance of results decreases correspondingly.

Our research work aims at providing a comparison of the software metrics threshold values for the object-oriented programming languages, namely Java, C++, C# and Python, considering known challenges. The comparison is based on the performed analysis of systematically derived threshold values for nine software metrics. By adopting and adjusting the existing derivation approach based on benchmark data and considering the distribution of software metric values, thresholds are calculated based on systematically collected benchmark data and uniformly collected input values.

## 3. Threshold derivation approach

As presented in the related work, software metric values are used as an input in different threshold derivation approaches. Lanza and Marinescu [19] do not consider data distribution, and Lavazza and Morasca [20], despite the presented changes, do not consider fully the distribution of software metric values. Vale and Figueiredo [41] and Vale et al. [42] present a method that also derives the lower thresholds that are not a priority when identifying deficient source code, and Lima et al. [21] present an approach targeting annotation for Java programming languages. Therefore, only papers by Alves et al. [1], Oliveira et al. [30] and Ferreira et al. [10], with improvements presented by Filo et al. [12], that present derivation approaches and resulting in concrete threshold values used for the evaluation of software projects, were selected for a detailed analysis. Approaches are similar, wherein Alves et al. [1] weigh the program entities based on their size expressed with lines of code software metric, Ferreira et al. [10] consider the frequency of a specific software metric value, and Oliveira et al. [30] introduce the concept of a relative threshold. As shown by the performed comparison, weighing by size results in very high threshold values, and the approach presented by Oliveira et al. [30], with the exception of the newly presented concept, resulted in threshold values consistent with thresholds provided by the approach proposed by Ferreira et al. [10].

We decided to adopt the approach presented by Ferreira et al. [10]. The approach focuses on the statistical properties of analyzed data and object-oriented programming languages. Thresholds are defined according to the frequency concept within the benchmark data. The proposed approach can be summarized with the following steps [10]:

1. The software metrics values of selected benchmark projects are gathered, forming an input data set.
2. The distribution of software metrics values is determined for each metric, using a visual analysis, and by using a distribution fitting tool.
3. The thresholds are derived based on the best-fitted distribution. If representative values exist for a best-fitted distribution, it is defined as the threshold. Otherwise, three areas are determined using a visual examination:
   – The *Good* area joins the values of software metrics with a high frequency of occurrence. Those values are used most commonly in practice.
   – The *Regular* area represents an intermediate zone, joining the values that are not commonly used, and, on the other hand, are also not very rare.
   – The *Bad* area joins values with a very low frequency of occurrence.

In the performed experiment, Ferreira et al. [10], with the use of derived thresholds, identify software classes with structural problems, wherein a bad value indicates the existence of design problems, and a good value indicates the absence of structural problems in a class.

The approach was repeated and upgraded by Filo et al.[12]. They introduced two main improvements. The first is connected to the identification of thresholds. Instead of using the visual identification, Filo et al. [12] introduced the use of two percentiles that divide the values into three areas. The percentiles are points dividing values into 100 equal parts [11]. The use of percentiles was adopted from Alves et al. [1]. The second improvement is connected to threshold naming. They complement existing names to achieve a better understanding of each defined threshold. The names of the ranges are as follows: *good/common*, *regular/casual* and *bad/uncommon*, but the use of derived software metrics' thresholds for identification of the anomalous values indicating a potential problem in source code, remains unchanged [12].

## 3.1. Adjustments of the adopted approach

Based on the analysis of the replicated approach presented by Ferreira et al. [10] and improvements introduced by Filo et al. [12], we propose some additional adjustments for the adopted threshold derivation approach. When software metric thresholds are fied using the presented steps, a few challenges arise. The first one is related to the fication of threshold values that limit the mentioned areas. A visual examination was already replaced by the use of two percentiles in the study presented by Filo et al. [12]. Adapted from Alves et al. [1], they use the 70th and 90th percentiles to form three risk areas, although the primary study by Alves et al. [1] proposed the use of three percentiles, i.e. 70th, 80th and 90th to form four risk areas.

We propose an adjustment of the replicated approach by using two or three percentiles, depending on the input data range and suitability of values. If the software metric values occupy a wide range of data, thresholds can be determined with three percentiles: 70th, 80th and 90th. If the values occupy a limited range of data, thresholds should be fied with two percentiles: 70th and 90th, and the very high risk area should not be included. Also, to exclude any subjectivity and to gain accuracy, we propose that the fication of percentiles is done based on raw data sets instead of the visual examinations used by Ferreira et al. [10] and Filo et al. [12].The detailed steps of the used threshold derivation approach are presented within algorithm 1.

**Algorithm 1** Threshold derivation process

```
1: collect and prepare input data set for each software metrics $SM_{1...i}$
2: for each software metric $SM_{1...i}$ do
3:     collect descriptive statistics
4:     obtain kurtosis and skewness
5:     analyze collected data set
6:     verify normal distribution
7: end for
8: for each software metric $SM_{1...i}$ do
9:     find best fitted distribution
10:    verify heavy tail distribution
11: end for
12: for each software metric $SM_{1...i}$ do
13:    derive threshold value $T_{1...i}$
14:    if distribution equals power law then
15:        determine thresholds value $T_{1...i}$ regarding the distribution
16:        using 70th percentile determine low risk area
17:        using 70th and 80th percentile determine moderate risk area
18:        using 80th and 90th percentile determine high risk area
19:        using 90th percentile determine very high risk area
20:    else
21:        determine thresholds value $T_{1...i}$ regarding the distribution
22:        using threshold values determine risk area
23:    end if
24: end for
```

Filo et al. [12] complemented the naming of areas proposed by Ferreira et al. [10] to increase the understanding of the derived thresholds. The proposed names reveal the frequency of use of values within each area, whereas we propose renaming the areas to express the risk each area represents within the context of quality evaluation. The naming was suggested by Alves et al. [1] and is based on the risk perspective. We propose the following naming of the fied areas:

- *low* risk, $\leq$70th,
- *moderate* risk, $>$70th and $\leq$80th,
- *high* risk, $>$80th and $\leq$90th and
- *very high* risk, $>$90th.

The formed areas express the risk that an evaluated program entity includes irregularities in the context of different smells, specific structural problems, or potential deficiencies. The low risk area is determined with the 70th percentile, therefore, coinciding with the good/common area, as proposed by Filo et al. [12], indicating the absence of structural problems in a class. On the other hand, if a very high risk area coincide with a bad/uncommon area, this indicates the existence of design problems within the chosen software entity. The described risk areas were used in the study by Beranič et al. [5] for the detection of deficient source code. The study detects deficient software entities based on the combination of quality aspects, increasing the reliability of the identification. Since the use of only one software metric covers a single quality dimension, the use of a combination is crucial for reliable results. The expert judgment performed within the evaluation of the proposed approach confirmed that software entities that have the majority of metric values in the area of very high risk, are evaluated accurately as deficient.

## 3.2. Selection of tools and data set for the empirical study

Within threshold derivation, one of the well-known challenges is how to provide comparable results between selected programming languages, and this constitutes precisely the key driver of our research. We aimed at providing threshold values for the same software metrics in four different programming languages. To overcome the distinctive definitions of software metrics within various software metric tools, the input data set into the derivation process should be gathered using a single software metric tool. Based on the performed analysis, we chose an Understand tool [35], that supports the collection of software metric values for multiple programming languages [36]. With this, the risk was addressed and eliminated of providing varying, tool dependent values for the same metric. Besides, collecting values with a single software metric tool enables a more objective comparison among different programming languages.

In the threshold derivation approach, the statistical properties of the input data set were assessed with an SPSS tool [16], and by using R [32]. The best fitted distribution according to the input data set was found with the EasyFit tool [26], and the fication of thresholds was obtained from software metric values using the SPSS tool [16].

The second major challenge that has to be addressed when calculating thresholds using a benchmark data approach is the input data set. To provide reliable thresholds, the input data has to be diverse, extensive and transparent. As Lochmann [23] found out, when the input data set is larger, the diversity of areas and the variance of results decreases correspondingly. Therefore, a large benchmark base reduces the impact of randomly selected software products [34,23]. To determine the optimal size of input data set, we reviewed studies deriving threshold values from benchmark data. The number of software projects used by each study is presented in Table 2.

| Study | Number of software projects |
|---|---|
| Alves et al. [1] | 100 |
| Arar and Ayan [2] | 10 |
| Ferreira et al. [10] | 40 |
| Filo et al. [12] | 111 (from Qualitas Corpus [40]) |
| Fontana et al. [13] | 74 (from Qualitas Corpus [40]) |
| Mori et al. [28] | 3,107 |
| Oliveira et al. [30] | 106 (from Qualitas Corpus [40]) |
| Yamashita et al. [44] | 4,780 |
| Vale et al. [42] | 103 (from Qualitas Corpus [40]) |

**Table 2.** Number of projects used as benchmark data within the threshold derivation process

Oliveira et al. [30], Fontana et al. [13], Filo et al. [12] and Vale et al. [42] used projects from Qualitas Corpus [40], whereas Alves et al. [1] used 100 software products, including open source and proprietary solutions. Yamashita et al. [44] also used the combination of open source and industrial software solutions, wherein 205 projects were proprietary and 4,757 projects were open source projects. Mori et al. [28] included 3,107 software systems divided into 15 domains, since their focus was to analyze the impact of domains on derived threshold values. On the other hand, Ferreira et al. [10] and Arar and Ayan [2] used a smaller benchmark, including 40 and 10 projects, respectively.

Summarizing the collected numbers, we decided to use 100 software projects and use the obtained values of software metrics as an input data set for threshold derivation. Since we derived thresholds for the programming languages Java, C++, C# and Python, the use of Qualitas Corpus collection [40] was not possible. The collection combines software developed in the Java programming language, and, not knowing the conditions by which the software projects were chosen, a comparable suite for the other three programming languages could not be gathered.

We formed a reusable suite of software products that enables repeatability, and contributes to the objectivity of the presented empirical research. The suite includes 400 software projects, 100 in each of the selected programming languages. The list of used software products is available at: https://bit.ly/2RIQhle. Since software projects were chosen and gathered systematically, it allows a reliable comparison of derived thresholds.

In the implemented study, only open source software was used. We collected the input software solution from SourceForge [39], that allows categorization of software projects using different criteria, e.g., programming language, operating system, license, user interface and others. Also, software projects are categorized into different thematic domains, where each domain includes a different number of projects. Therefore, the ratio within every programming language was transferred to the selected sample of 100 projects to keep a ratio of the population. The impact of application domains on benchmark data set values was studied by Ferreira et al. [10]. Thresholds were derived using the benchmark data from 11 application domains. The results show that software metric values follow the same probability distribution, regardless of the application domain, and that there is only a slight difference between the thresholds derived for the used domains and the thresholds derived using a full data set [10]. As they conclude, regardless of the observed minor differences, the general results can be used for all application domains [10].

Within the scope of our experimental study, the selection of software projects was performed in several steps. In the first step, we applied the programming language filter, therefore, four lists were formed, a list of Java, C++, C# and Python projects. In the second step, we applied different filters to the project lists to fy only those projects that are regularly maintained and stable, which suggests that they follow best software development and maintenance practices. We considered criteria related to status and freshness, which were chosen by using the following filters: (1) status - production/stable, (2) freshness - recently updated. The third step sorted the filtered software projects by their popularity in descending order, forming another filter for fication of stable projects. In the fourth step, ordered and filtered lists were divided into different thematic domains using a category filter, and producing a final input list into the software selection step. In the fifth step, the corresponding number of software projects were chosen from each category, considering the ratio accessible in the population. The data about each project were documented, and the actual version of source code was downloaded. Each project got a unique fication key that allows for traceability across a derivation approach.

The descriptive statistic of the established reusable suite is presented in Table 3. It presents the statistics for selected software projects for each of four programming lan-

| | # files | # classes | # lines | # code lines |
|---|---|---|---|---|
| | Java | | | |
| Min | 12 | 12 | 1,511 | 957 |
| Max | 18,469 | 22,837 | 4,897,144 | 2,823,916 |
| Avg | 1,436.1 | 2,266.3 | 314,915.9 | 184,994.5 |
| | C++ | | | |
| Min | 6 | 1 | 1,236 | 687 |
| Max | 26,517 | 34,718 | 10,032,886 | 5,996,402 |
| Avg | 1,644.8 | 987.4 | 542,526.0 | 312,326.8 |
| | C# | | | |
| Min | 5 | 4 | 570 | 378 |
| Max | 7,656 | 10,422 | 1,311,745 | 857,729 |
| Avg | 529.3 | 686.4 | 124,241.2 | 80,875.4 |
| | Python | | | |
| Min | 6 | 1 | 2,139 | 1,114 |
| Max | 4,167 | 10,085 | 915,078 | 583,278 |
| Avg | 287.8 | 587.9 | 82,126.5 | 54,830.1 |

**Table 3.** Descriptive statistics of selected software projects in a reusable suite

guages, presenting the minimum, maximum and average values of the number of files, classes, the total number of lines, and number of lines of code.

Since the methodology of the selection of software products is presented and documented systematically, it can be repeated, and, with this, the formed suite of software projects can be extended to other programming languages.

## 3.3. Calculation of threshold values

The focus of our research was on deriving threshold values for class level software metrics. As presented in Chapter 3.2, a reusable suite of software products was established, including 400 software products, 100 for each of the selected programming languages. For every software project in the suite, software metrics were collected with the Understand tool [35]. Input files were prepared according to guidelines presented by the replicated approach [10], wherein the input file for the Java programming language included 206,730 records, the file for C++ 98,762 records, the file for C# had 81,293 records and the input file for Python included 60,462 records.

The Understand tool [35] allows for the collection of 102 software metrics, evaluating different levels. Our study is limited to Java, C++, C# and Python. Since all software metrics are not supported in all programming languages, meaning that the support for Python is limited, we decided to calculate thresholds for nine software metrics:

- *CountLineCode* - number of lines of code in a class,
- *AvgLineCode* - average size of methods in a class in lines of code,
- *SumCyclomatic* - the sum of cyclomatic complexities for all the methods in a class,
- *AvgCyclomatic* - the average value of cyclomatic complexity in the methods in a class,
- *MaxNesting* - the maximal nesting level in a class,
- *CountClassCoupled* - number of classes to which a class is coupled,
- *PercentLackOfCohesion* - the lack of cohesion in a class,
- *CountDeclMethodAll* - number of methods in a class, including inherited ones, and
- *MaxInheritanceTree* - the maximum depth of a class in the inheritance hierarchy.

Metrics evaluating size and complexity are probably the most widely used software measurements [19]. Size-related software metrics are aimed to quantify the size of a software [37], for example, the metric *CountLineCode* expresses the number of lines of source code in a chosen software class, excluding blank lines and comment lines, and *AvgLineCode* expresses the average method size in a class, expressed with the number of lines of code. Related to the latter, is also the metric *CountDeclMethodAll*, counting all the methods within a class, taking into account inherited methods [36]. McCabe [27], in 1976, introduced a complexity measure known as Cyclomatic Complexity. *SumCyclomatic* and *AvgCyclomatic* are measuring complexity, wherein *SumCyclomatic* expresses the sum of cyclomatic complexities of all the methods in a class, and metric *AvgCyclomatic* gives an overview, expressing the average value of cyclomatic complexity of all the methods in a class. Another aspect affecting the complexity of a program entity is covered by metric *MaxNesting* [46], expressing the maximal nesting level in a class.

In addition to the above-mentioned software metrics, thresholds were also derived for different object-oriented software metrics. Chidamber and Kemerer [6] proposed a metrics suite aimed at measuring specific object-oriented properties. Among others, they define a metric measuring coupling between object classes, a metric expressing lack of cohesion in methods, and a metric expressing depth of the inheritance tree. The latter are implemented in Understand tool [35] as *CountClassCoupled*, *PercentLackOfCohesion* and *MaxInheritanceTree*, respectively. The software metric *CountClassCoupled* measures the coupling of a class to other classes. Two classes are coupled if one class uses the methods and variables defined in another class [6]. High coupling is not desirable, since the reuse is difficult because of decreased modularity [37]. On the other hand, the software metric *PercentLackOfCohesion* expresses the lack of cohesion in a class. High cohesion means that methods and attributes cooperate with each other and form a logical whole [25]. The lack of cohesion may suggest that a class should be divided [37]. One of the advantages of object-oriented design is the reuse of program entities. We can form classes that inherit functionalities from their parent class [19]. Software metric *MaxInheritanceTree* expresses the maximum depth of a class in an inheritance hierarchy. The *MaxInheritanceTree* of the root node is 0 [36]. In a case of multiple inheritances, the metric expresses only the maximum length from the class node to the root of the inheritance tree [6]. The deeper the class is in a hierarchy, the more methods could be inherited, which, consequently, increases the complexity in the design [37].

For Java, C++ and C#, thresholds were derived for nine different software metrics, and for Python, thresholds were derived for seven software metrics, since the metrics *CountClassCoupled* and *PercentLackOfCohesion* are not supported by the used software metric collection tool. Following the approach presented by Ferreira et al. [10], the threshold derivation approach starts by checking the distribution of the input data set, which is to say, by finding the best-fitted distribution. First, it was checked to see if data are distributed normally with the use of descriptive statistics. The latter was used to confirm the power law distribution in the data set by Shatnawi and Althebyan [38]. Within normal distributions, the values are centralized strongly around the arithmetic mean, meaning that the latter presents a representative value that a random variable can occupy [38]. For each software metric in each of the selected programming languages, we gathered values for the arithmetic mean, median, standard deviation and maximal value. Also, the values of kurtosis and skewness were obtained, that enable an insight into data distribution and in-

| | *CountLineCode* | | | | *AvgLineCode* | | | |
|---|---|---|---|---|---|---|---|---|
| | Java | C++ | C# | Python | Java | C++ | C# | Python |
| kurtosis | 2,439.1 | 5,911.9 | 15,921.9 | 834.9 | 17,284.6 | 310.7 | 6,056.9 | 5,966.7 |
| skewness | 32.1 | 60.2 | 113.7 | 19.6 | 91.2 | 10.7 | 64.7 | 59.3 |
| arithmetic mean | 88.3 | 114.9 | 136.9 | 58.6 | 9.4 | 7.8 | 9.5 | 7.6 |
| median | 27 | 29 | 41 | 17 | 6 | 4 | 5 | 4 |
| standard deviation | 283.4 | 568.8 | 995.8 | 166.6 | 18.9 | 12.3 | 28.9 | 19.0 |
| maximum | 36,273 | 74,278 | 156,163 | 11,798 | 4,312 | 666 | 3,414 | 2,159 |
| kurtosis | leptokurtic | | | | leptokurtic | | | |
| skewness | positive | | | | positive | | | |

| | *SumCyclomatic* | | | | *AvgCyclomatic* | | | |
|---|---|---|---|---|---|---|---|---|
| | Java | C++ | C# | Python | Java | C++ | C# | Python |
| kurtosis | 4,182.6 | 11,670.1 | 24,038.2 | 677.3 | 1,297.9 | 783.2 | 7,018.4 | 348.8 |
| skewness | 45.7 | 84.3 | 148.8 | 17.8 | 21.8 | 18.5 | 68.3 | 12.9 |
| arithmetic mean | 14.8 | 22.5 | 18.2 | 13.7 | 1.7 | 1.9 | 1.6 | 1.7 |
| median | 4 | 6 | 5 | 4 | 1 | 1 | 1 | 1 |
| standard deviation | 52.1 | 125.9 | 208.5 | 40.1 | 2.2 | 3.5 | 4.3 | 2.7 |
| maximum | 7,026 | 21,581 | 34,702 | 2,430 | 206 | 228 | 524 | 150 |
| kurtosis | leptokurtic | | | | leptokurtic | | | |
| skewness | positive | | | | positive | | | |

| | *MaxNesting* | | | | *CountClassCoupled* | | | |
|---|---|---|---|---|---|---|---|---|
| | Java | C++ | C# | Python | Java | C++ | C# | Python |
| kurtosis | 5.3 | 3.2 | 3.5 | 3.3 | 522.4 | 140.1 | 58.3 | n/a |
| skewness | 1.9 | 1.6 | 1.7 | 1.6 | 12.0 | 10.1 | 4.9 | n/a |
| arithmetic mean | 1.1 | 1.2 | 1.3 | 1.2 | 4.8 | 6.6 | 9.9 | n/a |
| median | 1 | 1 | 1 | 1 | 2 | 3 | 6 | n/a |
| standard deviation | 1.4 | 1.6 | 1.6 | 1.5 | 8.4 | 14.2 | 12.7 | n/a |
| maximum | 21 | 16 | 18 | 16 | 704 | 328 | 403 | n/a |
| kurtosis | leptokurtic | | | | leptokurtic | | | |
| skewness | positive | | | | positive | | | |

| | *PercentLackOfCohesion* | | | | *CountDeclMethodAll* | | | |
|---|---|---|---|---|---|---|---|---|
| | Java | C++ | C# | Python | Java | C++ | C# | Python |
| kurtosis | -1.4 | -1.6 | -1.6 | n/a | 2,699.3 | 248.7 | 11,728.9 | 7.8 |
| skewness | 0.5 | -0.1 | 0.2 | n/a | 22.8 | 13.3 | 82.7 | 2.6 |
| arithmetic mean | 32.8 | 48.9 | 37.2 | n/a | 20.21 | 43.9 | 30.8 | 27.24 |
| median | 0 | 52 | 33 | n/a | 5 | 16 | 16 | 11 |
| standard deviation | 38.1 | 40.8 | 37.2 | n/a | 46.8 | 128.1 | 90.5 | 41.3 |
| maximum | 100 | 100 | 100 | n/a | 7,113 | 3,776 | 14,246 | 464 |
| kurtosis | mesokurtic | | | | leptokurtic | | | |
| skewness | symmetric | | | | positive | | | |

| | MaxInheritanceTree | | | |
|---|---|---|---|---|
| | Java | C++ | C# | Python |
| kurtosis | 655.5 | 2.7 | 422.1 | 1.1 |
| skewness | 8.4 | 1.4 | 14.5 | 1.0 |
| arithmetic mean | 1.7 | 1.2 | 0.9 | 1.8 |
| median | 1 | 1 | 1 | 1 |
| standard deviation | 1.1 | 1.2 | 1.5 | 1.8 |
| maximum | 118 | 11 | 58 | 10 |
| | leptokurtic | | | |
| | positive | | | |

**Table 4.** Descriptive statistics of analyzed software metrics values

dicate a deviation from the normal distribution. Kurtosis expresses the size of peaks and skewness measures the symmetry of the used data set. Data that are normally distributed have a value of kurtosis and skewness of approximately zero [11]. When the values move away from zero it proves that they are not following a normal distribution and values are gathered on one end of the scale, and values are distributed in a peak or are flattened. Data that follow heavy tailed distributions have a positive skew. For the positive skew, also apply [38]:

$$\begin{gathered} \textit{standard deviation} \gg \textit{arithmetic mean} \\ \textit{standard deviation} \gg \textit{median} \end{gathered} \tag{1}$$

$$\textit{maximum} \gg \textit{arithmetic mean} \tag{2}$$

Besides the mentioned, the positive skew is indicated by:

$$\textit{arithmetic mean} \geq \textit{median} \geq \textit{mode} \tag{3}$$

Descriptive statistics for the evaluated metrics are presented in Table 4. Software metric values are limited to the left, with a value 0, and unlimited to the right, since the maximum value is usually not defined [13]. Among the gathered values, data describing the metric *PercentLackOfCohesion* that expresses a lack of cohesion in a class, stand out. The metric can occupy a value from 0 to 100, since the result is expressed in percentages. Based on the numbers, it is the only metric for which descriptive statistics do not discard normal distribution. Other software metrics, without a doubt, do not follow a normal distribution, as reflected by their positive skew and leptokurtic distribution. Namely, when the values of skewness are more than 0, a positive skew is present, which is reflected with values gathered on the left, and individual values on the right that form a tail [11,38]. On the other hand, the positive value of kurtosis is shown in a bigger peak of distribution, and indicates that the values are forming a heavy tail [11].

After the descriptive statistics were analyzed, the best fitted distribution for data was determined using an EasyFit tool [26]. More than 55 distributions are available, and the tool checks how well a chosen distribution fits an input data set, and arranges them according to performance. Table 5 presents the best-fitted distributions for selected software metrics values in four programming languages.

The threshold values were determined after the data distribution was determined for each software metric in all of the four programming languages. All software metrics, except *PercentLackOfCohesion*, correspond to a heavy tail distribution. Because of this, the derivation could be done as proposed by Ferreira et al. [10], by using percentiles, and considering the proposed adjustments related to risk areas. As suggested in 3.1, thresholds were determined using two or three percentiles using an SPSS tool [16].

The values of software metrics *AvgCyclomatic*, *MaxNesting* and *MaxInheritanceTree* are presented within a small range of data. For example, the metric *MaxNesting* has the same value for the 70th and 80th percentile for C++ and C#. Because forming the area with such small differences between the borders is not feasible, the 80th percentiles was excluded and only the 70th and 90th percentiles were used for forming the threshold risk areas. The metric *PercentLackOfCohesion* follows a Uniform distribution, and thresholds cannot be determined using percentiles. For this purpose, the threshold value was determined using the arithmetic mean and standard deviation.

| Software metric | Programming language | Distribution |
|---|---|---|
| *CountLineCode* | Java | Inverse Gaussian |
| | C++ | Dagum |
| | C# | Pareto 2 |
| | Python | Wakeby |
| *AvgLineCode* | Java | Generalized Pareto |
| | C++ | Generalized Pareto |
| | C# | Generalized Logistic |
| | Python | Generalized Logistic |
| *SumCyclomatic* | Java | Phased Bi-Weibull |
| | C++ | Generalized Pareto |
| | C# | Generalized Pareto |
| | Python | Generalized Pareto |
| *AvgCyclomatic* | Java | Generalized Logistic |
| | C++ | Wakeby |
| | C# | Phased Bi-Exponential |
| | Python | Phased Bi-Exponential |
| *MaxNesting* | Java | Gumber Max |
| | C++ | Gumber Max |
| | C# | Gumber Max |
| | Python | Gumber Max |
| *CountClassCoupled* | Java | Phased Bi-Weibull |
| | C++ | Generalized Logistic |
| | C# | Wakeby |
| | Python | n/a |
| *PercentLackOfCohesion* | Java | Uniform |
| | C++ | Uniform |
| | C# | Uniform |
| | Python | n/a |
| *CountDeclMethodAll* | Java | Wakeby |
| | C++ | Wakeby |
| | C# | Generealized Pareto |
| | Python | Johnson SB |
| *MaxInheritanceTree* | Java | Gamma |
| | C++ | Gumber Max |
| | C# | Logistic |
| | Python | Johnson SB |

**Table 5.** Best fitted distributions

## 4. Empirical analysis of derived threshold values

Based on derived threshold values, calculated points were used to set three or four risk areas. Thresholds are presented in the form of areas.

Areas, as determined in Chapter 3.1, are formed according to the risk that an evaluated program entity includes irregularities. For example, if a class has 300 lines of code, a very high risk (VHR) exists that something within the entity is not optimal. This does not mean that defects are present, but that there may be some irregularities in the context of different smells or specific technical debts. However, we have to be aware, that combining different software metrics when evaluating software quality could improve the reliability of provided results significantly. Values lower than the 70th percentile belong to a low risk (LR) area, values between the 70th and 80th percentiles form a moderate risk (MR) area, and values bigger than the 80th percentile and smaller, or equal to the 90th percentile

| | | Java | C++ | C# | Python |
|---|---|---|---|---|---|
| *CountLineCode* | LR | x⩽61 | x⩽66 | x⩽90 | x⩽43 |
| | MR | 61<x⩽100 | 66<x⩽112 | 90<x⩽144 | 43<x⩽71 |
| | HR | 100<x⩽197 | 112<x⩽235 | 144<x⩽278 | 71<x⩽135 |
| | VHR | x>197 | x>235 | x>278 | x>135 |
| *AvgLineCode* | LR | x⩽9 | x⩽8 | x⩽10 | x⩽8 |
| | MR | 9<x⩽13 | 8<x⩽11 | 10<x⩽14 | 8<x⩽11 |
| | HR | 13<x⩽19 | 11<x⩽18 | 14<x⩽20 | 11<x⩽17 |
| | VHR | x>19 | x>18 | x>20 | x>17 |
| *SumCyclomatic* | LR | x⩽10 | x⩽13 | x⩽11 | x⩽9 |
| | MR | 10<x⩽17 | 13<x⩽22 | 11<x⩽18 | 9<x⩽16 |
| | HR | 17<x⩽33 | 22<x⩽45 | 18<x⩽36 | 16<x⩽33 |
| | VHR | x>33 | x>45 | x>36 | x>33 |
| *AvgCyclomatic* | LR | x⩽2 | x⩽2 | x⩽1 | x⩽2 |
| | MR | 2<x⩽3 | 2<x⩽4 | 1<x⩽3 | 2<x⩽4 |
| | HR | x>3 | x>4 | x>3 | x>4 |
| *MaxNesting* | LR | x⩽1 | x⩽2 | x⩽2 | x⩽2 |
| | MR | 1<x⩽3 | 2<x⩽3 | 2<x⩽4 | 2<x⩽3 |
| | HR | x>3 | x>3 | x>4 | x>3 |
| *CountClassCoupled* | LR | x⩽5 | x⩽6 | x⩽11 | n/a |
| | MR | 5<x⩽7 | 6<x⩽9 | 11<x⩽15 | n/a |
| | HR | 7<x⩽11 | 9<x⩽14 | 15<x⩽23 | n/a |
| | VHR | x>11 | x>14 | x>23 | n/a |
| *PercentLackOfCohesion* | LR | x⩽71 | x⩽90 | x⩽74 | n/a |
| | HR | x>71 | x>90 | x>74 | n/a |
| *CountDeclMethodAll* | LR | x⩽14 | x⩽42 | x⩽26 | x⩽24 |
| | MR | 14<x⩽24 | 42<x⩽49 | 26<x⩽34 | 24<x⩽51 |
| | HR | 24<x⩽51 | 49<x⩽90 | 34<x⩽60 | 51<x⩽70 |
| | VHR | x>51 | x>90 | x>60 | x>70 |
| *MaxInheritanceTree* | LR | x⩽2 | x⩽2 | x⩽1 | x⩽2 |
| | MR | 2<x⩽3 | 2<x⩽3 | 1<x⩽2 | 2<x⩽4 |
| | HR | x>3 | x>3 | x>2 | x>4 |

**Table 6.** Risk areas (low risk (LH), moderate risk (MR), high risk (HR) and very high risk (VHR)) based on threshold values

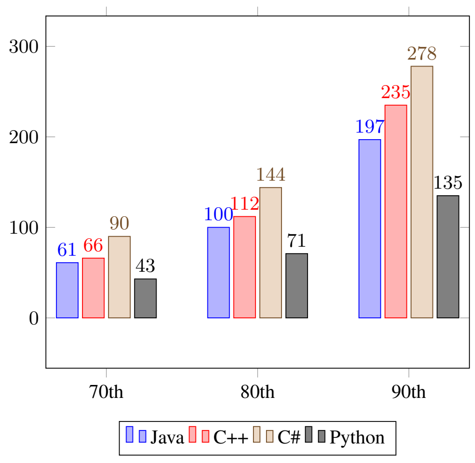
constitute high risk (HR) area, and values that are bigger than determined with the 90th percentile are considered to be in the area of very high risk (VHR). Table 6 presents the defined risk areas and corresponding threshold values. The values are shown for nine software metrics in four programming languages. Where areas are determined with only two percentiles, i.e. in the case of *AvgCyclomatic*, *MaxNesting* and *MaxInheritanceTree*, only three areas are given. Values lower than those determined with the 70th percentile are in the area of low risk (LR), between the 70th and 90th percentile there is a moderate risk (MR) area, and values in a high risk (HR) area are values bigger than determined with the 90th percentile. A special case is the metric *PercentLackOfCohesion*, where only one area is defined, based on the calculated threshold value. Values that are bigger than the threshold are in the area of high risk.

As presented within the related work in section 2, different threshold derivation approaches exist. To allow the comparison, the threshold values have to be derived using the same benchmark data, the software metric tools with coincidental definitions of implemented software metrics, and, finally, using the same threshold derivation approach. Therefore, comparison of our results with threshold values provided by Ferreira et al. [10]

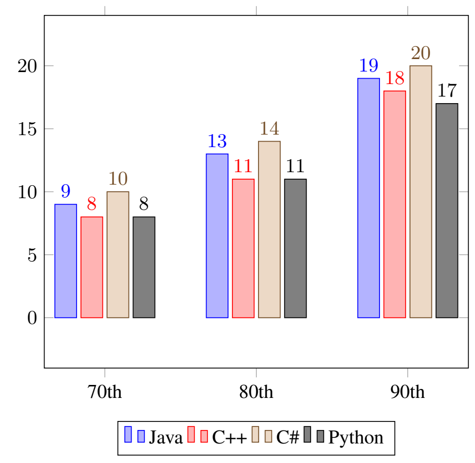
or Filo et al. [12] in a meaningful way is not possible, due primarily to use of different software metrics' definitions, followed by varying input data.

## 4.1. Comparison of derived threshold values

Figures 1, 2, 3 and 4 present threshold values for the 70th, 80th and 90th percentiles of the same software metrics for different programming languages. A visual comparison of threshold values for Java, C++, C# and Python is enabled with this. Furthermore, since the approach for the threshold derivation is based on the frequency of values within the software, the results also indicate the structure of software written in the four selected programming languages.



**(a)** CountLineCode



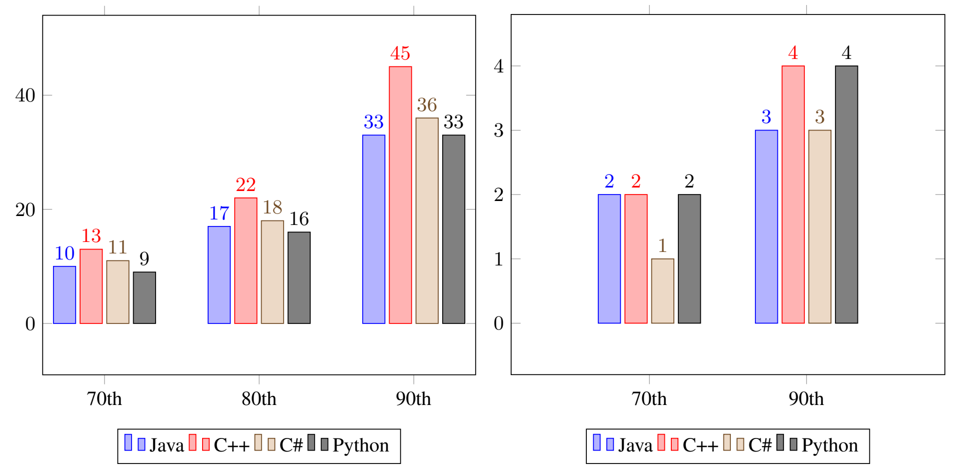
**(b)** AvgLineCode

**Fig. 1.** Threshold values of the software metrics *CountLineCode* and *AvgLineCode*

Figure 1 illustrates risk areas and threshold values of the 70th, 80th and 90th percentiles for software metrics *CountLineCode* and *AvgLineCode* measuring lines of code in a software class. Axis *x* shows programming languages, and axis *y* threshold values. In every figure, there are three lines: green, representing the 70th percentile, orange, representing the 80th percentile, and red, representing the 90th percentile. Three lines form four risk areas, while connecting values of the same percentile value among programming languages. The green color presents a low risk area (LR), orange presents a moderate risk area (MR), light red color stands for a high risk area (HR), and red presents a very high risk area (VHR).

As indicated in Figure 1a, the threshold determining very high risk is the highest in the C# programming language, whereas the smallest is within Python. The same ratio is also between thresholds formed using the 80th and 70th percentiles. Figure 1b plots the threshold values for the average size of methods in a class. The values are closer in comparison to the metric measuring lines of code in a class, but still, values vary. The 90th percentile is again the highest for C# and the lowest for Python, whereas the values for

Java and C++ are in between. The derived values show that the most extensive software classes can be found in the C# programming language, followed by C++, Java and Python classes. Given the small difference in the average size of methods within a class, we can conclude that software classes written in C# possess more methods than classes developed in the Python programming language.



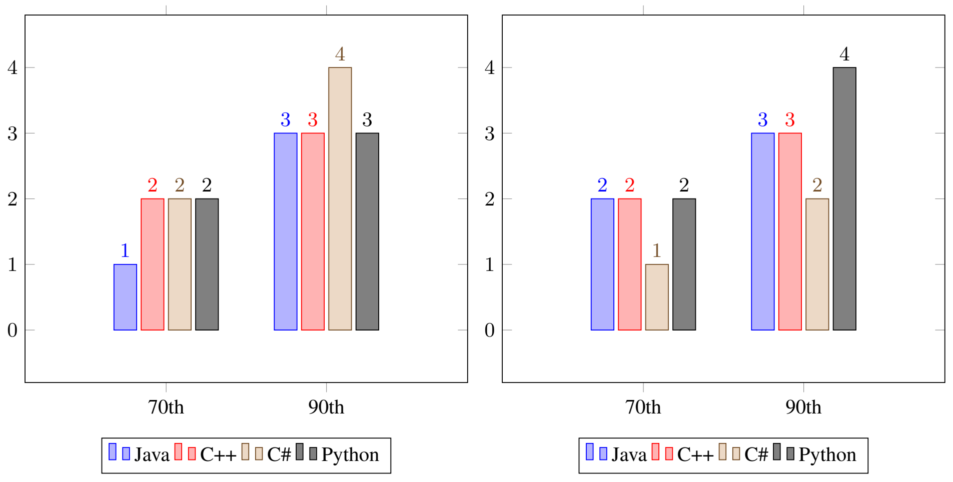
**(a)** SumCyclomatic **(b)** AvgCyclomatic

**Fig. 2.** Threshold values of the software metrics *SumCyclomatic* and *AvgCyclomatic*

Figure 2 presents the threshold values of the software metrics *SumCyclomatic* and *AvgCyclomatic*. Figure 2a visualizes the thresholds for the sum of cyclomatic complexities for all the methods in a class. The highest threshold value is derived for the programming language C++, followed by C#, Java and Python. A noticeable leap can be detected in C++, whereas in other programming languages, the threshold values are rising more gradually. If we consider a number of methods in classes expressed with software metrics *CountLineCode* and *AvgLineCode*, we can conclude that methods written in Python have the highest cyclomatic complexity, whereas the smallest complexity is present in methods developed in C#. This is also confirmed with threshold values for the metric *AvgCyclomatic* in Figure 2b, presenting the average value of the methods in a class. The *AvgCyclomatic* is also one of the metrics where derived threshold values are very close. Because of this, the 80$^{th}$ percentile was excluded, and only three risk areas were formed.

Figure 3 presents threshold values for the software metrics *MaxNesting* and *MaxInheritanceTree*. Figure 3a illustrate the thresholds of a software metric measuring the maximum nesting level in a class which affects class complexity. The 70$^{th}$ and 90$^{th}$ percentiles were included, forming three risk areas. As can be seen, the 90$^{th}$ percentile is the highest in C# and the lowest, but coinciding, for Java, C++ and Python. Another software metric that has a thresholds value that is only defined for three areas is *MaxInheritanceTree*, presented in Figure 3b. Based on statistical properties, it is very similar to the metric

*MaxNesting*. The 90th percentile is the highest within Python and the lowest in C#, meaning that inheritance hierarchy is the deepest in software projects developed in Python.

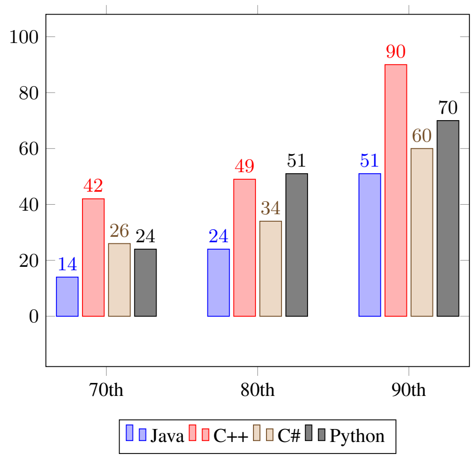


**(a)** MaxNesting **(b)** MaxInheritanceTree

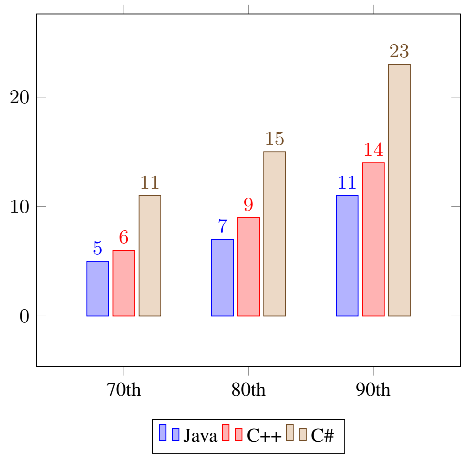
**Fig. 3.** Threshold values of the software metrics *MaxNesting* and *MaxInheritanceTree*

Closely connected to the mentioned metric is also the metric *CountDeclMethodAll*, counting methods in the classes, including inherited ones. The threshold values are presented in Figure 4a. With a 90th percentile value, the C++ threshold set is the biggest, followed by values derived for Python, C# and the Java programming language. If we connect the findings to the determined number of methods based on software metrics *CountLineCode* and *AvgLineCode*, we can see that the ranking by values is different, since the metric *CountDeclMethodAll* also considers inherited methods. As presented in Figure 4a, the highest number of methods can be detected in C++, which is due to a bigger inheritance hierarchy, as presented in Figure 3b. A high number of methods, according to the metric *CountDeclMethodAll*, can also be found with Python classes, though they have the smallest metric value, based on lines of code. Again, this is due to a deeper inheritance hierarchy. On the other hand, Java classes have fewer methods than classes developed in Python, according to *CountDeclMethodAll*, but based on the number of lines of code, the case is different. Since the inheritance hierarchy for Java is smaller, this is the logical conclusion.

The threshold values for the software metric *CountClassCoupled* are presented in Figure 4. The mentioned metrics were not calculated for Python, since the Understand tool [35] does not support that calculation. Therefore, the results are only presented for Java, C++ and C#. Figure 4b presents threshold values for metric measuring coupling with other classes. The threshold defining the area of the very high risk is the biggest for C# classes. The lowest is the threshold value for Java, whereas the programming language C++ is in between. Thresholds indicate that C# classes are the most coupled with others, which can be related to a large number of lines of code, as seen in Figure 1. On the other



**(a)** CountDeclMethodAll



**(b)** CountClassCoupled

**Fig. 4.** Threshold values of the software metrics *CountClassCoupled* and *PercentLackOfCohesion*

hand, the coupling is lowest for Java classes. Another aspect that can impact the use of coupling is also the age of the projects. Since the C# programming language is much younger that C++, the age could influence the threshold values.

The *PercentLackOfCohesion* metric measures the lack of cohesion in classes, and is presented with percentages. The threshold is presented only with one value, that divides the area into low and high risk, since the threshold was not derived based on percentiles. The values can be seen in Table 6. The defined threshold values again vary between programming languages, and are the highest for C++, where classes that exceed the value 90 present the high risk of containing irregularities. Within C#, the threshold value is the highest, while the values indicate that the C++ classes are least cohesive, which could be connected to the high cyclomatic complexity of classes and methods with a large number of lines. On the other hand, classes in C# are also large, but with a lower cyclomatic complexity, which is reflected in class cohesion. The connection to cyclomatic complexity can also be confirmed for the Java programming language, where cohesion is better and complexity lower.

As can be concluded based on the presented analysis, the threshold values of software metrics vary between different programming languages. Therefore, they have to be calculated for each programming language separately.

## 5. Limitations

In this research, some limitations and potential threats to validity arise. They are presented here.

We limited ourselves to object-oriented programming languages and software metrics supported by the used tool. The results may be affected by the tool used for collecting metric values and the corresponding implementation of software metrics. To reduce the threat, a single tool was used throughout the entire research, and for all four of the programming languages. In this way, software metrics were calculated in the same way, regardless of the programming language.

The results can also be affected by the approach used for deriving the threshold values of software metrics. With the use of only a single approach for all the metrics and all the programming languages, the risk of providing inconsistent results was limited. Another threat surrounds the benchmark data used for derivations. To limit the impact, the data set was collected in a transparent and systematic way, covering a broad scope of different properties. Also, the size of the benchmark data was determined based on related work and good practices.

The definitions of software metrics present a limitation within the research. The validation of software metrics was not a part of the presented study.

## 6. Conclusion

Quantification with software metrics is important, especially when we make decisions related to software quality [1,9], thereby knowing that the reliable thresholds are crucial. Within the presented empirical study, threshold values were derived for nine software metrics for four object-oriented programming languages, namely Java, C++, C# and Python. Using the replicated threshold derivation approach and proposed adjustments, threshold values were derived considering challenges arising in the software metrics domain. Since the approach uses benchmark data, the latter were collected systematically and transparently, allowing repeatability and supplementation. For each programming language, a suite of 100 software projects was selected, which is, according to related work, an optimal number. Input values were gathered using a single software metric tool, and threshold values were provided using a single threshold derivation approach by following well-defined steps.

The main research question driving the presented study was *if software metric threshold values vary between different object-oriented programming languages*. By this, we could provide information about whether thresholds have to be derived for each programming language separately, or if a single threshold can be applied to all programming languages. Thresholds derived for a particular software metric were analyzed and compared to provide the answer. Based on the findings, we can conclude that threshold values for the same software metric vary among different programming languages. This can be attributed to different structural properties for programming languages, and established practices used in a specific community. Therefore, the derivation for each programming language has to be done separately.

In future work, we plan to use the threshold derivation process to provide threshold values for other programming languages, and expect to derive threshold values for software metrics on different levels, i.e. the method and file levels. Also, we will analyze the rules and properties of the different programming languages, in order to explain the reasons for the differences. In addition, we plan to study different factors impacting derived thresholds within each programming language.

**Acknowledgments.** This work was supported by the Slovenian Research Agency (SRA) under The Young Researchers Program (SICRIS/SRA code 35512, RO 0796, Program P2-0057).

## References

1. Alves, T.L., Ypma, C., Visser, J.: Deriving metric thresholds from benchmark data. In: 2010 IEEE International Conference on Software Maintenance (2010)
2. Arar, Ö.F., Ayan, K.: Deriving thresholds of software metrics to predict faults on open source software: Replicated case studies. Expert Systems with Applications 61, 106 – 121 (2016)
3. Benlarbi, S., Emam, K.E., Goel, N., Rai, S.: Thresholds for object-oriented measures. In: Proceedings 11th International Symposium on Software Reliability Engineering (ISSRE 2000) (2000)
4. Beranič, T., Heričko, M.: Approaches for software metrics threshold derivation: A preliminary review. In: Sixth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications SQAMIA 2017, Proceedings (2017)
5. Beranič, T., Podgorelec, V., Heričko, M.: Towards a reliable identification of deficient code with a combination of software metrics. Applied Sciences 8(10) (2018)
6. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20(6), 476–493 (Jun 1994)
7. Concas, G., Marchesi, M., Pinna, S., Serra, N.: Power-laws in a large object-oriented software system. IEEE Transactions on Software Engineering 33(10), 687–708 (Oct 2007)
8. Dósea, M., Sant'Anna, C., da Silva, B.C.: How do design decisions affect the distribution of software metrics? In: Proceedings of the 26th Conference on Program Comprehension (2018)
9. Fenton, N.E., Neil, M.: Software metrics: Roadmap. In: Proceedings of the Conference on The Future of Software Engineering (2000)
10. Ferreira, K.A., Bigonha, M.A., Bigonha, R.S., Mendes, L.F., Almeida, H.C.: Identifying thresholds for object-oriented software metrics. Journal of Systems and Software 85(2), 244 – 257 (2012)
11. Field, A.: Discovering Statistics Using IBM SPSS Statistics. Sage Publications Ltd., 4th edn. (2013)
12. Filó, T.G.S., da Silva Bigonha, M.A., Ferreira, K.A.M.: A catalogue of thresholds for object-oriented software metrics. In: First International Conference on Advances and Trends in Software Engineering (2015)
13. Fontana, F.A., Ferme, V., Zanoni, M., Yamashita, A.: Automatic metric thresholds derivation for code smell detection. In: IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics (2015)
14. Gerlec, C., Rakić, G., Budimac, Z., Heričko, M.: A programming language independent framework for metrics-based software evolution and analysis. Computer Science and Information Systems 9(3), 1155–1186 (2012)
15. Gil, J.Y., Lalouche, G.: When do software complexity metrics mean nothing? – when examined out of context. Journal of Object Technology 15(1), 2:1–25 (Feb 2016)
16. IBM: Spss. https://www.ibm.com/analytics/data-science/predictive-analytics/spss-statistical-software (2018), accessed: 2. 2. 2018
17. ISO/IEC/IEEE 24765:2017: ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary pp. 1–541 (2017)
18. Kitchenham, B.: What's up with software metrics? – a preliminary mapping study. Journal of Systems and Software 83(1), 37 – 51 (2010)
19. Lanza, M., Marinescu, R.: Object-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems. Springer-Verlag Berlin Heidelberg (2006)
20. Lavazza, L., Morasca, S.: An empirical evaluation of distribution-based thresholds for internal software measures. In: Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering (2016)

21. Lima, P., Guerra, E., Meirelles, P., Kanashiro, L., Silva, H., Silveira, F.F.: A metrics suite for code annotation assessment. Journal of Systems and Software 137, 163 – 183 (2018)
22. Lincke, R., Lundberg, J., Löwe, W.: Comparing software metrics tools. In: Proceedings of the 2008 International Symposium on Software Testing and Analysis (2008)
23. Lochmann, K.: A benchmarking-inspired approach to determine threshold values for metrics. SIGSOFT Softw. Eng. Notes 37(6), 1–8 (Nov 2012)
24. Lorenz, M., Kidd, J.: Object-oriented Software Metrics: A Practical Guide. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1994)
25. Martin, R.C.: Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edn. (2009)
26. MathWave Technologies: Easyfit. http://www.mathwave.com (2004–2018), accessed: 20. 12. 2017
27. McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering SE-2(4), 308–320 (Dec 1976)
28. Mori, A., Vale, G., Viggiato, M., Oliveira, J., Figueiredo, E., Cirilo, E., Jamshidi, P., Kastner, C.: Evaluating domain-specific metric thresholds: An empirical study. In: Proceedings of the 2018 International Conference on Technical Debt (2018)
29. Nuñez-Varela, A.S., Pérez-Gonzalez, H.G., Martínez-Perez, F.E., Soubervielle-Montalvo, C.: Source code metrics: A systematic mapping study. Journal of Systems and Software 128, 164 – 197 (2017)
30. Oliveira, P., Valente, M.T., Lima, F.P.: Extracting relative thresholds for source code metrics. In: 2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE) (2014)
31. Oliveira, P., Lima, F.P., Valente, M.T., Serebrenik, A.: RTTool: A tool for extracting relative thresholds for source code metrics. In: Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution (2014)
32. R Core Team: R: A language and environment for statistical computing. https://www.R-project.org (2017), accessed: 13. 12. 2017
33. Rakić, G., Budimac, Z., Savić, M., Ivanović, M.: Towards the formalization of software measurement by involving network theory. In: SQAMIA (2015)
34. Remencius, T., Sillitti, A., Succi, G.: Assessment of software developed by a third-party: A case study and comparison. Information Sciences 328, 237–249 (2016)
35. Scientific Toolworks Inc.: Understand$^{\text{TM}}$. https://scitools.com (1996–2018), accessed: 18. 8. 2018
36. Scientific Toolworks, Inc.: Understand, User Guide and Refrence Manual (2017)
37. Sharma, A., Dubey, S.K.: Comparison of software quality metrics for object-oriented system. International Journal of Computer Science & Management Studies (IJCSMS) 12, 12–24 (2012)
38. Shatnawi, R., Althebyan, Q.: An empirical study of the effect of power law distribution on the interpretation of oo metrics. ISRN Software Engineering 2013 (March 2013)
39. SourceForge: https://sourceforge.net (2017), accessed: 16. 8. 2017
40. Tempero, E., Anslow, C., Dietrich, J., Han, T., Li, J., Lumpe, M., Melton, H., Noble, J.: Qualitas corpus: A curated collection of java code for empirical studies. In: 2010 Asia Pacific Software Engineering Conference (APSEC2010). pp. 336–345 (Dec 2010)
41. Vale, G.A.D., Figueiredo, E.M.L.: A method to derive metric thresholds for software product lines. In: 2015 29th Brazilian Symposium on Software Engineering (2015)
42. Vale, G., Fernandes, E., Figueiredo, E.: On the proposal and evaluation of a benchmark-based threshold derivation method. Software Quality Journal pp. 1–32 (May 2018)
43. Veado, L., Vale, G., Fernandes, E., Figueiredo, E.: TDTool: Threshold derivation tool. In: Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (2016)

44. Yamashita, K., Huang, C., Nagappan, M., Kamei, Y., Mockus, A., Hassan, A.E., Ubayashi, N.: Thresholds for size and complexity metrics: A case study from the perspective of defect density. In: 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS) (2016)
45. Zhang, F., Mockus, A., Zou, Y., Khomh, F., Hassan, A.E.: How does context affect the distribution of software maintainability metrics? In: 2013 IEEE International Conference on Software Maintenance. pp. 350–359 (Sept 2013)
46. Zou, Y., Kontogiannis, K.: Migration to object oriented platforms: a state transformation approach. In: International Conference on Software Maintenance, 2002. Proceedings. (2002)

**Tina Beranič** received a PhD degree in computer science and informatics from the University of Maribor in 2018. She is a Teaching Assistant and a Researcher at the Faculty of Electrical Engineering and Computer Science, University of Maribor. Her research interests are in the area of software quality, especially the domain of software metrics and software metrics thresholds. She is working on their involvement in the quality assessment process.

**Marjan Heričko** received a PhD degree in computer science and informatics from the University of Maribor in 1998. He is currently a Full Professor at the Faculty of Electrical Engineering and Computer Science, University of Maribor, where he is also the Head of the Institute of Informatics. His main research interests include all aspects of IS development with an emphasis on software engineering, software process improvement, software metrics, and process modeling.

*Received: October 12, 2018; Accepted: November 15, 2019.*

Computer Science and Information Systems 17(1):205–227 https://doi.org/10.2298/CSIS181220019V

# Regression Verification for Automated Evaluation of Students Programs

Milena Vujošević Janičić[1] and Filip Marić[1]

University of Belgrade, Faculty of Mathematics, Studentski trg 16
11000 Belgrade, Serbia
{milena,filip}@matf.bg.ac.rs

**Abstract.** Regression verification is a form of software verification based on formal static analysis of code, which is used, since recently, in several domains. In this paper we examine potentials of using it in one novel domain — in automated evaluation of students programs. We propose an approach that provides precise assessment of functional correctness of student programs (while it does not address nor affect the teaching methodology). We describe our open-source, publicly available implementation of the approach, which is built on top of the compiler infrastructure LLVM and the software verification tool LAV. The results of evaluating the proposed approach on two real-world corpora of student programs and on a number of classic algorithms show that the proposed approach can be used as a precise and reliable supplementary technique in grading of student programs at introductory programming courses, algorithms courses and programming competitions.

**Keywords:** software verification, regression verification, automated evaluation of student programs, computer-supported education

## 1. Introduction

Despite many successful applications of software verification techniques, their potential is still to be explored in a number of new application domains. One domain are programming courses where automated evaluation of student programs is becoming progressively important. Namely, computer science is recognized as a fundamental field which is delivered in both universities and schools [69]. Also, the number of students enrolled at programming courses has rapidly grown over the last years [3]. Everyone benefits from automated evaluation [57,79]: the teachers get help in the grading process, while immediate feedback helps students in acquiring knowledge. The importance of automated evaluation is even more significant in the context of online learning where the adequate assessment is recognized as a challenging problem since contact with a teacher is minimal or even non-existent [61,73], while the number of students also grows quickly [56].

It is very important to provide a high quality, objective, precise and reliable automated evaluation [55]. Automated grading must (i) correctly classify correct and incorrect student solutions, (ii) correctly explain mistakes that students make, and (iii) run efficiently in practice [37]. There are different approaches for automated evaluation of student programs [2,35,58], considering many important aspects (e.g. functional correctness, code readability, modularity, complexity, efficiency). Various teachers and universities have various grading policies depending on such factors. In most cases, functional correctness is very highly valued [35] and in some educational settings is even essential. Such settings are commonly encountered at the university level at programming courses for future computer science majors and software engineers. Also, functional correctness is traditionally a must at IOI and ACM style programming competitions[1], which usually deal with problems that are very similar to problems taught at university level algorithms courses. Such algorithmic, competition style problems are also highly valued for employment in the high-end software companies and are usually asked at job interviews. In settings where students are required to produce fully functionally correct code and where subtle errors and hidden bugs are not allowed, attention must be put on all corner cases and it should be ensured that the grading process takes them into account.

Classifying correct and incorrect solutions of algorithmic problems is usually based on automated testing [17] and grading is performed solely by thorough testing on a number of test-cases. For example, this holds for online judges — web-platforms devoted to training for programming contests and interviews [23,31,50,51,65,72]. However, assessing functional correctness only by testing may give a misleading confidence since it may be error prone: the obtained results are directly influenced by the choice of test cases [78]. The problem is that the test cases are usually designed according to the expected solutions, while the teacher cannot predict all possible solutions and all important paths through a student solution. Moreover, no matter how well test cases are designed, testing cannot guarantee functional correctness [16]. Therefore, if a reliable automated evaluation is needed, it is necessary to apply some more involved techniques. A more promising choice are software verification techniques and we propose a verification based approach for improving classification of correct and incorrect solutions.

In this paper, we propose assessing functional correctness of students solutions by checking equivalence with teacher solutions. We are interested in showing equivalence of algorithmic problems that usually have short solutions, but can be very hard and complicated, thus their correctness can often be at stake. We describe how to apply formal static software verification techniques for assessing different kinds of equivalence of two programs and we focus on *regression verification* techniques. Development of regression verification techniques is often guided by applications in various industrial domains. The existing algorithms are advanced and there are still no general purpose implementations that are publicly available. Also, in the context of automated evaluation of programming assignments, it is necessary to adjust solutions in a way that makes these algorithms applicable, which also contains some nontrivial steps. Therefore, our work aims at enabling application of regression verification techniques in automated evaluation of programming assignments. We describe characteristics of programs that can be evaluated this way. We provide an open-source implementation of necessary transformations for automating this process. We present lessons learned from applying regression verification on three different corpora: a corpus of student solutions from an introductory programming course for computer science majors, a corpus of solutions submitted during national programming competitions, and a corpus of classic algorithms that are usually taught at algorithms courses. We show that, by our approach, functional correctness of significant amount of programs in introductory and algorithms courses can be automatically proved. We also show that our approach makes a good supplementary technique, aimed at the best solutions that successfully passed testing: it can reveal very subtle problems and point stu-

[1] IOI: `http://ioinformatics.org` , ICPC: `https://icpc.baylor.edu/`

dents to errors that they are not aware of. In the context of programming competitions, it can break ties and help differentiating the very best few competitors that qualify for next rounds. In some situations verification can even fully replace testing, eliminating the effort necessary to prepare tests.

**Overview of the paper.** Section 2 contains information about related work. Section 3 introduces our approach and describes its implementation. Section 4 gives results of experimental evaluation of the proposed approach with discussion of quantitative and qualitative analysis of capabilities of the approach. It also discusses possible threats to validity. In Section 5 we compare the proposed approach with other related approaches and tools. Section 6 gives conclusions and outlines possible directions for future work.

## 2. Related work

In this section we give a brief overview of related approaches and tools, both in the field of software verification and in automated evaluation of programming assignments.

**Software verification and automated bug finding.** Automated software verification tools aim to automatically check correctness properties of a given program or to find violations to some common features (the latter is known as automated bug-finding) [9]. There are different automated approaches [13,15,39] and there is a variety of tools based on these approaches like PEX [71], JPF [75], KLEE [10], CBMC [12], LAV [77]. CBMC and LAV are general purpose tools for statically verifying user-specified assertions and locating bugs such as buffer overflows, pointer errors and division by zero. CBMC is state of the art bounded model checker for C/C++ programs. LAV is primarily aimed at analysing programs written in the programming language C, but for the purpose of this work we have extended LAV with some constructs of C++ (used in the context of programming competitions, and present in our corpus).

**Equivalence checking.** Functional correctness of a program can be formulated in terms of precise formal specifications [32,43]. Also, it can be formulated in terms of the behavior of another program: two programs are equivalent if they exhibit the same behavior in all relevant aspects on all input values [26]. This includes checking termination and complexity of computation, but often only equivalence of outputs is considered [25]. The notion of correctness in this case has several positive aspects: it is not necessary to formulate a specification and, in general, checking equivalence of two programs is less computationally demanding than functional verification with respect to a formal specification [67]. Checking equivalence of two programs was considered already in 1960s [32], but the progress has been limited and not always practically applicable. Recent approaches introduced new possibilities [20,27]. There are different variations of program equivalence [25]. Programs are *partially equivalent* if any two terminating executions which start from equal inputs produce equal outputs. Another, weaker, notion of equivalence is *k-equivalence* — programs are $k$-equivalent if any two executions where loops and recursions have at most $k$ iterations or calls, which start on equal inputs, produce equal outputs. The problem whether two programs are partially equivalent is an undecidable problem [68], while the problem whether two programs are $k$-equivalent (for some specific $k$, assuming that finite variable-domains are used) is decidable [25].

**Regression verification.** Applying testing to check whether two similar programs are equivalent is widely and intensively used in software development and is called regression testing [53]. *Regression verification* [20,67] attempts to achieve the same goals, but using techniques from formal verification. Here, checking equivalence means formally proving a mathematical statement about two programs that usually corresponds to some weaker form of equivalence. If successful, regression verification gives higher reliability since it guarantees full coverage [27]. Also, that it does not require additional expenses to develop and maintain a test suite. Since the problem of determining partial equivalence is undecidable [68], automating this process is challenging. Development of regression verification techniques is often guided by the application in different concrete areas, like security verification applications [4,62], multimedia systems [74], backward compatibility and refactoring [80], cryptographic algorithms [8,59], and hardware design [34]. General purpose automated regression verification techniques [6,20,27] are developed for large scale systems. These techniques consist of two steps: efficiently identifying functions that are affected by changes, and proving functional equivalence of these functions.

**Functional correctness in automated evaluation.** Automated testing is the most common way of evaluating student programs [17]. Test cases are usually supplied by the teacher and/or randomly generated [47]. Testing is used as an evaluation component of a number of web-based submission and evaluation systems [11,18,23,31,33,50,51,65,72]. Aside from checking functional correctness, testing can also be used for analysing efficiency, memory violations and run-time errors [1]. Software verification techniques are getting more commonly used in automated evaluation, usually for automated bug finding or for automated test case generation [36,37,71,78]. One formal approach for assessing functional correctness of student solutions, is based on rewriting techniques [40]. In this approach, it is necessary to write a formal specification of a desired solution.

**Other important aspects in automated evaluation.** There are other important aspects that are impossible or difficult to test or to be assessed by verification techniques, but that have to be taken into account in precise and high quality evaluation. For example, these are coding style, the design of the program, modularity, performance issues and the algorithm used. Therefore, other techniques are required for their assessment [29,52,54,70,78]. These techniques usually compare a predefined solution to the student solution. New approaches emphasize the importance of generating useful feedback for students [24,28,29,38,41,48,60]. Usually, the feedback is generated by failed test-cases or by peer-feedback [19,42,46]. Some approaches use both reference implementation and error model consisting of potential corrections to errors that students might make [64] and with this additional information are capable of making feedback that suggests possible corrections to incorrect student solution. Another kind of feedback is generated by computing behavioral similarity between two programs [45]. In this case, different metrics are used to calculate similarity to the model solution, which is then used as a measure of student progress. Machine learning techniques can be used for syntactically classifying similar solutions [55] or for clustering similar solutions by static and dynamic analysis [24]. The feedback is then generated by the teacher but only for each group of solutions.

## 3. Proposed approach and its implementation

In this section we discuss our open-source implementation based on regression verification techniques which is implemented on top of the software verification tool LAV [77], the LLVM system [44] and its C-language front-end Clang. We describe its implementation, as regression verification techniques are still rather new and advanced, and there are no implementations that are publicly available. Although regression verification is originally used for showing equivalence between two versions of the evolving program, we shall use it to show equivalence between the student and the teacher solution. The same techniques could be used for showing equivalence between different student solutions. The techniques described in this section and parts of our implementation can be adapted to work with other underlying verification systems by making an extension for specifying that some function calls should be encoded as uninterpreted functions calls.

Finding parts of code that are potentially equivalent is an important task for regression verification tools. There are different techniques for solving it (based on the analysis of control flow graphs and function names that preserve equivalent in different versions of programs [6,20,27]). In our setting, that problem is simple as corresponding functions are the teacher's and the student's solutions.

## 3.1. Regression verification in LAV

The input to the system LAV is a C program that may contain assertions, which can be accompanied by some assumptions (given width `assert`/`assume` function calls). Such assumptions are used to limit verification only to the cases allowed by the problem specification. User can put limits on the input variables in a way that subtle details get ignored or important preconditions are enforced (e.g., that some array is sorted). By enforcing additional assumptions, verification can be done against an arbitrary input specification.

In regression verification we try to prove the equivalence between the two solutions that are encoded by different functions that share the same interface. The implementation of these functions can be quite different (concerning used algorithms, computation that can be split into different auxiliary functions, etc.). Figure 1 contains different implementations of the function for finding maximum of three given numbers (these are all real-world examples, taken from our corpus described in Section 4.1, and reflect the possible diversity in solutions even for a very simple problem). To check equivalence of the functions `maxA` and `maxB` from Figure 1 using the system LAV, it is sufficient to verify the program illustrated in Figure 2. Calling the `assert` function in this program refers to the equality check of return values of these two functions (for arbitrary input values). Similarly, the function `maxC` can be shown to be equivalent to `maxA` and `maxB`. However, the function `maxD` contains a subtle bug and is not equivalent to the previous three ones. Since the C language does not allow returning arrays as function results, checking equivalence of functions that modify arrays is done by multiple assertions (Figure 2).

To verify an assertion, LAV encodes the asserted expression as a first-order logic formula and checks its validity by an underlying SMT solver [7]. We will focus on integer variables that are modelled either by the theory of linear arithmetic (LA), or by the theory of bit-vector arithmetic (BVA). Although there are important semantic differences between LA and BVA, in the context of education some of these differences are not relevant (for example, at introductory level, overflows/underflows are usually not considered). LA is very efficient, but does not support many operators that BVA supports and that are used in C-programs. For efficiency reason, BVA will be used only when that is necessary. We will focus on programs containing loops and/or recursive functions, since their treatment is the most delicate aspect in verification. Since loops are not supported in SMT formulas, functions have to be transformed into some loop-free form. We will consider

```
int maxA(int x, int y, int z) {          int maxB(int a, int b, int c) {
   int m = x;                               int max;
   if(y > m) m = y;                         max = a;
   if(z > m) m = z;                         if (b>max && b>c) max=b;
   return m;                                else if(c>max) max=c;
}                                           return max;
                                          }

int maxC(int i, int j, int k) {          int maxD(int o, int p, int q) {
   int max;                                 if(o>p && o>q)
   if(i>j && i>k) max= i;                       return o;
   else if(j>k) max = j;                    else if(p>o && p>q)
   else max = k;                                return p;
   return max;                              else
}                                               return q;
                                          }
```

**Fig. 1.** Different implementations for determining the maximum value

```
                             #include "modifyAB.h"
#include "maxAB.h"           int main() {
int main() {                   int i; char s[MAX], t[MAX];
  int a, b, c;                 scanf("%s",s);
  scanf("%d%d%d",              for(i = 0; s[i]; i++)
        &a, &b, &c);             assume(t[i] == s[i]);
  assert(maxA(a,b,c) ==        resultA = modifyA(s); resultB = modifyB(t);
         maxB(a,b,c));         assert(resultA == resultB);
  return 0;                    for(i = 0; s[i]; i++)
}                                assert(t[i] == s[i]);
                               return 0;
                             }
```

**Fig. 2.** Checking equivalence of two functions: (left-hand side) functions from the Figure 1 and (right-hand side) functions that modify contents of arrays

two different techniques for loop elimination: (i) loop unrolling for proving $k$-equivalence (ii) transforming loops into recursive functions and then using uninterpreted functions to express the inductive hypothesis [27,67].

**$K$-equivalence by loop unrolling.** Functions that contain loops with a fixed upper bound can be transformed into equivalent functions that do not contain loops. However, unrolling loops a large number of times may introduce complex formulas that cannot always be efficiently reasoned about. Checking equivalence of functions with arbitrary loops is a major challenge and is generally not solvable. Therefore, we must resort to using some approximation. For example, instead of proving equivalence of two functions we can try proving their $k$-equivalence. In such case, loops are unrolled $k$ times, for some given value $k$. Figure 3 shows a loop that is unrolled $k = 3$ times. When proving $k$-equivalence, the choice of an appropriate value for $k$ is very important. Higher values of $k$ are giving a higher level of confidence to the code under evaluation, but increasing $k$ can introduce scalability issues. On the other hand, some verification tools rely on common experience that many errors can be discovered in only one loop iteration [5,21]. Note that the number $k$ often corresponds to the length of the input series for which the algorithm is verified, although this need not be the case always (for example, in binary search, unrolling loop for $k$ times guarantees the correctness for the arrays with at most $2^k$ elements). In our experiments, we usually used $k = 5$, as for this value the analysis was efficient and results showed to be reliable. We discuss this choice in more details in Section 4.1. Similar to loop unrolling is the recursive function call unrolling. However, recursive function unrolling

```
float mean_valueA(int a[], int n) {
  float s = 0;
  int i;
  for (i=0; i<n; i++)
    s += a[i];
  return s/n;
}

float mean_valueB(int a[], int n) {
  int i;
  float m;
  m = i = 0;
  while(i < n)
    m = m + a[i++];
  m = m/n;
  return m;
}
```

```
float mean_valueA_k3(int a[], int n) {
  float s = 0;  int i;
  i = 0;
  if(i < n) {
    s += a[i];
    i++;
    if(i < n) {
      s += a[i];
      i++;
      if(i < n) {
        s += a[i];
        i++;
      }
    }
  }
  return s/n;
}
```

**Fig. 3.** Calculating the mean value of an array (left-hand side), unrolling $k = 3$ times a loop of the function `mean_valueA` (right-hand side)

can lead to significantly slower verification, due to introduced stack-frame modeling, and due to exponential code growth when there is more than one recursive call.

**Partial equivalence by uninterpreted functions.** Instead of loop unrolling, in some situations we can use inductive reasoning to prove partial equivalence between the two functions by using uninterpreted functions to model inductive hypothesis [27]. To succeed in proving partial equivalence by uninterpreted functions in programs that contain loops it is necessary to have solutions where entry point, exit condition, and loop invariant are the same (while the body of the loop can differ).

**Preprocessing.** There are several constructs in C that complicate elimination of loops (e.g., `break`, `continue` and `return`), and in the preprocessing phase we automatically transform the program to eliminate such constructs. Also, we transform all loops to the `while` loop. Removing `return` statements is illustrated in Figure 4. If the `return` statement occurs within a nested loop, the transformation is applied once for each loop, starting from the innermost loop. This transformation introduces a special value `RET_UNDEF` that cannot occur as the return value of the function. Similar transformations are applied to eliminate `break` and `continue` statements.

```
while(<cond>) {
   ....
      <return> <val>;
   ....
}
```

```
<retvar> = RET_UNDEF;
while (<cond> && <retvar> == RET_UNDEF) { ...
      <retvar> = <val>;
   if (<retvar> == RET_UNDEF)
...}
if (<retvar> != RET_UNDEF)
   return <retvar>;
```

**Fig. 4.** Preprocessing transformations: `return` statement elimination

**Introducing uninterpreted functions.** Consider the functions given on top of Figure 5 (also taken from our corpus). After preprocessing the next step is to transform loops into recursive functions (as illustrated in the middle of Figure 5). An important requirement (that is often satisfied) is that the loop changes exactly one variable that is alive after the loop (its value is read and used before it is eventually changed). In the function idx_minA, the variable min is such a variable and in the function idx_minB, the variable idx is such a variable. Then, the recursive equivalent of the loop will be a function whose return value will be exactly that variable. The function can have many input parameters (the variables that are accessed within the loop, except the ones that are declared in the loop or are always assigned a value before their value is read). Equivalence of the recursive functions can be proved by induction on the number of recursive calls made during their execution. The base case is when no recursive calls are made. As the induction hypothesis we can assume that the statement will hold for recursive calls i.e., that recursive calls return the same values. Under that assumption and the definition of the recursive functions it should be proved that the statement holds i.e., that the functions return the same values in the case when recursive calls are made (in the code on Figure 5, that is when `i < n`). The crucial part of the technique is to encode such induction hypothesis by replacing recursive calls by a call to an uninterpreted function (as illustrated at the bottom of Figure 5). After those replacements, we are left with a loop-free and recursion-free functions that can be shown equivalent using the techniques for loop-free, recursion-free programs. Once the recursion is removed, there is no need to have auxiliary functions representing loops, thus, for simplicity, they can be inlined back (as illustrated on the bottom of Figure 5). A more complicated example from our corpus is given in Figure 6. An important question is how to order parameters of uninterpreted functions (since solutions must use the same order of parameters). The names of the variables, and the order of their declarations can vary between alternative solutions, therefore some kind of semantic matching between the corresponding variables is needed. Currently, to solve this problem, our transformation uses a heuristic: parameters are first ordered by their type, and then by their name. In most cases students use canonical variable names (e.g., min for a minimum value), and the heuristic works. However, when it fails, all possible combinations of parameter ordering can be checked (usually there are not many parameters).

## 3.2. Interpreting results of regression verification

When using regression verification in evaluation process, it is crucial to correctly interpret obtained results and to understand relationship between different evaluation techniques.

**Function calls.** Inlining is the only fully precise technique for modeling function calls. Other techniques incur loss of information about the exact program behavior. Therefore, when other techniques are used, it might not be possible to prove the equivalence.

**$K$-equivalence vs partial equivalence.** The fact that functions are $k$-equivalent for some value $k$, does not guarantee that these functions are partially equivalent, or even $k$-equivalent for some larger value $k$. It only guarantees that these functions will give same outputs for each input value, if restricted to $k$ or less loop iterations. However, in our experimental evaluation in both our corpora, we did not find two functions that were $k$-equivalent and not partially equivalent (with exception to the functions that used unmodelled library function calls, which were detected independently). This is partially due to the fact that these programs were also thoroughly tested and checked for bugs before checked for $k$-equivalence. However, if two functions are not $k$-equivalent for some value $k$, then that means that these functions are not equivalent. In our corpora, there were several cases where $k$-equivalence discovered a bug that the testing missed (Section 4.1).

**$K$-equivalence vs testing.** Like testing, $k$-equivalence can always be applied. Proving $k$-equivalence is usually much stronger information than information obtained by testing.

```
int idx_minA(float a[], int n) {
  int min = 0; int i;
  for (i = 1; i < n; i++)

    if (a[i] <= a[min])
       min = i;
  return min;
}
```

```
int idx_minB(float a[], int n) {
  int i, idx; float min;
  for (i = 1, min = a[0], idx=0;
       i < n; i++)
     if (min >= a[i]) {
       min = a[i];
       idx = i;
     }
  return idx;
}
```

---

```
float idx_minA(float a[], int n) {
  int min = 0; int i = 1;
  min = idx_minA_loop(a, n, i, min);
  return min;
}

float idx_minA_loop(float a[],
                    int n,
                    int i,
                    int min) {
  if (i < n) {
    if (a[i] <= a[min])
       min = i;
    i++;
    min = idx_minA_loop(a, n, i, min);
  }
  return min;
}
```

```
float idx_minB(float a[], int n) {
  int i, idx; float min;
  i = 1, min = a[0], idx=0;
  idx = idx_minB_loop(a, n, i, min, idx);
  return idx;
}

int idx_minB_loop(float a[], int n,
                            int i, int min,
                            int idx) {
  if (i < n) {
    if (min >= a[i]) {
       min = a[i];
       idx = i;
    }
    i++;
    idx = idx_minB_loop(a, n, i,
                              min, idx);
  }
  return idx;
}
```

---

```
float idx_minA(float a[], int n) {
  int min = 0; int i = 1;
  if (i < n) {
    if (a[i] <= a[min])
       min = i;
    i++;
    min = uf(a, n, i, a[min], min);
  }
  return min;
}
```

```
float idx_minB(float a[], int n) {
  int i, idx; float min;
  i = 1, min = a[0], idx=0;
  if (i < n) {
    if (min >= a[i]) {
       min = a[i];
       idx = i;
    }
    i++;
    idx = uf(a, n, i, min, idx);
  }
  return idx;
}
```

**Fig. 5.** Finding the index of the minimum element: implementation, transformation into recursive function and replacement by an uninterpreted function

By testing, it is checked that for one single set of inputs the functions give the same outputs. Here we are not restricted to the possible inputs, but just for the number of loop iterations. For example, if we prove that functions shown in Figure 3 are $k$-equivalent for $k = 5$, that means that for each array of the size 5 or less, these functions calculate the same outputs. This is equivalent to testing the functions with $\Sigma_{i=1}^{5}(2^{(sizeof(int))})^{i}$ different test cases, which, for $sizeof(int) = 32$, approximately equals to $1.46 \times 10^{48}$. Also, complete path coverage is achieved in all cases were loop iterations are restricted to $k$. However, there are situations when checking $k$-equivalence does not provide better information compared to testing, like if there is only one input value and the number of loop iterations together with the resulting values are controlled only by this value.

```
int strcspnA(char s[], char t[]) {
  int i, j;
  for(i=0; s[i]; i++) {
    for(j=0; t[j]; j++)
      if(s[i] == t[j])
         return i;
  }
  return -1;
}
```

```
int strcspnA (char s[], char t[]) {
  int i, j;
  i = 0; int ret1 = RET_UNDEF;
  if (s[i] && ret1 == RET_UNDEF) {
    j = 0; int ret2 = RET_UNDEF;
    if (t[j] && ret2 == RET_UNDEF) {
      if (s[i] == t[j]) ret2 = i;
      if (ret2 == RET_UNDEF) j++;
      ret2 = uf1(ret2, i, j, s, t);
    }
    if (ret2 != RET_UNDEF) ret1 = ret2;
    if (ret1 == RET_UNDEF) i++;
    ret1 = uf2(ret1, i, s, t);
  }
  if (ret1 != RET_UNDEF) return ret1;
  return -1;
}
```

**Fig. 6.** Transforming a function with double `for` loop and a `return` inside

**Partial equivalence vs uninterpreted functions.** For proving partial equivalence by uninterpreted functions, it is necessary to perform the described transformation. If equivalence of two transformed functions is proved, then the original functions are also equivalent. Both the entry point to a loop and the loop-exit condition influence the proof of this equivalence [27]. Therefore, it can be useful to have several model solutions. If equivalence of the two functions cannot be proved, then that does not imply that the original functions are not equivalent: it may happen that their equivalence only cannot be proved this way. There is a number of such examples [27], but, in practice, there are many cases where this technique can be successfully applied (discussed in Section 4.2).

**Uninterpreted functions vs $k$-equivalence.** The obvious advantage of using uninterpreted functions to $k$-equivalence is that partial equivalence is a stronger property. Also, using uninterpreted functions is usually more efficient than loop unrolling with a high value for $k$. However, uninterpreted functions model only changes captured by a single scalar return value. Therefore, they cannot be applied when more than one variable is modified in a loop that is live after the loop, or when the loop modifies values of an array.

## 4. Evaluation and results

To illustrate applicability of regression verification, we analyzed two corpora of problems solved by students and a corpus of classic algorithms that are usually taught at introductory and algorithms courses. Regression verification, in the first context, refers to determining equivalence of solutions provided by the teacher and by the student, and in the second between all pairs of different proposed solutions. All experiments were performed on a computer with an Intel Core i3-4000M on 2.40GHz and with 3.9GB of RAM.

### 4.1. Verifying student solutions

We have conducted an experimental evaluation on two real-world corpora: one from an university introductory programming course for computer science majors (we call this corpus *exam corpus*), and the other from the national programming competitions (we call this corpus *competition corpus*). We chose to use these corpora as automated evaluation is especially important when the number of enrolled students is large, and these are good examples of such situations. In both corpora we analyzed the programs that: (i) successfully compile; (ii) pass all manually designed test cases (12 per problem for the first, and between 15 and 25 for the second corpus);[2] (iii) where a bug-finding tool (we used LAV) does not find any bugs. We chose to use such programs since they are expected to be functionally correct: programs that fail to meet the above requirements are obviously not functionally equivalent to the model solutions, thus there is no need to further analyze them. Also, another important reason for using these corpora is that testing (sometimes enhanced by automated bug finding) is an established approach widely used for automated evaluation. Therefore, by using corpora that successfully pass manually designed test cases and where a bug-finding tool does not find any bugs, we wanted to demonstrate that the proposed approach can add value to preciseness of the automated evaluation.

Both corpora, problem descriptions and the used test cases are publicly available [76]. Statistics showing the number of problems, distribution of number of solutions per problems, lines of code and cyclomatic complexity [49] are given in Table 1.

**Table 1.** Distribution of number of solutions per problem, lines of code (LOC) and cyclomatic complexity (CC) per solution

| **Exam corpus**<br>**12 problems, 224 solutions, 4104 LOC** | | | | | | **Competition corpus**<br>**10 problems, 214 solutions, 4857 LOC** | | | | | | **Classic algorithms**<br>**59 problems, 159 solutions, 2007 LOC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Min | Max | Avg. | Med. | Std. dev. | | Min | Max | Avg. | Med. | Std. dev. | | Min | Max | Avg. | Med. | Std. dev. |
| Solutions per prob. | 3 | 44 | 18.67 | 18 | 12.84 | Solutions per prob. | 4 | 39 | 21.4 | 22.5 | 12.26 | Solutions per prob. | 2 | 7 | 2.69 | 2 | 1.16 |
| LOC per sol. | 5 | 62 | 18.32 | 19 | 11.54 | LOC per sol. | 5 | 83 | 22.7 | 21 | 11.97 | LOC per sol. | 3 | 36 | 12.78 | 11 | 6.95 |
| CC per sol. | 1 | 17 | 6.22 | 6 | 4.01 | CC per sol. | 1 | 50 | 9.01 | 8 | 6.86 | CC per sol. | 2 | 28 | 5.85 | 4 | 4.07 |

**A) Description of the corpora**

**Exam corpus** consists of programs written by students, during programming exams in the programming language C [78]. Originally 1277 solutions to 15 given problems were collected. Programs that do not compile or do not pass testing (1011 solutions), and programs that contain memory violations or other bugs (additional 35 solutions) were eliminated. Three problems (28 solutions) were not suitable for regression verification (in two cases it was more efficient to thoroughly test these solutions, as described in Section 3.2, while in one case the problem requires complex data structures not supported by our verification tool). The filtered corpus consists of 12 problems (203 solutions).

**Competition corpus** consists of programs written by primary school pupils (aged 12 to 16) competing at the national competitions in Serbia (in 2017).[3] This competition is organized in accordance to International Olympiad in Informatics (IOI) guidelines, and scoring and ranking is done solely based on results obtained by automated testing on test-cases, prepared in advance. Among four stages, we considered the second and the third

[2] Test cases were carefully designed for grading purposes and contain different important usage scenarios. For all teacher solutions, 100% of code coverage is achieved by these tests, measured by *gcov* tool [22].

[3] The competition is organised by Mathematical Society of Serbia which is a member of European Mathematical Society. The site of the competition is `https://dms.rs/informatika-osnovne-skole/`

stage. For the first stage, there was no central repository of solutions, while for the forth stage there were just a few solutions that passed testing. We considered 10 different problems with 629 solutions written in C/C++ (that was around 80% of all submitted solutions, other solutions were written in Pascal, Small Basic and C#). After the programs that do not compile or pass testing (411 solutions) and the programs where the bug finding tool detected bugs (4 solutions) were eliminated, the final corpus consists of 214 programs.

**Differences between these two corpora.** In the exam corpus, student solutions are required to be robust and report errors for incorrect inputs, while in the competition corpus it was allowed to assume that the input is always correct (in accordance to the problem specification). Also, after the testing-phase, student solutions were manually inspected and modularity, readability and other aspects were additionally graded. On the other hand, structure and modularity of programs written during competition was quite bad (usually everything was contained in the `main` function), which made them harder to verify.

**Preparing for verification.** To aid verification, the teacher and the student solution should be represented as separate functions that take their input and return output solely through function parameters and the return value. However, in most cases the student solution was implemented within a function that reads the data from the standard input and writes the results on the standard output (especially in the competition corpus). Therefore, our implementation supports an automatic transformation that does the function extraction. The transformed programs are also publicly available [76].

The functions that read the input data and contain assertions that teacher and student solution match (like in the programs from Figure 2) were manually written. In the exam corpus, these functions were simple, including only necessary assertions, while in the competition corpus, these functions contained all necessary additional constraints on input values (imposed by problem descriptions).

## B) Results

The results are summarized in Table 2. The problems from both corpora can be divided into two types. The first type of problems (denoted as A) does not require using loops in their solutions or only requires the use of bounded loops. For this type of programs, the used approach is exact, i.e. it does not make neither false positives nor false negatives. The second type of problems (denoted as B) requires the use of loops in their solutions. These do not have upper bounds or have high upper bounds that cannot be completely unrolled due to time and memory limits.

**Table 2.** The results of regression verification applied on exam and competition corpus

| **Type of problem** | Corpus | Num. of problems | Num. of solutions (total / per problem) | Num. of functions | Equivalent by RV/manually | Non-equivalent by RV/manually |
|---|---|---|---|---|---|---|
| **(A) Problem requires using bounded loops or no loops** | Exam corpus | 6 | 136 / [3,7,18,31,33,44] | 136 | 129 / 129 | 7 / 7 |
| | Competition corpus | 5 | 88 / [4,4,12,29,39] | 88 | 77 / 80 | 8 / 8 |
| | Total | 11 | 224 | 224 | 206 / 209 | 15 / 15 |
| **(B) Problem requires using high upper bounds or no bounds** | Exam corpus | | | | | |
| | – UF + k-equivalence | 2 | 41 / [20,21] | 62 | 38 + 16 / 54 | 8 / 8 |
| | – only $k$-equivalence | 4 | 26 / [4,5,7,10] | 26 | 20 / 20 | 6 / 6 |
| | Competition corpus | | | | | |
| | – only $k$-equivalence | 5 | 126 / [16,20,25,32,33] | 126 | 106 / 111 | 15 / 15 |
| | Total | 11 | 193 | 214 | 180 / 185 | 29 / 29 |

**Exam corpus.** The exam corpus contains 6 problems with 136 solutions of type A. LAV successfully shows equivalence for 129 (correct) solutions and finds 7 solutions that are not functionally equivalent to the model solutions. The exam corpus contains 6 different problems with 67 solutions of type B.

**Partial equivalence using uninterpreted functions:** For two problems with 41 solutions and 62 pairs of checked functions, it was possible to check equivalence by uninterpreted functions and in 38 cases the equivalence is proved. For remaining 24 functions, equivalence cannot be proved by uninterpreted functions. In these cases, we checked for $k$-equivalence for $k = 5$, and 16 functions are proved $k$-equivalent, while 8 functions (in five different solutions) are proved to be non $k$-equivalent.

**$k$-equivalence by loop unrolling:** The remaining 4 problems with 26 solutions cannot be modelled by uninterpreted functions. We tried proving $k$-equivalence, and decided to use $k = 5$ as we find it a reasonable compromise between scalability and reliability of obtained results (scalability is discussed in Section 4.2). In many cases, $k = 5$ corresponds to equivalence checking of an algorithm that is applied on all arrays of the maximum size 5 and obtains full path coverage in such cases. Knowing the nature of these problems, we expected that there should not be a significant difference between, for example, an array of the size 5 and an array of a bigger size, and our experimental results confirmed this assumption, showing that even smaller values for $k$ could have been used without compromising preciseness of the results. LAV successfully proved $k$-equivalence of 20 solutions which are indeed functionally equivalent (based on manual check). For the remaining 6 solutions, LAV proved non $k$-equivalence. All non $k$-equivalent solutions could have been found already with $k = 2$.

The time for program transformation was negligible. The average time for verification per solution was 0.7s, while the median value was 0.05s. The LA theory was used whenever it was possible, as it provides faster verification. Otherwise, BVA was used. For example, LAV generates formulas and verifies functional equivalence of functions `maxA` and `maxB` (from Figure 1) with respect to the theory of LA and the Z3 SMT solver in 0.02 seconds. If a solver for the BVA theory is used, then the time necessary for proving this equivalence is 0.16 seconds with the SMT solver Boolector, and 0.84 seconds with the SMT solver Z3. In an analogous way, LAV can also prove that the functions `maxA` and `maxD` are not functionally equivalent. The time needed for this is 0.02 seconds in the context of linear arithmetic, and 0.09 in the context of the theory of bit-vectors. LAV generates a counterexample ($a = 29$, $b = 29$, $c = 30$), i.e. the values of the variables for which this equivalence does not hold. This counterexample can be useful for understanding the bug in the function `maxD`. Proving partial equivalence by using uninterpreted functions was usually faster than showing $k$-equivalence. Proving partial equivalence of functions from Figure 5 takes 0.028 seconds in the context of LA, and proving 5-equivalence of functions from Figure 3 takes 0.068 seconds.

**Competition corpus.** Results for the competition corpus are very similar. Because of the poor modularity and almost total absence of user defined functions in the code, on this corpus we could not apply regression verification with uninterpreted functions. In order to check the claim that similar verification tools can also be used for this purpose, in addition to LAV, we ran the CBMC tool. As expected, we got the same results. There were 8 solutions in this corpus that contain library function calls that are not precisely modelled by both tools or that contain advanced C++ concepts (from standard template library) that are unsupported by both tools. Therefore, these solutions were not considered. The results are summarized in Table 2. The average time per solution for CBMC was 1.4s, while for LAV it was 7s. The median value for CBMC was 0.8s and for LAV was 0.7s. We also applied random testing to all programs in this corpus (we generated fresh 25 random tests for each task), but random testing detected bugs in only 2 solutions.

## C) Discussion

We performed a quantitative analysis of the obtained results to assess in what extent the proposed approach can add to quality of automated evaluation. We also performed a detailed qualitative analysis of the obtained results to detect in what situations it can be expected to get the most from the proposed approach.

**Quantitative analysis of results.** The percentage of non-equivalent solutions is approximately the same in both corpora: it is around 10% of all solutions that have been analyzed (and graded as being functionally correct). It is relatively low since programs were thoroughly tested and bug finding tool was applied.[4] However, it is definitely not negligible and reveals that in spite of very thorough testing and bug-finding, around 10% of programs still contain bugs that go undetected. This illustrates the limited power of these approaches and shows that the proposed approach can add to the quality and precision of automated evaluation.

**Qualitative analysis of results.** We manually analyzed all programs that were shown to be non-equivalent to the model solution, in order to detect what kind of bugs are found by regression verification. In the following text we summarize such examples.

**Completely different logic valid in most cases.** There were some solutions that are very different from the expected solution and which work for most input values. For example, one model solution required calculating $\lceil \frac{x}{3} \rceil \cdot \lceil \frac{y}{3} \rceil$ while a student submitted a solution with 47 lines of code that introduced 9 auxiliary variables, with 10 different branches. Another example (found in several solutions) is comparison of two dates by converting them to integers, using the formula $d + m \cdot 30 + y \cdot 365$.

**Missing branches.** In several cases, students introduced unnecessary branching which left the input uncovered. One such example is illustrated on Figure 7 where the branch for $K < 5$ and $R = 5$ is missing. It is hard to cover such situations by test-cases, since the branching is not the part of the problem semantics, but is artificially introduced by the student. In some cases, branching ends with an `else` branch and all missing branches will execute its code. For example, the function `maxD` in Figure 1 contains such an error, i.e. the branch for $o = p$ and $q < o$ is missing.

**Specific input series.** Some errors were due to wrong behavior of programs when the input series of numbers were specific in some sense, for example, series containing just a single element or series containing elements in some specific order. Although such errors could be caught by careful testing, it is hard to predict all such special inputs in advance. Applying regression verification removes the burden from the test designer, making grading much more reliable.

[4] Automated bug-finding in this context searched for bugs such as buffer overflows, division by zero or null pointer dereferencing. For the exam corpus, a detailed testing and automated bug finding is described in [78]. The same approach is applied in the case of the competition corpus.

```
if(K<5 && R<5)
  i=(K/2)*(R/2);
else if(K<5 && R>5)
  i=(K/2)*(((R-1)/3)+1);
else if(K>=5 && R<5)
  i=(((K-1)/3)+1)*(R/2);
else if(K>=5 && R>=5)
  i=(((K-1)/3)+1)*(((R-1)/3)+1);
```

```
int y;
while (yr > 0) {
  y = y + 1;
  yr = yr - 3;
}
```

**Fig. 7.** A missing branch (left-hand side) and uninitialized variable (right-hand side)

**Uninitialized variables.** In several cases uninitialized local variables were used, like the solution from the competition corpus which contained the code shown on Figure 7. Although the initial value of local variables at run-time cannot be predicted, in many cases it is zero (and it is usually the correct initial value) and the tests pass.

**Potential errors in variable range.** In some cases, solutions used constructs that could potentially introduce integer overflows. In the concrete tasks, limits were such that those solutions were detected to be safe. However, if the assumptions for the limits are removed, the verification detects non-equivalence and this could be used to signal potential errors to novice programmers. For example, for sorting three integer variables some solutions found the minimum, the maximum, and calculated the middle one as the sum minus the minimum and the maximum. A similar situation was comparing dates by converting them to integers using the formula $1000 \cdot y + 50 \cdot m + d$ or when maximum/minimum is initialized to arbitrary values (in our corpus, we have seen minimum being initialized to 1000000000, 454545454, 1000, 9990000, 12345, and 99999999). Regression verification detects these solutions as non-equivalent when no additional assumptions are given, and that can be used to warn programmers about bad programming style and potential errors.

Most of the errors were found in programs containing rich branching structure. Programs that do not contain branching (whether or not they contain loops) and that pass testing, usually do not contain errors or contain only errors detected by bug finding (buffer-overflows, division by zero etc.). On the other hand, programs where control flow can follow various paths are much harder to verify only by testing and applying regression verification is most beneficial in such situations. A good indicator where regression verification can be beneficial is the presence of solutions that fail just in a few test cases. That indicates that some solutions failed only in some branches, and it is reasonable to expect that the solutions that passed all the tests could also contain errors (in some other branches that were not covered by test cases).

## 4.2. Verifying classic algorithms

To illustrate the type of problems that can be assessed by regression verification, we have applied regression verification to same standard algorithms that are usually covered in introductory and algorithms courses [14,63]. Detailed problem descriptions and corresponding source codes are available on web page [76] together with the two other corpora. Statistics summarizing the number of problems and solutions, their length, and cyclomatic complexity are given in Table 1.

**A) Description of the corpus**

We applied our approach on some loop free algorithms (most of them based on different forms of branching, like branching based on discrete values, intervals, lexicographic comparison, or hierarchical nested branching) and on some programs with loops, using the technique of uninterpreted functions (algorithms that calculate statistics, perform linear search and filter series, map all series elements by applying a given transformation, and various combinations of such algorithms). $K$-equivalence can be successfully applied on a wide set of problems. We examined 15 problems with 59 fundamentally different solutions that yielded 100 pairs that were checked for $k$-equivalence. For example, we have considered the problem of finding a sub-array of contiguous elements with the maximal sum and have shown $k$-equivalence between the brute-force solution, its optimized variant based on two-pointer technique, the solution based on Kadane's (dynamic programming) algorithm, the solution based on maximizing the difference between the array partial sums, and a solution based on the recursively implemented divide-and-conquer approach.

**B) Results**

Times needed for showing partial equivalence are summarized in Table 3, showing that if this approach is applicable, then the verification is usually very fast. Distribution of times needed for showing $k$-equivalence for different values of $k$ in more advanced algorithms using CBMC are summarized in Table 4. Table shows that required verification times quickly grow. Verifying recursive solutions is the most time consuming: all 10 cases where the timeout of 60 seconds was violated for $k = 5$ involved at least one recursive solution. We also applied LAV on 32 non-recursive solutions (since it does not support recursive function unrolling) and the results were very similar.

**Table 3.** Partial equivalence of classic algorithms: number of problems, solutions and checked pairs of solutions; minimum, maximum and median time in seconds

| Group name | Num. of problems | Num. of solutions | Num. of pairs | Min. | Max. | Median |
|---|---|---|---|---|---|---|
| Loop free programs | 14 | 36 | 33 | 0.01 | 0.27 | 0.01 |
| Loops – unininterpreted functions | 30 | 64 | 35 | 0.01 | 2.68 | 0.01 |

**C) Discussion**

Proving functional equivalence of two equivalent solutions is more time demanding than finding a difference between two non-equivalent solutions. The reason is that in proving functional equivalence all possible paths through two different solutions must be analyzed, while for finding a difference all possible paths through solutions are analyzed only in the worst case. Since our analysis applied on classic algorithms corpus included only functionally equivalent solutions, this suggests that the same or less amount of time is needed in case of considering non-equivalent solutions of the proposed problems, which is an important use-case in context of students solutions.

**Table 4.** $k$-equivalence of classic algorithms for different values of $k$, where each pair of solutions is checked for equivalence — first row: a number of proved pairs (time out set to 60 seconds) vs. number of all pairs; second row: min.–max.(median) times (in seconds)

| **Problem name (Num. of solutions)** | **proved/all pairs min - max (median)** | | | |
|---|---|---|---|---|
| | $k = 3$ | $k = 4$ | $k = 5$ | $k = 6$ |
| 1. Search (5) | 10/10 | 10/10 | 9/10 | 7/10 |
| | 0.17 - 9.6 (0.42) | 0.23 - 13.79 (2.94) | 0.4 - 43.31 (1.86) | 0.59 - 21.49 (4.68) |
| 2. Sort (7) | 21/21 | 21/21 | 13/21 | 6/21 |
| | 0.18 - 3.35 (1.44) | 0.44 - 22.93 (6.29) | 1.56 - 58.07 (38.86) | 11.16 - 21.89 (17.57) |
| 3. $K$-th element (3) | 3/3 | 3/3 | 2/3 | 0/3 |
| | 0.35 - 3.14 (1.03) | 1.45 - 55.04 (4.96) | 12.34 - 25.87 (19.105) | - |
| 4. Majority (3) | 3/3 | 3/3 | 3/3 | 3/3 |
| | 0.12 - 0.31 (0.3) | 0.15 - 0.63 (0.63) | 0.18 - 1.17 (1.12) | 0.26 - 3.82 (2.95) |
| 5. Fibonacci (4) | 6/6 | 6/6 | 6/6 | 6/6 |
| | 0.09 - 0.36 (0.22) | 0.1 - 0.72 (0.4) | 0.1 - 1.58 (0.78) | 0.1 - 3.82 (1.79) |
| 6. Longest increasing subsequence (4) | 6/6 | 6/6 | 6/6 | 6/6 |
| | 0.14 - 0.22 (0.18) | 0.16 - 0.46 (0.34) | 0.3 - 1.31 (0.69) | 0.53 - 6.32 (2.48) |
| 7. Min. coins (6) | 15/15 | 15/15 | 15/15 | 15/15 |
| | 0.28 - 0.29 (0.28) | 0.41 - 0.43 (0.42) | 0.9 - 0.93 (0.92) | 5.84 - 5.93 (5.86) |
| 8. Stock span (2) | 1/1 | 1/1 | 1/1 | 1/1 |
| | 0.15 - 0.15 (0.15) | 0.34 - 0.34 (0.34) | 1.14 - 1.14 (1.14) | 5.63 - 5.63 (5.63) |
| 9. Max. segment - sum (5) | 10/10 | 10/10 | 10/10 | 5/10 |
| | 0.12 - 0.56 (0.18) | 0.3 - 2.35 (0.56) | 0.83 - 24.85 (5.48) | 2.79 - 29.56 (20.2) |
| 10. Num. of segments (3) | 3/3 | 3/3 | 3/3 | 0/3 |
| | 0.21 - 0.25 (0.22) | 1.36 - 2.83 (1.66) | 13.75 - 56.06 (22.75) | - |
| 11. Num. of pairs - sum (3) | 3/3 | 3/3 | 3/3 | 3/3 |
| | 0.15 - 0.33 (0.24) | 0.25 - 0.59 (0.49) | 0.66 - 1.69 (1.23) | 1.44 - 4.29 (3.78) |
| 12. Num. of pairs - diff (3) | 3/3 | 3/3 | 3/3 | 0/3 |
| | 0.28 - 0.70 (0.43) | 1.78 - 6.45 (3.05) | 14.87 - 45.67 (39.9) | - |
| 13. MaxPairing (5) | 10/10 | 10/10 | 10/10 | 6/10 |
| | 0.18 - 0.31 (0.26) | 0.32 - 1.01 (0.61) | 0.66 - 7.99 (1.52) | 1.64 - 4.91 (3.19) |
| 14. Longest palindromic substring (3) | 3/3 | 3/3 | 3/3 | 3/3 |
| | 0.17 - 0.55 (0.41) | 0.26 - 1.96 (1.52) | 0.4 - 6.84 (5.10) | 0.6 - 19.89 (15.43) |
| 15. Prefix - suffix (3) | 3/3 | 3/3 | 3/3 | 3/3 |
| | 0.13 - 0.14 (0.13) | 0.16 - 0.20 (0.18) | 0.28 - 0.34 (0.28) | 0.41 - 0.55 (0.43) |
| Total (59) | 100/100 | 100/100 | 90/100 | 64/100 |
| | 0.09 - 9.6 (0.28) | 0.1 - 55.04 (0.53) | 0.1 - 58.07 (1.46) | 0.1 - 29.56 (5.27) |

## 4.3. Threats to validity

Our experimental results give good promises for real-world applications in education, but their generalization to other situations have to be discussed.

Languages C/C++ are present at introductory programming and algorithms courses at many leading universities, but are not the most popular choices [30]. Although the tools we use are tailored for C/C++, the proposed approach can be adapted for other languages.

We do not consider equivalence of bigger sized projects, but this issue could be addressed by showing equivalence of their smaller parts (like modules or functions). The problem of detecting and aligning parts of code is discussed in regression verification [6,20,27], but that is an orthogonal problem to the one we studied. Also, different solutions of students projects might be completely different, as such projects are usually not given by strict specifications, while a certain level of creativity is even expected.

Finally, there is a question of the number of errors that are undetected by testing and bug finding and are found by regression verification. In our experiments, percentage of such programs was around 10% in both corpora. Obviously, the percentage depends on how test-cases are designed, but since our corpora are taken from real-world exams and competitions, we expect that these are representative or at least very illustrative examples. By the detailed analysis of the detected errors, we have shown that there are situations where test cases are very hard or almost impossible to predict in advance. Therefore, applying the proposed approach releases the burden of creating such tests.

## 5. Relationship to other approaches

In Section 2 we identified all important aspects of the related work, and here we discuss them in the context of the proposed approach.

Regression verification techniques customized for specific domains [4,8,34,59,62,74,80] are usually only semi-automated and require additional information from an expert (like inductive invariants). However, it is not likely that students, or even teachers, would be able to provide such expertise. Therefore, these customized techniques do not seem applicable for the evaluation of student solutions. In regression verification of large systems [6,20,27], program behavior is usually checked only for $k$-equivalence [6], while we consider both $k$-equivalence and partial equivalence in order to get more accurate results. In regression verification of student programs, the code is usually short and, in contrast to regression verification of large systems, it is not difficult to determine which functions should be checked for equivalence. On the other hand, checking equivalence of matched functions in our context can be very challenging since these functions did not evolve from the same code and therefore may have a high level of diversity.

The rewriting-based approach [40] for verification of student programs requires formal specifications to be available. An evaluation of this approach was performed on a corpus consisting of 41 programs (all chosen to be functionally correct) – solutions of 5 different simple problems. The student solutions were transformed manually and the system successfully verified 27 programs out of 41. The downside of this approach is that it can be difficult task for a teacher to create formal specifications. In our approach, code transformations are performed automatically. The corpora used in our evaluation are bigger and wider, hence lead to more conclusive results.

Concerning evaluation based on automated testing [11,18,23,31,33,50,51,65,72] and automated bug-finding [36,37,71,78], our approach is complementary and it gives an additional confidence on functional correctness of the program. We think that the best way to use it is to apply it on programs where cheaper techniques such as testing and automated bug-finding did not discover any bugs. Still, it can also be used complementary to the grading techniques which do not asses functional correctness (e.g. grading techniques based solely on machine learning [66]).

Concerning other important aspects that should be taken into account within a detailed evaluation of programs (see Section 2), our approach, in some cases, can be used for these purposes, too. For example, given that partial equivalence between the student solution and some particular predefined solution has been proven, it confirms that these programs share main characteristics of the used algorithms. Also, our approach can be used for generating failing test-cases (as illustrated in Section 4.1) as a useful feedback for students, as it is done in approaches based on automated testing. Although the main purpose of our approach is to provide high-quality and precise grading at final tests or at competitions, it can be also used as a support for clustering of programs, potentially leading to a finer feedback for specific clusters (e.g. in synergy with approaches for classification and clustering of programs based on static or dynamic analysis [24,55]).

## 6. Conclusions and further work

There is a significant theoretical and practical progress that has been made recently in the field of regression verification. We have shown that regression verification can be successfully used in automated evaluation of programs at introductory programming courses, more advanced algorithms courses, and programming competitions, and that it can be very useful for both teachers and students (without affecting the teaching methodology itself). Showing equivalence with the teacher solution gives a much higher confidence in the correctness of student program. We find that regression verification should be used as an extension of classical evaluation process. Moreover, for loop-free programs, regression verification may even replace testing, as results obtained for such programs are definite.

The implementation of the proposed approach transforms C/C++ programs and prepares them for regression verification. We have shown that our system LAV can be successfully used in this context. The presented results have shown that our tools were able to automatically show some kind of equivalence for almost all student programs that are equivalent to model solutions (except a few of those that used unmodeled library functions). For loop-free programs, the total equivalence was shown, while for programs with loops, in some cases a very strong relation of partial equivalence was fully automatically shown, and in all other cases $k$-equivalence was shown. In 10% of the programs from the two considered real world corpora regression verification found bugs that were not previously discovered by testing and automated bug finding. This shows that even when test-cases are carefully manually crafted and achieve complete code coverage of the model solutions, testers fail to predict all possible situations where the student solution might go wrong and bugs can go undetected. Therefore, regression verification can add to quality and precision of automated evaluation, offering an important complement to testing and, in some cases, even a good alternative for a hard and time-consuming job of manually designing test-cases. The precision of the obtained results shows that these techniques could be integrated into a grading system that would be more reliable than those based on testing. We have analyzed functionally non-equivalent solutions and identified that we can get the most from regression verification in situations where solutions have rich branching structure (either imposed by problem definition or artificially introduced by students). We have described some of the most common sources of bugs and we have analyzed and described the types of problems that can be efficiently assessed by regression verification. Uninterpreted functions can be successfully applied to showing partial equivalence in some cases, but not always.

We are planning to introduce other, more powerful, regression verification techniques [20] and also to develop new ones, such that regression verification can be successfully applied to a wider set of problems. We are also planning to improve our tool for automated transformation of programs and to integrate fully automated regression verification into our set of techniques for automated evaluation of students programs.

**Acknowledgements.** This work was partially supported by the Serbian Ministry of Science grant 174021 and by the COST action CA15123.

## References

1. Afzal, W., Torkar, R., Feldt, R.: A systematic review of search-based testing for non-functional system properties. Information and Software Technology 51(6), 957–976 (2009)
2. Ala-Mutka, K.M.: A Survey of Automated Assessment Approaches for Programming Assignments. Computer Science Education 15, 83–102 (2005)
3. Allen, I.E., Seaman, J.: Learning on demand: Online education in the United Statesf. Tech. rep., The Sloan Consortium (2010)
4. Amtoft, T., Bandhakavi, S., Banerjee, A.: A logic for information flow in object-oriented programs. In: POPL. pp. 91–102. ACM (2006)
5. Babic, D., Hu, A.J.: Calysto: Scalable and Precise Extended Static Checking. In: ICSE. pp. 211–220. ACM (2008)
6. Backes, J., Person, S., Rungta, N., Tkachuk, O.: Regression verification using impact summaries. In: SPIN'13. pp. 99–116 (2013)
7. Barrett, C., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability Modulo Theories. In: Handbook of Satisfiability. vol. 185, pp. 825–885. IOS Press (2009)
8. Barthe, G., Grégoire, B., Kunz, C., Lakhnech, Y., Béguelin, S.Z.: Automation in computer-aided cryptography: Proofs, attacks and designs. In: CPP. pp. 7–8 (2012)
9. Beyer, D.: Automatic Verification of C and Java Programs: SV-COMP 2019. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 133–155. Springer (2019)
10. Cadar, C., Dunbar, D., Engler, D.: KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs. In: OSDI. pp. 209–224. USENIX (2008)
11. Cheang, B., Kurnia, A., Lim, A., Oon, W.C.: On Automated Grading of Programming Assignments in an Academic Institution. Computers and Education 41(2), 121–131 (2003)
12. Clarke, E., Kroening, D., Lerda, F.: A Tool for Checking ANSI-C Programs. In: TACAS. pp. 168–176. Springer (2004)
13. Clarke, E.M.: 25 Years of Model Checking — The Birth of Model Checking. Springer (2008)
14. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT (2009)
15. Cousot, P., Cousot, R.: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: POPL. ACM (1977)
16. Dijkstra, E.: Notes on structured programming. EUT report. WSK, Dept. of Mathematics and Computing Science, Technische Hogeschool Eindhoven, 2nd ed. edn. (1970)
17. Douce, C., Livingstone, D., Orwell, J.: Automatic Test-based Assessment of Programming: A Review. Journal on Educational Resources in Computing 5(3) (2005)
18. Ellsworth, C.C., Fenwick, Jr., J.B., Kurtz, B.L.: The Quiver System. In: SIGCSE. ACM (2004)
19. Ertmer, P.A., Richardson, J.C., Belland, B., Camin, D., Connolly, P., Coulthard, G., Lei, K., Mong, C.: Using Peer Feedback to Enhance the Quality of Student Online Postings: An Exploratory Study. Journal of Computer-Mediated Communication 12(2), 412–433 (2007)
20. Felsing, D., Grebing, S., Klebanov, V., Rümmer, P., Ulbrich, M.: Automating regression verification. In: ASE. pp. 349–360. ACM (2014)
21. Flanagan, C., Leino, K.R.M., Lillibridge, M., Nelson, G., Saxe, J.B., Stata, R.: Extended Static Checking for Java. SIGPLAN Not. 37(5), 234–245 (2002)
22. Foundation, F.S.: Using the GNU Compiler Collection: gcov — a Test Coverage Program (1988-2019), `https://gcc.gnu.org/onlinedocs/gcc/Gcov.html`
23. GeeksforGeeks: A CS portal for geeks (2019), `https://www.geeksforgeeks.org/`
24. Glassman, E.L., Scott, J., Singh, R., Guo, P.J., Miller, R.C.: Overcode: Visualizing variation in student solutions to programming problems at scale. Comput.-Hum. Interact. 22(2) (2015)
25. Godlin, B.: Regression verification: Theoretical and implementation aspects (2008), masters Thesis, Technion, Israel Institute of Technology
26. Godlin, B., Strichman, O.: Regression verification. In: Proceedings of the 46th Annual Design Automation Conference. pp. 466–471. DAC '09, ACM, New York, NY, USA (2009)

27. Godlin, B., Strichman, O.: Regression verification: proving the equivalence of similar programs. Softw. Test., Verif. Reliab. 23(3), 241–258 (2013)
28. Grivokostopoulou, F., Perikos, I., Hatzilygeroudis, I.: An educational system for learning search algorithms and automatically assessing student performance. International Journal of Artificial Intelligence in Education (1), 207–240 (2017)
29. Gulwani, S., Radiček, I., Zuleger, F.: Feedback generation for performance problems in introductory programming assignments. In: FSE. pp. 41–51. ACM (2014)
30. Guo, P.: Python is the Most Popular Introductory Teaching Language at Top U.S. Uni. (2014)
31. HackerRank: For devs, companies and schools. (2019), `https://www.hackerrank.com`
32. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM 12(10) (1969)
33. Huang, L., Holcombe, M.: Empirical investigation towards the effectiveness of Test First programming. Information and Software Technology 51(1), 182–194 (2009)
34. Huang, S.Y., Cheng, K.T.: Formal equivalence checking and design debugging. Kluwer (1998)
35. Ihantola, P., Ahoniemi, T., Karavirta, V., Seppälä, O.: Review of Recent Systems for Automatic Assessment of Programming Assignments. In: Koli Calling. pp. 86–93. ACM (2010)
36. Ihantola, P.: Creating and Visualizing Test Data From Programming Exercises. Informatics in education 6(1), 81–102 (2007)
37. Juniwal, G., Donzé, A., Jensen, J.C., Seshia, S.A.: Cpsgrader: Synthesizing temporal logic testers for auto-grading an embedded systems laboratory. In: EMSOFT (2014)
38. Kefalas, P., Stamatopoulou, I.: Using screencasts to enhance coding skills: The case of logic programming. Comput. Sci. Inf. Syst. 15(3), 775–798 (2018)
39. King, J.C.: Symbolic Execution and Program Testing. Commun. ACM 19(7) (1976)
40. Kop, C., Nishida, N.: Automatic constrained rewriting induction towards verifying procedural programs. In: Programming Languages and Systems, LNCS, vol. 8858. Springer (2014)
41. Krusche, S., Seitz, A.: Artemis: An automatic assessment management system for interactive learning. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education. pp. 284–289. SIGCSE '18, ACM (2018)
42. Kulkarni, C., Wei, K.P., Le, H., Chia, D., Papadopoulos, K., Cheng, J., Koller, D., Klemmer, S.R.: Peer and self assessment in massive online classes. Comput.-Hum. Interact. 20(6) (2013)
43. Laski, J., Stanley, W.: Software Verification and Analysis: An Integrated, Hands-On Approach. Springer, 1 edn. (2009)
44. Lattner, C.: The LLVM Compiler Infrastructure (2012), `http://llvm.org/`
45. Li, S., Xiao, X., Bassett, B., Xie, T., Tillmann, N.: Measuring code behavioral similarity for programming and software engineering education. In: ICSE (2016)
46. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory programming: A systematic literature review. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. pp. 55–106. ITiCSE 2018, ACM (2018)
47. Mandal, A.K., Mandal, C.A., Reade, C.: A System for Automatic Evaluation of C Programs: Features and Interfaces. IJ. of Web-Based Learning and Teaching Technologies 2(4) (2007)
48. Marin, V.J., Pereira, T., Sridharan, S., Rivero, C.R.: Automated personalized feedback in introductory java programming moocs. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE). pp. 1259–1270 (2017)
49. McCabe, T.: Structured testing. Tutorial Texts Series, IEEE (1983)
50. Miguel A. Revilla: Uva online judge. (1995-2019), `https://uva.onlinejudge.org/`
51. Mike Mirzayanov: Codeforces (2010 - 2019), `http://codeforces.com/`
52. Moghadam, J.B., Choudhury, R.R., Yin, H., Fox, A.: Autostyle: Toward coding style feedback at scale. In: ACM Conference on Learning @ Scale. pp. 261–266. L@S, ACM (2015)
53. Myers, G.J., Sandler, C., Badgett, T.: The Art of Software Testing. Wiley Publishing, 3rd edn. (2011)
54. Naudé, K.A., Greyling, J.H., Vogts, D.: Marking Student Programs Using Graph Similarity. Computers and Education 54(2), 545–561 (2010)

55. Nguyen, A., Piech, C., Huang, J., Guibas, L.: Codewebs: Scalable homework search for massive open online programming courses. In: WWW. pp. 491–502. ACM (2014)
56. Pappano, L.: The year of the MOOC (2012)
57. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., Paterson, J.: A Survey of Literature on the Teaching of Intr. Prog. In: WG reports on ITiCSE. ACM (2007)
58. Pieterse, V.: Automated assessment of programming assignments. In: CSERC (2013)
59. Post, H., Sinz, C.: Proving functional equivalence of two AES implementations using bounded model checking. In: ICST. pp. 31–40 (2009)
60. Rivers, K., Koedinger, K.: Automating hint generation with solution space path construction. In: 12th Intl. Conf. on Intelligent Tutoring Systems (2014)
61. Rizzardini, R.H., Garca-Pealvo, F.J., Kloos, C.D.: Massive open online courses: Combining methodologies and architecture for a success learning. JUCS 21(5), 636–637 (2015)
62. Scheben, C., Schmitt, P.H.: Efficient self-composition for weakest precondition calculi. In: FM 2014: Formal Methods, LNCS, vol. 8442, pp. 579–594. Springer (2014)
63. Sedgewick, R., Wayne, K.: Algorithms. Addison-Wesley Professional, 4th edn. (2011)
64. Singh, R., Gulwani, S., Solar-Lezama, A.: Automated feedback generation for introductory programming assignments. In: PLDI. pp. 15–26. ACM (2013)
65. Sphere Research Labs: Sphere Online Judge (SPOJ) (2019), http://www.spoj.com/
66. Srikant, S., Aggarwal, V.: A system to grade computer programming skills using machine learning. In: ACM KDD. pp. 1887–1896. ACM (2014)
67. Strichman, O., Godlin, B.: Regression verification - a practical way to verify programs. In: VSTTE. LNCS, vol. 4171, pp. 496–501. Springer (2005)
68. Strichman, O.: Special issue: program equivalence. Formal Methods in System Design 52(3), 227–228 (Jun 2018), https://doi.org/10.1007/s10703-018-0318-y
69. Stuikys, V., Burbaite, R., Damasevicius, R.: Teaching of computer science topics using meta-programming-based glos and LEGO robots. Informatics in Education 12(1), 125–142 (2013)
70. Taherkhani, A., Korhonen, A., Malmi, L.: Automatic recognition of students' sorting algorithm implementations in a data structures and algorithms course. In: Koli Calling. ACM (2012)
71. Tillmann, N., Halleux, J.: Pex – White Box Test Generation for .NET . In: TAP. LNCS, vol. 4966, pp. 134–153. Springer (2008)
72. Topcoder: Topcoder (2001–2019), https://www.topcoder.com/
73. Valiente, J.A.R., Merino, P.J.M., Díaz, H.J.P., Ruiz, J.S., Kloos, C.D.: Evaluation of a learning analytics application for open edX platform. Comput. Sci. Inf. Syst. 14(1), 51–73 (2017)
74. Verdoolaege, S., Palkovic, M., Bruynooghe, M., Janssens, G., Catthoor, F.: Experience with widening based equiv. checking in realistic multimedia systems. J. Elec. Testing 26(2) (2010)
75. Visser, W., Havelund, K., Brat, G., Park, S., Lerda, F.: Model checking programs. Automated Software Eng. 10(2), 203–232 (2003)
76. Vujošević Janičić, M.: LAV (2009 -), http://argo.matf.bg.ac.rs/?content=lav
77. Vujošević Janičić, M., Kuncak, V.: Development and Evaluation of LAV: An SMT-Based Error Finding Platform. In: VSTTE. pp. 98–113. LNCS, Springer (2012)
78. Vujošević Janičić, M., Nikolić, M., Tošić, D., Kuncak, V.: Software verification and graph similarity for automated evaluation of students assignments. Inf. and Soft. Tech. 55(6) (2013)
79. Vujošević Janičić, M., Tošić, D.: The Role of Programming Paradigms in the First Programming Courses. The Teaching of Mathematics XI(2), 63–83 (2008)
80. Welsch, Y., Poetzsch-Heffter, A.: A fully abstract trace-based semantics for reasoning about backward compatibility of class libraries. Sci. Comput. Program. 92, 129–161 (2014)

**Milena Vujošević Janičić** is currently an assistant professor at the Department of Computer Science, Faculty of Mathematics, University of Belgrade. Her main research interests are in automated bug finding, model checking and application of software verification techniques in different fields. She is a member of the Automated Reasoning GrOup (ARGO) at the University of Belgrade.

**Filip Marić** is currently an associate professor at the Department of Computer Science, Faculty of Mathematics, University of Belgrade. His main research interests are in interactive theorem proving and its applications in formalization of mathematics and software verification. He is also interested in SAT and SMT solving and their applications and in teaching programming at introductory level. He is a member of the Automated Reasoning GrOup (ARGO) at the University of Belgrade.

*Received: December 20, 2018; Accepted: July 10, 2019.*

Computer Science and Information Systems 17(1):229–252 https://doi.org/10.2298/CSIS180717038L

# Visualization of path patterns in semantic graphs [⋆]

José Paulo Leal

CRACS & INESC-Tec LA, Faculty of Sciences, University of Porto
Porto, Portugal
zp@dcc.fc.up.pt

**Abstract.** Graphs with a large number of nodes and edges are difficult to visualize. Semantic graphs add to the challenge since their nodes and edges have types and this information must be mirrored in the visualization. A common approach to cope with this difficulty is to omit certain nodes and edges, displaying sub-graphs of smaller size. However, other transformations can be used to summarize semantic graphs and this research explores a particular one, both to reduce the graph's size and to focus on its path patterns.
A-graphs are a novel kind of graph designed to highlight path patterns using this kind of summarization. They are composed of a-nodes connected by a-edges, and these reflect respectively edges and nodes of the semantic graph.
A-graphs trade the visualization of nodes and edges by the visualization of graph path patterns involving typed edges. Thus, they are targeted to users that require a deep understanding of the semantic graph it represents, in particular of its path patterns, rather than to users wanting to browse the semantic graph's content. A-graphs help programmers querying the semantic graph or designers of semantic measures interested in using it as a semantic proxy. Hence, a-graphs are not expected to compete with other forms of semantic graph visualization but rather to be used as a complementary tool.
This paper provides a precise definition both of a-graphs and of the mapping of semantic graphs into a-graphs. Their visualization is obtained with a-graphs diagrams. A web application to visualize and interact with these diagrams was implemented to validate the proposed approach.
Diagrams of well-known semantic graphs are presented to illustrate the use of a-graphs for discovering path patterns in different settings, such as the visualization of massive semantic graphs, the codification of SPARQL or the definition of semantic measures.
The validation with large semantic graphs is the basis for a discussion on the insights provided by a-graphs on large semantic graphs: the difference between a-graphs and ontologies, path pattern visualization using a-graphs and the challenges posed by large semantic graphs.

**Keywords:** Semantic Graph Visualization, Semantic Graph Summarization, Linked Data Visualization, Path Pattern Discovery, Semantic Graph Transformation

## 1. Introduction

Graphs, like most mathematical entities, are inherently visual. In fact, our mathematical intuition relies heavily on our ability to visualize angles, functions, vectors or geometric figures. It fails us, for instance, when we try to visualize a hypercube. However, the

[⋆] This is an extended version of an article presented at SLATE'18 [16].

projection of multidimensional solids in 3- or 2-dimensional spaces gives us an idea of these entities' shape. The visualization of the hypercube is an apt metaphor of the key insight that drives this research: the understanding of a complex entity may be improved by looking at the shadow it casts.

Graphs have a particular role in visualization since they are the data model of most diagrams. Diagrams enrich graphs in two ways: a graphical syntax for nodes and edges; and the layout of nodes and edges on a surface. Different kinds of diagrams have been developed for many purposes. These diagrammatic languages are used for modeling and visualizing relationships among entities, even when they are purely abstract.

In spite of their ability to show relationships and symmetries, large diagrams are difficult to visualize. Too many nodes and too many entangled edges reduce our perception on the underlying graph. This is particularly the case of the semantic web, where graphs are growing increasingly larger and denser. Section 2 presents different attempts to provide visualizations of large semantic graphs, with efficient approaches to process large quantities of data and methods to abstract them, mostly by omitting nodes and edges with certain features. These diagrams represent the graphs themselves, and the layout may highlight general properties, such as symmetry, but usually they do not reveal specific features such as patterns formed by nodes and edges.

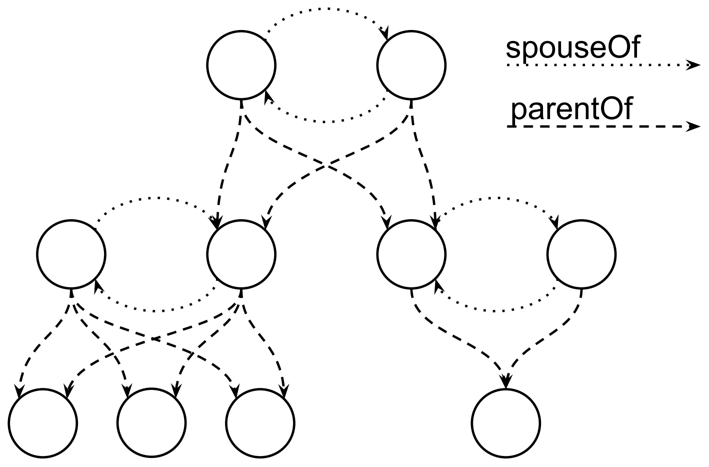


**Fig. 1.** Immediate family relationships graph

Consider the diagram in Figure 1 that depicts immediate family relationships. Nodes represent persons and there are two types of edges: the dotted edges represent the *spouseOf* relationship and the dashed edges represent *parentOf*. The edges in this graph can be summarized by the *a-graph* in Figure 2 that represents edge *types* as nodes. This kind of graph resembles *line graphs* [12] that represent adjacencies between edges of a graph, although a-graphs represent adjacencies between edge types. For that reason, the corresponding a-graph is much smaller than the original graph. Moreover, even if the number of nodes and edges increases in the original graph, the structure of the a-graph may remain unchanged.

For instance, if the immediate family relationship graph was extended to consider all the humans that ever lived, the corresponding a-graph would still be similar to the one depicted in Figure 2.

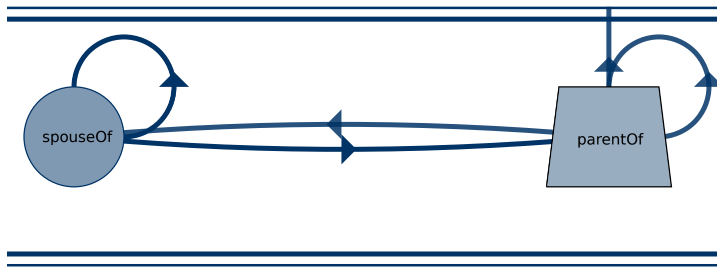


**Fig. 2.** Immediate family relationships a-graph

The novelty of the a-graph approach is the abstraction of large semantic graphs into a much smaller graph highlighting its *path* patterns. The abstraction process consist on mapping semantic graphs into a-graphs, a particular kind of graph with an associated diagram type. Sets of edges with the same type are mapped into nodes and sets of nodes into edges. Section 3 defines a-graphs and their relationship with semantic graphs. It also introduces the a-graph diagrams used for their visualization and provides small examples of this kind of diagram. The final subsection compares a-graphs with other forms of representing the properties of semantic graphs, such as ontologies.

An implementation of the mapping and diagram layout is described in Section 4. It is deployed as a web application for browsing a-graphs and exporting them as vector images, such as the diagram shown in Figure 2. It is available online[1] and is useful to validate the proposed approach. Section 5 presents examples of diagrams produced with this tool to illustrate different techniques to create meaningful visualizations of large semantic graphs and discover relevant path patterns. Section 6 discusses the relationship between a-graphs and ontologies, the efficacy and efficiency of a-graphs to display path patterns and the ability and limitations of the current implementation to manage large semantic graphs. The last section summarizes the a-graph proposal, highlights its main contributions and identifies opportunities for future research.

## 2. Related Work

A-graphs are abstract representations of semantic graphs and thus related to *graph summarization* [18]. This research field covers a wide variety of graph types, from plain graphs described by an adjacency matrix to dynamic graphs described by data streams, encompassing also labelled graphs. Semantic graphs are *typed* and types of nodes and edges can be seen as labels.

The most popular approaches to graph summarization involve compression and grouping techniques. The former is inspired by information theory concepts, namely minimum

[1] `http://quilter.dcc.fc.up.pt/antigraph`

description length (MDL), to detect and compress frequent structures in graphs. The latter aggregates nodes into supernodes connected by super edges to create a new structure, a supergraph. In the case of labelled graphs, grouped nodes are structurally close to each other. Many of these graph summarization approaches are database-centred and only a few emerged from semantic graphs [18]. For instance, Song et al. [19] proposed an approach to summarize knowledge graph characterized by graph patterns, called *d*-summaries, that produce a supergraph as summarization. None of these approaches reverses edges for nodes.

Graphs summarization has several applications. Some of these applications are outside the scope of this research, such as to reduce data volume in storing of massive graphs, to speedup graphs algorithms and queries. Other applications are in tune with the objectives of a-graphs, such as to reduce noise in graphs or to support interactive visualization of massive graphs. However, most approaches to semantic graph visualization described in the literature do not rely on graph summarization.

Knowledge bases such as WordNet[2] [8], Yago[3] [15] and DBpedia[4] [4], have a massive amount of information. A typical representation of these knowledge bases are node-link multigraphs, where each node has a type and nodes are connected by links representing the relationship between them.

A convenient way to analyze this data is using data visualization. The most common type of visualization is focused on the analysis of resources, in particular, those with a high outdegree. The main challenge of semantic graph visualization and management is related to the graph size. This type of graphs has several thousands of nodes and edges and is usually very dense.

The literature presents several approaches to handle the visualization and management of node-link graphs. Most of the related work on massive graphs visualization is handled through hierarchical visualization. This type of approach has low memory requirements, however, it depends on the characteristics of the graph. The graph hierarchy can be extracted using different kinds of methods. Tools such as ASK-GraphView [1], Tulip [3] and Gephi [5] explore clustering and partitioning methods, creating an abstraction of the original graph, using graph summarization, that is easier to visualize. Another technique used to build hierarchies is based on the combination of edge accumulation with density-based node aggregation [21]. Visual complexity can also be reduced by hub-based hierarchies, where the graph is fragmented into smaller components, containing many nodes and edges, making meta nodes, as described in [17]. GrouseFlocks [2] allows users to manually define their own hierarchies.

There are specific tools when the semantic graph is in Resource Description Framework (RDF) format, however, they require loading the full graph. Some desktop-based tools, such as Protégé [5] and RDF Gravityare mainly used with purpose of aiding developers to construct their ontologies, providing also complex graph visualizations. Of all available tools for linked data visualization the most notable ones are the following. Fenfire [13] is a generic RDF browser and editor that provides a conventional graph representation of the RDF model. The visualization is scalable by focusing on one central

[2] `https://wordnet.princeton.edu/`

[3] `https://www.mpi-inf.mpg.de/yago-naga/yago`

[4] `http://wiki.dbpedia.org/`

[5] `http://protege.stanford.edu/`

node and its surroundings. RelFinder[6] [14] is a tool that extracts from a Linked Open Data (LOD) source the graph of the relationships between two subjects. It provides an interactive visualization by supporting systematic analysis of the relationships, such as highlighting, previewing and filtering features. ZoomRDF [20] is a framework for RDF data visualization and navigation that uses three special features to support large scale graphs. It uses *space-optimized* visualization algorithms that display data as a node-link diagram using all visual space available. *Fish-eye zooming* is another feature that allows the exploration of selected elements details, while providing the global context. The last feature is the *Semantic Degree of Interest* assigned to all resources that consider both the relevance of data and user interactions. LODeX [6] produces a high-level summarization of a LOD source and its inferred schema using SPARQL endpoints. The representative summary is both visual and navigable. The platform graphVizdb[7] [7] is a tool for efficient visualization and graph exploration. It is based on a *spatial-oriented* approach that uses a disk-based implementation to support interactions with the graph.

## 3. A-graph definition

The most distinctive feature of a-graphs is that nodes and edges are reversed relatively to the semantic graphs that generated them. Subsection 3.1 explains the motivation behind this decision and characterizes the main components of a-graphs, namely a-nodes and a-edges, as well as their features.

The proposed approach to the visualization of semantic graphs can be divided into two parts. Firstly, the semantic graph is abstracted to another graph – the *a-graph* – that promotes types of edges. Secondly, this abstracted graph is visualized using a special kind of diagram – the *a-graph diagram* – that emphasises path patterns. The following two subsections detail each facet of the a-graph approach.

### 3.1. Motivation

Nodes have the main role in a graph. Edges connect nodes and establish relationships among them. The goal of a-graphs is to abstract a given graph, highlighting edges and reducing its size. Hence, in an a-graph nodes and edges are reversed, i.e. an *a-node* abstracts edges and an *a-edge* abstracts nodes.

It is important to note that an a-node abstracts an edge *type* rather than a single edge. Hence the order (the number of nodes) of an a-graph is in general much smaller than that of the graph it abstracts. For instance, the graph of WordNet 2.1 has about 2 million edges with 27 edge types, hence 27 is the order of the reductions that abstract it.

An a-edge expresses a relationship between a pair of a-nodes, namely that the edge types it represents can be connected to form a length 2 path. Two edges form a length 2 path when the target of the first is the source of the second. Since an a-edge represents a set of nodes, the size (the number of edges) of an a-graph is much smaller than the size of the graph it abstracts. Considering that a-edges can be loops, the number of a-edges is less or equal to $n^2$, where $n$ is the number of a-nodes. For instance, the size of the WordNet

[6] `http://www.visualdataweb.org/relfinder.php`
[7] `graphvizdb.imis.athena-innovation.gr/`

2.1 graph is about half a million but the size of its a-graph is only $214$, well below the maximum of $27^2 = 729$.

The expressiveness of a-nodes and a-edges is increased by adding weights to them. The weight of an a-node is the percentage of edges with the type it abstracts. For instance, if a graph has half of its edges of type $t$ then the a-node reflecting $t$ has weight $1/2$. Hence, a-nodes with higher weight reflect edge types that are more frequent in the graph. Obviously, the sum of a-node weights must be 1.

By the same token, the weight of an a-edge is the percentage of nodes that participate in length 2 paths involving edge types they have as source and target, respectively. For instance, if an a-node reflects the edge type $t_1$ and another the edge type $t_2$, and $1/3$ of the nodes are target of $t_1$ and source of $t_2$, then the weight of the a-edge $t_1 \rightarrow t_2$ is $1/3$.

One would expect every node to be reflected by an a-edge, but for that to happen the nodes that are just sources (not the target of any edge) or just targets (not the source of any edge) must also be abstracted by a-edges. To ensure that all nodes are reflected by a-edges it is necessary to introduce two special a-nodes: *bottom*, denoted as $\perp$; and *top*, denoted by $\top$. The bottom a-node represents a nonexisting edge type that would come before the start of a path. Conversely, the top a-node represents a nonexisting edge type that would come after the end of a path. Both special a-nodes have weight 0, thus maintaining the invariant that the sum of all weights is 1.

The two special a-nodes – bottom and top – allow the definition of a-edges that abstract nodes that are only source or target of edges. These a-nodes are considered *special* to differentiate them from *regular* a-nodes, that have an associated edge type. The a-edge $\perp\rightarrow t$ abstracts the nodes with a null indegree that are sources of edges with type $t$, and the a-edge $t \rightarrow \top$ abstracts the nodes with a null outdegree that are targeted by edges of type $t$.

In fact, both the indegrees and outdegrees of nodes must be taken into consideration in the weight of all a-edges. Consider a node $n$ with indegree 2 and outdegree 3. For instance, if the two incoming edges and the three outgoing are of different types then the contribution of that node to each a-edge is $1/6$. Thus, the weight of an a-edge is the percentage of connecting nodes in paths formed by the edge types, pondered by their in(out)degrees. With this definition, the sum of a-edges weights is also 1.

As explained above, the introduction of the special a-nodes bottom and top is essential to abstract all the nodes of the original graph in a-edges connecting them. One may wonder what other a-nodes types should be considered. It should be noted that a-nodes may have a-edge loops if the graph contains homogeneous paths, i.e. paths formed by a single type of edge. Since the goal of a-graphs is to highlight path patterns, it is important to distinguish different cases that would be amalgamated by generic a-nodes with loops.

Certainly, not all a-nodes have loops. These are considered *shallow* a-nodes since they have at most paths of length 1. In contrast a *deep* a-node has homogeneous paths of higher length through its loop. Special cases of deep a-nodes can be also considered: *cyclical*, where the loops contain homogeneous cycles, i.e. cycles using only the type of edge represented by the a-node; and *hierarchical*, where the loops represent confluent paths, i.e. where the nodes in homogeneous paths have branching factor above or equal to 2. These types provide information on the kind of paths formed “within” an a-node, similar to the information that can be extracted from other a-edges relating different a-nodes.

In summary, an a-graph is an abstraction of a semantic graph. This does *not* mean that an a-graph is a sort of schema. A semantic graph does not comply with its a-graph, it is the other way round: a-graphs have a functional dependency to semantic graphs. Thus, the information provided by an a-graph is of a different nature of that of an RDF or OWL ontology, as discussed in Subsection 6.1.

## 3.2. Abstraction map

The previous subsection introduced the concepts of a-node, a-edge and their weights, as well as the map between a semantic graph and the a-graph that it abstracts. This subsection formalizes those concepts, starting by precising the concept of semantic graph. A semantic graph $\mathbb{G}$ can be defined as a tuple $\mathbb{G} = (N, E, T_N, T_E, \mathrm{t_N}, \mathrm{t_E})$ where:

**set of nodes** $N$
**set of edges** $E \subseteq \{(s,t) : s \in N \wedge t \in N\}$
**set of types of nodes**[9] $T_N$
**set of types of edges**[10] $T_E$
**types of nodes** $\mathrm{t_N} : N \to T_N$
**types of edges** $\mathrm{t_E} : E \to T_E$

The *a-graph* $\mathbb{A}$ of graph $\mathbb{G}$ is produced by a map defined as

$$\begin{aligned} \mathrm{abstract}\colon \mathbb{G} & \to \mathbb{A} \\ (N, E, T_N, T_E, M_N, M_E) & \mapsto (N', E', \mathcal{W}'_N, \mathcal{W}'_E, \mathrm{t}_{N'}) \end{aligned}$$

It should be noted that the a-graph $\mathbb{A}$ is an abstraction of a semantic graph $\mathbb{G}$, not itself a semantic graph. The a-graph $\mathbb{A}$ is a tuple where:

**set of a-nodes** $N' = T_E \cup \{\top\ \bot\}$
**set of a-edges** $E' = E'_0 \cup E'_\bot \cup E'_\top$
**weight of a-nodes** $\mathrm{w_N} : N' \to [0,1]$
**weight of a-edges** $\mathrm{w_E} : E' \to ]0,1]$
**type of a-nodes** $\mathrm{t}_{N'} : N' \to T'$ where
$T' = \{$shallow deep cyclic hierarchical top bottom$\}$

As defined above, the set of a-nodes is the union of types of edges of $G$ with special nodes $\top$ (top) and $\bot$ (bottom). The definition of the set of a-edges is also the union of three sets, namely the set of regular a-edges $E'_0$, the set of bottom a-edges $E'_\bot$, and the set of top a-edges $E'_\top$.

An ordered pair $(\bot, t')$ is in the set $E'_\bot$ if there is an edge of type $t'$ where the source node $s$ has a null indegree ($\deg^-(s) = 0$).

[9] In an RDF graph, this would be set of URIs referring to resources rather than a set of RDFS classes
[10] In an RDF graph, this would be set of URIs referring to properties

$$E'_\perp = \{(\perp, t') \colon t' \in T_E \wedge (\exists (s,t) \in E \colon \mathrm{t_E}((s,t)) = t') \wedge \deg^{-}(s) = 0 \}$$

Similarly, an ordered pair $(s', \top)$ is in the set $E'_\top$ if there is an edge of this type where the target node $t$ has a null outdegree ($\deg^{+}(t) = 0$).

$$E'_\top = \{(s', \top) \colon s' \in T_E \wedge (\exists (s,t) \in E \colon \mathrm{t_E}((s,t)) = s') \wedge \deg^{+}(t) = 0 \}$$

Finally, an ordered pair $(s', t')$ of a-nodes is in a set of regular a-edges if there is a path in the graph involving the two edge types.

$$E'_0 = \{(s', t') \colon s' \in T_E \ \wedge \ t' \in T_E \wedge (\exists (s,m) \in E \colon \mathrm{t_E}((s,m)) = s') \wedge (\exists (m,t) \in E \colon \mathrm{t_E}((m,t)) = t') \}$$

By definition, the weight of the special a-nodes, top and bottom, is null; and these are the only a-nodes with null weight. The nonnull weight of a regular a-node $n'$ is the ratio between the number of edges with that type and the total number of edges

$$\mathrm{w_N}(n') = \frac{\sharp\{e \colon e \in E \wedge \mathrm{t_E}(e) = n'\}}{\sharp E}$$

The weight of an a-edge must be computed differently when its source $s'$ or target $t'$ have special a-nodes. If the source $s' = \perp$ then the n-tuple $B_{t'}$ of nodes to consider contains those that are sources of an edge of type $t'$ with a null indegree. The reader should note that this is an n-tuple rather than a set, where each node $s$ may appear more than once. The order of the nodes in the n-tuples is immaterial. The purpose of the n-tuples is merely to count the number of nodes. In all n-tuples $B_{t'}$, as defined below, each node $s$ is repeated as many times as its outdegree $\deg^{+}(s)$.

$$B_{t'} = (s \colon (s,t) \in E \wedge \mathrm{t_E}((s,t)) = t' \wedge \deg^{-}(s) = 0)$$

Similarly, if the target $t' = \top$ then the n-tuple of nodes to consider is those that are the target of an edge of type $s'$ with a null outdegree. In all n-tuples $T_{s'}$ each node $t$ appears repeated as many times as its indegree $\deg^{-}(t)$.

$$T_{s'} = (t \colon (s,t) \in E \wedge \mathrm{t_E}((s,t)) = s' \wedge \deg^{+}(t) = 0)$$

Otherwise, if none of them is a special a-node then the nodes to consider are those that participate in paths of length 2 where the first edge has type $s'$ and the second has type $t'$. In this case the node $m$ appears repeated $\deg^{-}(m) \deg^{+}(m)$ times.

$$R_{s'}^{t'} = (m\colon (s,m) \in E \wedge (m,t) \in E \wedge \mathrm{t_E}((s,m)) = s' \wedge \mathrm{t_E}((m,t)) = t')$$

The weight of an a-edge $\mathrm{w_N}((s',t'))$ sums the contribution of n-tuples sets according to each case. The contribution of each node is pondered with the inverse of its indegrees or outdegrees, when these are not null. Thus, the definition of weight function is the following,

$$\mathrm{w_N}((s',t')) = \begin{cases} \frac{1}{\sharp E} \sum_{s \in B_{t'}} \frac{1}{\deg^+(s)} & \text{if } s' = \bot \\ \frac{1}{\sharp E} \sum_{t \in T_{s'}} \frac{1}{\deg^-(t)} & \text{if } t' = \top \\ \frac{1}{\sharp E} \sum_{m \in R_{s'}^{t'}} \frac{1}{\deg^-(m)\deg^+(m)} & \text{otherwise} \end{cases}$$

Finally, the $\mathrm{t}_{N'}$ function maps a-nodes to a type in $T'$. The definition of this function is based on the concept of *graph reduction*. A reduction of the graph $\mathbb{G}$ by the type $t \in T$ is the largest subgraph of $\mathbb{G}$ that has only edges of type $t$ and without disconnected nodes; i.e. nodes that are not the source or target of edges of type $t$ are removed. It should be noted that in general a graph reduction $\mathbb{G}_t$ has many strongly connected components, in some cases as many as the number of edges (a value property, for instance). Consider the following functions defined over the set of $\mathcal{G}$ of all graph reductions:

**size of the largest path** $\mathrm{slp} : \mathcal{G} \to \mathbb{N}$
**number of cycles** $\mathrm{nc} : \mathcal{G} \to \mathbb{N}$
**average branching factor** $\mathrm{abf} : \mathcal{G} \to \mathbb{R}$

Using these functions over graph reductions the a-node type function is defined as follows.

$$\mathrm{t}_{N'}(n') = \begin{cases} \text{shallow} & \text{if } n' \in T_E \wedge \mathrm{nc}(\mathbb{G}_{n'}) = 0 \\ & \quad \wedge\, \mathrm{slp}(\mathbb{G}_{n'}) < 3 \\ \text{deep} & \text{if } n' \in T_E \wedge \mathrm{nc}(\mathbb{G}_{n'}) = 0 \\ & \quad \wedge\, \mathrm{slp}(\mathbb{G}_{n'}) >= 3 \wedge \mathrm{abf}(\mathbb{G}_{n'}) < 2 \\ \text{hierarchic} & \text{if } n' \in T_E \wedge \mathrm{nc}(\mathbb{G}_{n'}) = 0 \\ & \quad \wedge\, \mathrm{slp}(\mathbb{G}_{n'}) >= 3 \wedge \mathrm{abf}(\mathbb{G}_{n'}) >= 2 \\ \text{cycle} & \text{if } n' \in T_E \wedge \mathrm{nc}(\mathbb{G}_{n'}) > 0 \\ \text{top} & \text{if } n' = \top \\ \text{bottom} & \text{if } n' = \bot \end{cases}$$

The codomain of $\mathrm{t}_{N'}$ captures a number of elemental path patterns involving a single edge type. Shallow a-nodes correspond to paths with a single edge. Deep a-nodes correspond to larger paths such as those resulting from transitive edges. Hierarchic a-nodes are in fact a special case of deep a-nodes, where paths converge to a single, or a set of, root nodes; these edges form hierarchies and are particularly interesting for discovering edges types for path based semantic measures [10]. Finally, cyclic a-nodes are created by reflexive edges.

The constants in the definition of $\mathrm{t}_{N'}$ require some explanation. The threshold of 0 in the number of cycles [$\mathrm{nc}(\mathbb{G}_{n'})$] to distinguish a cycle from other regular types is rather obvious, but *not* the threshold of 3 in the size of largest path [$\mathrm{slp}(\mathbb{G}_{n'})$]. A threshold of 2 was, in fact, the first choice, but there are cases where paths of size 3 occur without changing the type of an a-node. The most notable example is *rdf:type*. Since RDF types have themselves a type (*rdfs:Class*), length 3 paths are usual in semantic graphs, but they should be considered shallow and not hierarchic. Finally, with an average branching factor [$\mathrm{abf}(\mathbb{G}_{n'})$] above or equal to 2 the paths in a graph reduction form a hierarchy that is suitable for classification. That is usually the case of *rdfs:subClassOf*, for instance, that in conjunction with *rdf:type* creates a taxonomic relationship [10].

## 3.3. Diagram language

As explained in the previous Subsection, an a-graph is an abstraction of a semantic graph. The a-graph diagram language is a visual representation of an a-graph intended to highlight the path patterns of the abstracted semantic graph. An a-graph has a-nodes of different types connected by a-edges.

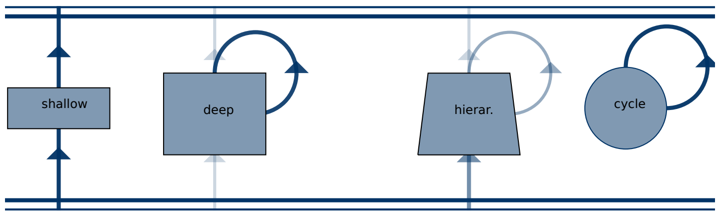


**Fig. 3.** Catalog of a-node types

The type of an a-node is conveyed by its shape. A shallow a-node is represented by a horizontal rectangle, while a deep a-node is represented by a vertical rectangle. The height of these rectangles is a visual cue of the path sizes contained in these a-nodes. A cyclical a-node is represented by a circle or an ellipse, and a hierarchical a-node is represented by an isosceles trapezoid. The position of these shapes is their geometric center. The a-graph depicted in Figure 3 is a sort of catalog of a-node types, where the label of each regular a-node is the type's name.

The bottom and top a-nodes are represented by a pair of parallel lines rather than shapes. As can be seen also in Figure 3, the parallel lines that represent each of these a-nodes have different widths. The bottom a-node has a larger upper line and the top a-node is the reverse. The bottom and top a-nodes are located respectively at the bottom and top of the diagram, as their names suggest. This way the paths created by a-edges tend to be directed upwards.

Unlike a-nodes, a-edges have a single type. Hence, they are represented all by solid lines with an arrowhead positioned in their middle pointing to the target. Lines connecting from the bottom a-node, or to top a-node, are straight. All the others are curved so that a-edges with opposite directions do not overlap.

A-nodes and a-edges have weights in the $[0, 1]$ interval. Actually, both regular a-nodes and a-edges have always nonnull weights; special a-nodes (top and bottom) have null weights by definition. The nonnull weights of regular a-nodes and a-edges are conveyed graphically too. The weight of an a-node is shown as a transparency, making dimmer the a-edges representing a smaller number of edges in the abstracted graph. The same principle applies to weights of a-edges. In this case, the weight is also shown as line width, making thicker the a-edges that represent a larger number of nodes. The semantic graph that originated the a-graph in Figure 3 has all edge types with the same number of edges, hence all a-nodes have the same weight, thus they all have the same shade. A different thing happens with a-edges; each has a different shade, reflecting their different weights.

The regular a-nodes in the catalog diagram are not connected to each other, just to bottom and top (with the exception of the cycle). This means that they do not form "joins". Using a syntax borrowed from SPARQL, it can be said that the semantic graph that generated it lacks triple patterns of the form

```
?a ?p ?b .
?b ?q ?c .
```

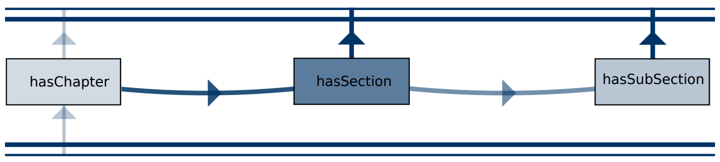


**Fig. 4.** Book structure

The example in Figure 4 represents the structure of books, where a book has chapters and these have sections. The a-graph of such semantic graph has the properties *hasChapter*, *hasSection* and *hasSubSection*.

In this case "joins" are created using multiple edge types hence the a-nodes have a-edges connecting them. In particular *hasChapter* is connected to *hasSection* and this to *hasSubSection*. The reader should note that the three regular a-nodes are connected to the top, meaning that there are chapters, sections and subsections that are not subdivided, and that only *hasChapter* is connected from bottom, meaning that only these are connected from root elements of the hierarchy.

The previous example reflects a hierarchical structure, although with a different type of edge for each layer. The example in Figure 2 reflects a semantic graph with a couple of family relationships, namely *spouseOf* and *parentOf*. Their associated a-nodes both have loops, which means that paths with a single type of edges can be created. The *parentOf* a-node has hierarchic as type, meaning that paths of length greater or equal to 3 can be created and has an average branching factor greater or equal to 2.

The simple patterns identified in the small examples above occur also in larger semantic graphs. Section 4 presents an a-graph browser that allows us to discover combinations of these patterns in larger examples, as those analyzed in Section 5.

## 4. A-graph browser

This section describes the design and implementation of a web application developed to validate the concept of a-graph. This web application – the a-graph browser – produces interactive a-graph diagrams from several data sources and is freely available online[11].

The a-graph browser is a Java web application developed with the Google Web Toolkit (GWT). It is composed of a client front-end running on a web browser and a server back end. The server is responsible for transforming a semantic graph in RDF format into an a-graph that is sent to the client. The front end is responsible for laying out diagrams and managing user interaction, as explained in the following subsections.

### 4.1. Back end processing

The mapping of semantic graphs into a-graphs is performed in two stages by the back end. Firstly, a set of graph reductions is produced from the semantic graph triples. Secondly, the a-graph data is computed by processing these graph reductions.

A graph reduction instance aggregates edges of a single type, that is, the semantic graph obtained by considering only the edges of that type. It records the nodes that are sources and those that are targets, and computes their in and outdegrees. The links between these nodes are also recorded to compute aggregate measures on the reduction such as the number of cycles, depth and branching factor.

Graph reductions are computed by processing a stream of RDF triples. For each subject-predicate-object triple the reduction corresponding to its predicate is selected, with the subject recorded as source and the object as target.

Each reduction corresponds to an a-node. Thus the second stage creates an a-node for each reduction found in the first stage, assigning it a weight computed as the percentage of edges in the graph. The top and bottom a-nodes, with null weight, are also created. Then it iterates over the pairs of reductions to create a-edges.

The computation of a-edges' weights is more complex than that of a-nodes, as it involves determining the intersection of the targets and source sets of nodes respectively of the source and target a-nodes of each a-edge. Also, the contribution of each of these nodes depends both on their in(out)degrees on the reduction. The pairs of a-nodes with nonnull weights create a-edges.

A-edges connecting a-nodes to top and bottom need also to be considered. These are created with the nodes that are not fully consumed to create a-edges among regular a-nodes, following the same approach to compute weights. It should be noted that links between top and bottom are impossible.

### 4.2. Diagram layout

A-graphs serialized in JSON are sent to the front end where they are visualized as diagrams. The layout of these diagrams is computed using a force-directed algorithm [11]. A-nodes repel each other according to Coulomb's law as if they were electrically charged particles with the same signal. A-edges bind them together as springs following Hooke's law.

[11] `http://quilter.dcc.fc.up.pt/antigraph`

The top and bottom a-nodes, as well as the a-edges connecting them, are ignored in this process. The layout is performed in a rectangular area that acts as a boundary that confines regular a-nodes. Top and bottom a-nodes are positioned respectively at the top and bottom of this rectangle, and a-edges connecting then are plotted perpendicularly to them.

One of the advantages of a force-directed algorithm is that it adjusts to changes, either of window dimensions or in the number of nodes. This enables the selection of a-nodes, choosing which to display and which to hide, with the quick readjustment of the layout. When an a-node is hidden so are the a-edges that link to it. Nevertheless, this algorithm may introduce undesirable changes due to user actions. To remedy this issue, the incremental layout can be toggled on or off, as explained in the next subsection.

A-graphs with a large number of a-nodes tend to have an even larger number of a-edges, cluttering the layout. In this case, the natural candidates to hide are those with smaller weight since they represent a smaller number of edges in the semantic graph. To simplify this kind of selection the a-graph browser provides a node weight threshold. If this threshold is provided then a-nodes are sorted by weight and their accumulated weight is computed in this order. When this value exceeds the threshold the remaining a-nodes are hidden, as well as the a-edges linking to them.

## 4.3. User interface

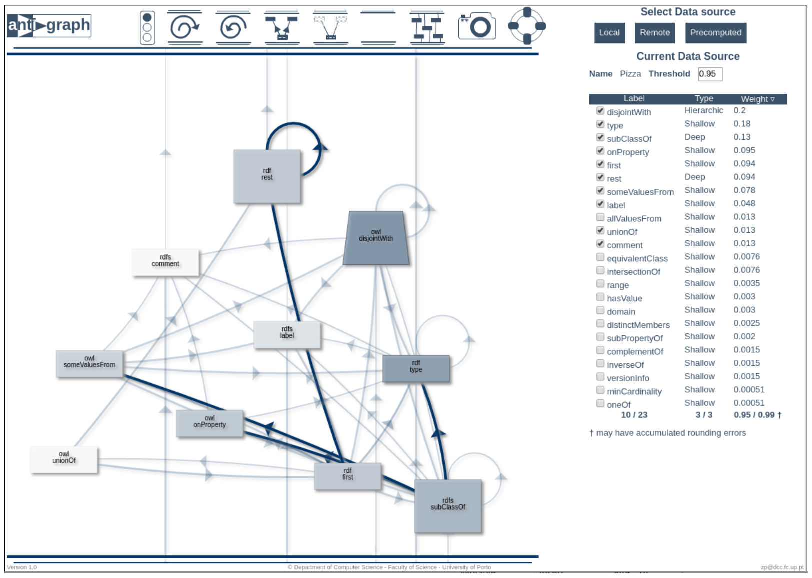
Figure 5 depicts the user interface of the a-graph browser available online. The main part is the left central region where the diagram is displayed, following the approach described in the previous subsection. Above this area, there is a toolbar with tools for controlling the diagram layout. The smaller panel on the right contains a data source selector and displays the current data source features. The remainder of this subsection describes these panels in detail.

The a-graph browser has a number of features to control the diagram layout. These features are accessible through the icons on the header toolbar. To start with, the incremental layout can be toggled on and off using the traffic light icon, on the left of the toolbar. The icons to its left provide ways to show and hide a-nodes, as well as the a-edges connecting them. The most relevant (with higher weight) hidden a-node is shown by pressing the outward spiral icon. Using this tool it is possible to gradually enlarge the diagram. The reverse tool, bound to the inward spiral icon, hides the least relevant shown a-node.

The following two icons operate on the currently selected a-node: to show all currently hidden a-nodes connected to it, or to hide all a-nodes connected to it. A-nodes are selected just by clicking on them. Clicking an a-node with the mouse's middle button also toggles a tool tip hovering the node. This tool tip displays the characteristics of the a-node, such as label, type and weight.

The hide all and show all tools allow the user to set the layout at the two extremes. These tools are respectively bound to the icons with an a-graph with no a-nodes and the a-graph with several a-nodes and a-edges. The header toolbar includes two other icons on its right side: the camera icon and the life saving icon. The latter opens a help window expanding the information in this paragraph.

The camera icon produces a *vector* image of the diagram presented in the browser. Using the normal browser features, it is possible to obtain a raster image of the diagram.



Select Data source

Local Remote Precomputed

Current Data Source

Name Pizza Threshold 0.95

| Label | Type | Weight ▿ |
|---|---|---|
| ✓ disjointWith | Hierarchic | 0.2 |
| ✓ type | Shallow | 0.18 |
| ✓ subClassOf | Deep | 0.13 |
| ✓ onProperty | Shallow | 0.095 |
| ✓ first | Shallow | 0.094 |
| ✓ rest | Deep | 0.094 |
| ✓ someValuesFrom | Shallow | 0.078 |
| ✓ label | Shallow | 0.048 |
| ☐ allValuesFrom | Shallow | 0.013 |
| ✓ unionOf | Shallow | 0.013 |
| ✓ comment | Shallow | 0.013 |
| ☐ equivalentClass | Shallow | 0.0076 |
| ☐ intersectionOf | Shallow | 0.0076 |
| ☐ range | Shallow | 0.0035 |
| ☐ hasValue | Shallow | 0.003 |
| ☐ domain | Shallow | 0.003 |
| ☐ distinctMembers | Shallow | 0.0025 |
| ☐ subPropertyOf | Shallow | 0.002 |
| ☐ complementOf | Shallow | 0.0015 |
| ☐ inverseOf | Shallow | 0.0015 |
| ☐ versionInfo | Shallow | 0.0015 |
| ☐ minCardinality | Shallow | 0.00051 |
| ☐ oneOf | Shallow | 0.00051 |
| **10 / 23** | **3 / 3** | **0.95 / 0.99 †** |

† may have accumulated rounding errors

**Fig. 5.** A-graph browser

However, this kind of image is inadequate for publication since it has a fixed and typically low resolution. The camera icon activates a feature that produces an SVG file with the diagram, using the same layout algorithm described above. This conversion uses the SVGKit[12] package, that works well for graphic primitives (e.g. lines, rectangles, ellipses) but has some limitations regarding fonts and shadows. The vector images look slightly different from their raster counterparts, but have better quality when printed. The diagrams of the next section, as well as those of Subsection 3.3, were produced using this tool.

The a-graph browser presents a second panel next to the diagram. Depending on the width of the web browser's window, this panel may be placed either to the right side (as in Figure 5) or below the diagram. The panel contains a data source selector and displays the main features of the current data source.

The upper part of the side panel is used for selecting a semantic graph as data source for generating an a-graph. It provides three kinds of semantic graph sources: local, remote and precomputed.

Local sources include small examples for testing the basic features of a-graphs, and were presented in Subsection 3.3. The dialog box for the selection of local graphs presents the RDF triples that will be processed to produce the a-graph. These triples are in N-Triples format in an editable window. The user may modify, add or delete these triples, to better understand how these changes are reflected on the a-graph diagram.

The remote sources are RDF graphs available on the web in XML/RDF format. This dialog box presents each graph's URLs and a threshold – the weight above which a-nodes are included in the diagram. The last entry of this dialog box allows the user to enter a

[12] `http://svgkit.sourceforge.net/`

URL to any RDF/XML file available on the web, and assign it an initial threshold. This threshold may be changed later on the current data source panel.

Both the local and remote data sources are processed on the fly by the server. The precomputed data sources provide access to larger semantic graphs that require long processing times and are already available on the client side. Most of these examples are analyzed in detail in the next section.

The current data source panel displays its name, threshold and a grid listing its a-nodes. This grid lists all the a-nodes in the a-graph, showing which are currently visible, their type and weight. By default, this information is ordered by descending a-node weight, but the user can change it. The user can also (de)select the visible a-nodes, which immediately changes the diagram layout. Also, changes in the diagram resulting from the tools described above are also immediately reflected in this grid.

## 5. Validation

This section shows with concrete examples how a-graph diagrams emphasize the most relevant path patterns of a semantic graph. It also explains how the tools in the a-graph browser help the discovery of path patterns in large semantic graphs, by temporarily hiding some of their a-nodes and the a-edges connecting them, thus producing meaningful diagrams with a reasonable small size.

The a-graph browser has an example selector on its right top corner. These examples are divided according to their availability: local examples, with their source text available for inspection and editing, including the possibility of creating one from scratch; remote examples, published on the web in RDF formats, including the possibility of providing a URL; precomputed examples, of large freely available semantic graphs, that require a much larger processing time. The data sources for these examples must be in RDF format, either serialized in RDF/XML or in NTriples. The local examples include all a-graphs presented in Subsection 3.3.

### 5.1. WordNet

Wordnet[8] 2.1, whose a-graph diagram is depicted in Figure 6, is a much larger graph than those presented in Subsection 3.3. However, this figure refers only to 95% of Wordnet 2.1 since the 5% least representative edges are omitted. By default, when this example is selected the threshold is set to 95%, but this value may be edited or removed in the corresponding field.

The WordNet 2.1 graph has 27 types of nodes and their corresponding a-nodes would clutter this figure. This approach quickly produces a simple visualization by temporarily hiding the $2/3$ least representative a-nodes, i.e. edge types. It is important to point out that this is not specific of WordNet. All the semantic graphs tested with the a-graph browser have most of their paths concentrated in a fairly small number of edge types, hence this approach can be systematically used to improve the a-graph visualization.

This diagram immediately shows that the edges types that participate in most triples are from imported namespaces – *rdf:type* and *rdfs:label* – since the corresponding a-nodes are darker. Two a-nodes of the *wn20schema* namespace stand out from the pack for having links to several others, namely *hyponymOf* and *containsWordSense*, but the former participates in more “joins”, as evidenced by the darker a-edges.

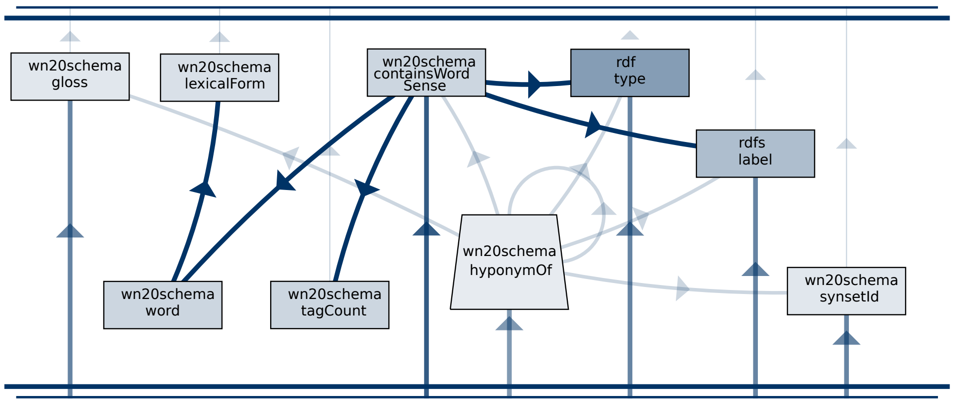


**Fig. 6.** An a-graph diagram of WordNet 2.1

WordNet is frequently used as a semantic proxy by path based semantic measures [10]. These measures rely on taxonomic relationships to identify a least common ancestor between two concept nodes and compute the shortest path between them. Taxonomic relationships are created using *partOf* (hierarchical) and *isA* relationships. For instance, the RDF and RDFS vocabularies provide the `rdf:type` and `rdfs:subclassOf` properties that can be used to create a taxonomic relationship between typed resources. However, in this version of WordNet `rdf:type` is available, but `rdfs:subclassOf` is missing.

The a-graph diagram in Figure 6 shows how the hierarchic relationship `hyponymOf`, complemented with another relationship, can be used to create a taxonomic relationship. The `rdfs:label` relationship is connected by an a-edge with `hyponymOf`, hence they can be combined to create a taxonomic relationship on words.

Of course, this is not new knowledge. It is well known that WordNet can be used as a semantic proxy using `hyponymOf` and another property to create a taxonomic relationship. The point is that the a-graph diagram highlights the most promising candidates to create a taxonomic relationship. This should be useful to discover candidates for taxonomic relationships in even larger semantic graphs, such as DBpedia [4].

## 5.2. Yago

Yago[13] [15] is a well known semantic knowledge base derived from several sources, such as DBPedia, WordNet, and GeoNames. It has over 10 million entities but for this study, only the core was used and labels were omitted. Still, this corresponds to over 20 million triples with 60 property types. Hence it produces an a-graph with that order and size 487. Even with a threshold of 80%, as it is by default on the a-graph browser, it is difficult to grasp.

The diagram in Figure 7 was obtained by selecting a single a-node, *hasArea*, the second most frequent edge type in this graph. Afterward, it was used the unhide tool to show a-nodes connected to the one currently selected. The point is to find property types related to concepts that have an area. Examples of such concepts would be cities, regions or

[13] `https://www.mpi-inf.mpg.de/yago-naga/yago`

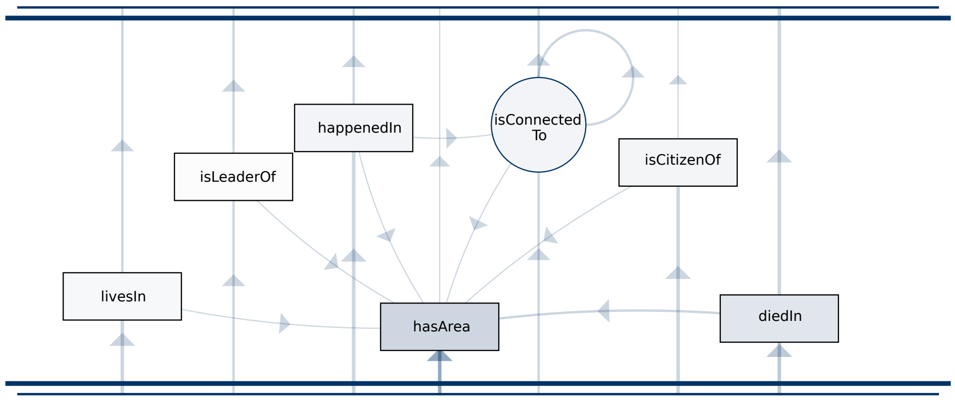


**Fig. 7.** Yago core - a-nodes connecting to *hasArea*

**Listing 1.1.** SPARQL query to count leaders of geographic areas

```
SELECT COUNT(*)
WHERE {
   ?p  yago:isLeaderOf ?g.
   ?g  yago:hasArea ?a.
}
```

countries. In a sense, *hasArea* can be seen as a defining property for a class of geographic concepts, although that is not explicit. The diagram shows that these geographic concepts are connected to other properties, such as *livesIn*, or *isLeaderOf*. That is, it is possible to retrieve information about who lives in or who is the leader of an concept that has an area. The SPARQL query in Figure 1.1 should produce a nonempty result set. In fact, it was checked on a Yago SPARQL endpoint[14] and the result is 5666.

**Listing 1.2.** Counting places connected to where something happened

```
SELECT COUNT(*)
WHERE {
   ?s yago:happenedIn ?g .
   ?g yago:isConnectedTo ?p .
}
```

Also, one can determine the area of entities where something happened, *happenedIn*, or that are connected to each other. The type of this last a-node is cyclic, meaning that its corresponds to a reflexive edge type. These two a-nodes are the only that are directly connected without using *hasArea*. Hence, it must be possible to obtain a non empty answer

[14] https://linkeddata1.calcul.u-psud.fr/sparql

to the query “what places are connected to the place where something happened?”, using the query in Listing 1.2, and it actually returns 888 solutions.

Surprisingly, the graph also indicates that one should not expect results for the query “what places are connected to the place of citizenship of x” since these two a-nodes are not connected. Running the SPARQL query in Listing 1.3 verifies that conclusion as the result is zero.

**Listing 1.3.** Are citizens connected to other places?

```
SELECT COUNT(*)
WHERE {
  ?s yago:isCitizenOf ?g .
  ?g yago:isConnectedTo ?p .
}
```

## 5.3. DBLP

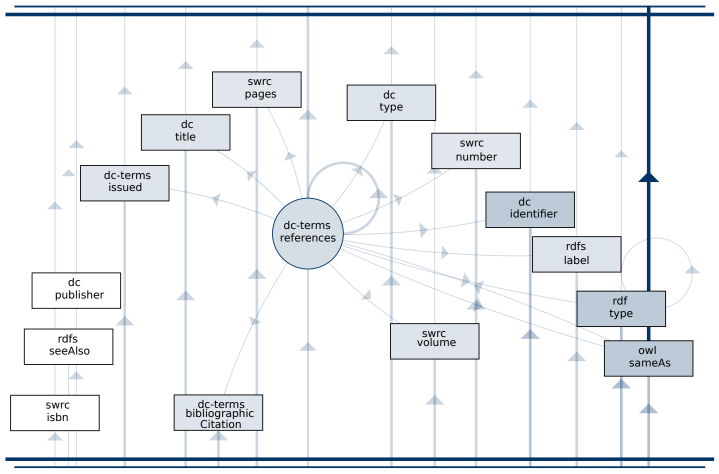


**Fig. 8.** DBLP open bibliographic information on computer science publications

DBLP[15] is an on-line reference for open bibliographic information on computer science journals and proceedings that publish its data in RDF format. Although it has a massive size, about 134 million triples, it can be processed by sampling since its RDF file has a regular structure. It is a sequence of blocks of triples, each block corresponding to a publication. The diagram depicted in Figure 8 was obtained by processing the initial 1%

[15] `http://dblp.uni-trier.de/`

of DBLP's RDF file. This approach was possible since this file keeps in consecutive lines the triples related to a single author. If it were the case that lines were sorted by property URL, for instance, then the first 1% of the file would not provide a meaningful representation of the complete graph. Thus, this approach cannot be applied systematically to any semantic graph since it assumes a uniform distribution of property URLs in the stream of triples.

The a-graph of DBLP reveals different patterns. The most simple are the a-nodes on the left side of the diagram, such as *swrc:isbn*, *dc:publisher* and *rdf:seeAlso* that simply connect the bottom to the top and not to any other a-node. Most of the other a-nodes have also an a-edge from the a-node *dc-terms:references* that thus assumes a pivotal role in this diagram and is the only cyclic a-node. The only other a-node with a loop is *rdf:type*, most probably due to the type of a class. Apart from these cycles formed by edges of the same type, there are also cycles with mixed types, composed of *owl:sameAs* and *dc-term:references*.

## 6. Discussion

The goal of a-graphs is to provide visualizations of path patterns in order to give new insights on large semantic graphs. In this section, we discuss if the concept of a-graph and the current implementation of the a-graph browser meet this goal. Three main questions are analyzed: how do a-graphs compare with ontologies in describing semantic graphs? do a-graphs provide more information regarding paths patterns than classical visualization tools? can a-graphs be computed in a reasonable time for large semantic graphs?

### 6.1. A-graphs and ontologies

Ontologies and a-graphs are somehow related in the sense that they both abstract semantic graphs. Thus, it is relevant to question if these two concepts – ontologies and a-graphs – overlap or compete in any way.

Semantic graphs are frequently encoded as sets of triples in the Resource Description Framework (RDF). This framework supports multiple vocabularies, including a vocabulary to describe other vocabularies – RDF Schema (RDFS) – which in turn lays the foundations for a richer ontological language – OWL. RDFS and OWL describe vocabularies in terms of classes and properties, where classes provide types for nodes and properties types for edges of semantic graphs, and define hierarchical relationships among those types.

The definition of semantic graph presented in Subsection 3.2, on which the definition of a-graphs relies, is also based on types. However, these types are of a different nature. These node and edge types are not RDFS or OWL classes and properties, and they are not hierarchically related among themselves. The node and edge types in the definition of a-graphs are the actual URIs used to label them.

The concept of ontology varies for different communities [9]. In the semantic web, an ontology is usually understood as a formal definition of a domain of discourse. It declares a taxonomy of concepts and relationships among them. For instance, an ontology may declare *cat* and *dog* as classes, both as subclasses of *pet*, and the property *hasName* associating pets to their names (strings). RDFS and OWL ontologies are themselves RDF graphs, although not all RDF graphs are ontologies. In fact, most RDF graphs assert facts on resources using types and properties, such as "Rex is a dog"[16], but they do not define hierarchies of classes (concepts) and properties (relationships).

By using inference with an ontology it is possible to entail new facts from existing ones, such as "Rex is a pet". The reverse, to induce an ontology from a collection of facts, is much more complex. It is possible to process statements such as "Rex is a dog" and "Fifi is a cat", "Rex is a pet" and "Fifi is a pet" and induce an ontology similar to the example in the previous paragraph. However, ontologies are not usually created this way.

Ontologies prescribe how certain semantic graphs must be. They are not summarizations of existing semantic graphs. Also, if an ontology is applicable to a particular semantic graph then the latter should be consistent with the former; and as more facts are added to the graph that consistency should be preserved without changing the ontology.

An a-graph is, in fact, a summarization of a semantic graph. It maps edges into a-nodes and nodes into a-edges in a way that the a-graph paths condense several paths of the semantic graph it abstracts. However, only paths that actually exist in the semantic graph are abstracted into a-graph paths, not all the paths that would be consistent with the ontology. Moreover, since a-nodes and a-edges have weights, the path frequency is also presented by the a-graph, which has no parallel in ontologies. As a semantic graph evolves and new nodes and edges are added (or removed), its a-graph may change to reflect it. In some cases, only the weights will be affected, if no kinds of path are created. In other cases, new a-nodes result from edge types that did not exist before.

In summary, a-graphs and ontologies are different kinds of abstractions. A-graphs abstract paths, highlighting the most frequent ones. Ontologies abstract relationships among concepts. The two abstractions are non-overlapping and are in fact complementary.

## 6.2. Path visualization

A-graphs were designed to discover path patterns in semantic graphs. In this particular point, a-graphs are quite different from other forms of graph visualization, including semantic graph visualization, surveyed in Section 2. These approaches display the actual paths, connecting a sequence of nodes, but do not reveal patterns in these paths.

A path in an a-graph corresponds to a *collection* of paths in the semantic graphs it reflects. Since edges types are replaced by a-nodes, a-graphs put edge types in evidence and aggregate a large number of nodes in a-edges. Paths built from a single edge type are condensed in a-nodes types. Hence, a-node types such as hierarchic or cyclic already condense path patterns.

Weights of a-nodes and a-graphs provide information on their relevance. This extra information can be used both for visualization and browsing. Weights are translated to colors and line widths in a-graphs to indicate the relevance of particular path patterns. Moreover, weights can be used to browse a-graphs and hide the least relevant a-nodes.

The examples analyzed in Section 5 illustrate the ability of a-graphs to reveal information about path patterns. Figure 6 immediately shows the existence of a hierarchic edge

[16] "Rex is a dog" are two RDF facts. Assuming `ex` as an alias for a namespace, that sentence would be represented by the RDF facts
```
ex:rex ex:hasName ``Rex''
ex:rex rdf:type ex:dog
```

type and the edge types that connect it, which is relevant information for someone building a semantic measure. Figure 7 displays a number of edge types related to a particular one that characterizes geographic places. It provides information on the type of path patterns that can be used in SPARQL queries, estimating the possibility of returning non empty result sets, which is relevant to someone interested in extracting information from a triple store.

Surely the kind of insight provided by a-graphs is relevant to a specialized group of users, those interested in path patterns rather than in individual paths. This is an issue since it makes more difficult a thorough validation of the a-graph browser involving actual users. The author acknowledges that such validation is necessary to unequivocally prove the usefulness of a-graphs. However, this kind of users are difficult to find, hence this validation has yet to be done.

## 6.3. Visualization of large semantic graphs

Discovering path patterns is particularly important in large semantic graphs. Hence, it is important to be able to plot a-graph diagrams for such graphs within a reasonable time. Plotting a-graph diagrams using a force-directed algorithm takes only a few seconds. The time-consuming part is the mapping between semantic graphs and a-graphs.

The computational complexity of the mapping process described in the previous sections is linear on the number of triples, nodes, and property types. The first stage has complexity $O(t)$ where $t$ is the number of triples in the semantic graph and the second stage as complexity $O(p*n)$ where $p$ is the number of property types and $n$ the number of nodes. Nevertheless, to process larger semantic graphs some optimizations are required.

Although the computational complexity of the a-graph mapping is small, processing large sources takes several days. As explained before, the process has two stages, each with a different complexity. Consider a large semantic graph with $k$ triples, $t$ edge types and $n$ different resource nodes, with $k$ and $n$ large (hundred of millions) and $t$ fairly small (less than a thousand). For the first stage, the complexity is linear on the number of triples, and the only data are the $t$ reductions. In contrast, for the second stage the complexity is $n \times t$, both in time and memory. Thus, for larger semantic graphs, the complexity is an issue, particularly if the graph is too large to keep in memory.

The first stage of the mapping can be parallelized by splitting triples according to the predicate (edges type). This can achieve a considerable speedup on the first stage but does not reduce either the number of nodes or the number of types of edges, hence it has no impact on the second stage.

An obvious way to improve efficiency would be to avoid computing weights altogether, since this is the most time consuming task of the a-graph mapping. Of course, this would reduce the informative value of a-graphs, that would not highlight the most relevant path patterns.

An alternative to improve efficiency is sampling. Mapping a small enough subset (sample) of all the triples in the graph has a significant improvement in performance. Sampling is acceptable if it produces the same a-graph structure – a-nodes and a-edges – with a small error on their weights.

A naive approach to sampling would be the random selection of triples. However, a small (around 1%) random sample of triples typically has an impact on the structure of the a-graph, by not correctly identifying all the a-nodes (a-edges are not usually affected). A larger sample (around 50%) solves that problem but significantly reduces the efficiency.

The DBLP example presented in Subsection 5.3 shows that this approach can be used if a small sample of the triples is representative of the complete set. In this case, this was achieved by considering the subset of triples related to the publications of a few authors. Unfortunately, in most cases this is impossible, since triples in RDF files are usually either grouped by edge type or just unordered.

Both approaches, dropping weight evaluation and sampling, have disadvantages and cannot be used systematically. However, they may be more effective if combined. More precisely, instead of dropping weight evaluation, an approximation may be computed using sampling. In this approach, the first stage is processed exactly in the way described in Subsection 4.1 but the second stage is modified. Hence, the a-nodes and their weights are computed exactly as they are defined in Subsection 3.2. This means that a-node weights maintain as invariant that their sum is 1. The same is not true for the second stage, where a-edges are determined as pairs of a-nodes with nonnull weight.

Computing a-edges weights is the major contributor to the computational complexity of the second stage. This is due to the fact that, in general, a single node may contribute to the weight of several a-edges, as explained in Subsection 3.2. If these weights are computed approximately using sampling then this complexity may be curbed.

Relaxing the computation of a-edge weights may have an impact on the structure of the a-graph, since some low weight a-edges may be missed. Additionally, it will be difficult, if not impossible, to maintain invariant the sum of a-edge weights. Everything considered, it is preferable to risk missing the least relevant a-edges than any a-node, and the weights of a-edges are less important than those of a-nodes to the visualization and browsing of a-graph diagrams. In any event, although promising, this approach of relaxing the computation of a-edges is not yet available and will require further research.

## 7. Conclusions and future work

Semantic graphs are hard to visualize due to a large number of typed nodes and edges. The a-graph approach to abstract semantic graphs maps edge information into a-nodes and node information into a-edges. The abstraction mapping produces a smaller graph that is easier to visualize and highlights the patterns of paths in the original semantic graph.

A-nodes and a-edges are assigned with weights that reflect the relevance of the edges and nodes they represent, and that can be used for further abstractions. For instance, a-nodes with small weights, corresponding to types of edges that seldom occur in the semantic graph, can be omitted to unclutter large a-graph diagrams.

The a-graph diagram is the proposed graphical syntax to represent a-graphs, and thus visualize the semantic graphs. This kind of diagrams uses different shapes to represent a-nodes according to their types, and transparency to denote weights. The special a-nodes top and bottom are represented as parallel lines respectively on the top and bottom of the diagram. In an a-graph, diagram paths are in general upwards, which facilitates their detection.

The web application for visualizing and interacting with a-graphs is also an important contribution of this research. It uses a force-directed algorithm, which allows the incremental layout of the diagram after reposition or removal of a-nodes. This application can use data from different sources: local data entered on the interface, remote data available on the web and precomputed data for a few preprocessed semantic graphs.

The proposed approach still faces the challenge of dealing with massive semantic graphs with millions of triples, such as those of Yago and DBpedia. The major problem is due to the computational complexity involving a-edge weights. However, there are approaches to curb this complexity that are currently being researched. After tackling this issue, the a-graph browser will be easier to evaluate with real users interested in discovering path patterns in large semantic graphs.

**Acknowledgments.** I am in debt to the anonymous reviewers for their careful reading of this manuscript and their many insightful comments and suggestions.

This work is financed by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme, by National Funds through the FCT Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project POCI-01-0145-FEDER-006961, and by FourEyes. FourEyes is a Research Line within project TEC4Growth Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01- 0145-FEDER-000020 financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

## References

1. Abello, J., Van Ham, F., Krishnan, N.: Ask-graphview: A large scale graph visualization system, Visualization and Computer Graphics, IEEE Transactions on 12 (5), 669–676 (2006)
2. Archambault, D., Munzner, T., Auber, D.: Grouseflocks: Steerable exploration of graph hierarchy space, Visualization and Computer Graphics, IEEE Transactions on 14 (4), 900–913 (2008)
3. Auber, D.: Tulipa huge graph visualization framework, in: Graph Drawing Software, Springer, 105–126 (2004)
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data, Springer, (2007)
5. Bastian, M., Heymann, S., Jacomy, M., et al.: Gephi: an open source software for exploring and manipulating networks., International Conference on Web and Social Media - ICWSM 8, 361–362 (2009)
6. Benedetti, F., Po, L., Bergamaschi, S.: A visual summary for linked open data sources, in: International Semantic Web Conference (2014)
7. Bikakis, N., Liagouris, J., Kromida, M., Papastefanatos, G., Sellis, T.: Towards scalable visual exploration of very large RDF graphs, in: The Semantic Web: ESWC 2015 Satellite Events, Springer, 9–13 (2015)
8. Fellbaum, C.: WordNet, Wiley Online Library, (1999)
9. Guarino, N., Oberle, D., Staab, S.: What is an ontology?, in: Handbook on ontologies, Springer, 1–17 (2009)
10. Harispe, S., Ranwez, S., Janaqi, S., Montmain, J.: Semantic similarity from natural language and ontology analysis, Synthesis Lectures on Human Language Technologies 8 (1), 1–254 (2015)
11. Kobourov, S. G. , Spring embedders and force directed graph drawing algorithms, CoRR abs/1201.3011.
URL http://arxiv.org/abs/1201.3011
12. Harary, F.: Graph Theory, Massachusetts: Addison-Wesley (1972)

13. Hastrup, T., Cyganiak, R., Bojars, U.: Browsing linked data with fenfire., in: Linked Data on the Web (2008)
14. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: Relfinder: Revealing relationships in rdf knowledge bases, in: Semantic Multimedia, Springer, 182–187 (2009)
15. Hoffart, J.,Suchanek, F. M.,Berberich, K., Weikum,G.: Yago2: A spatially and temporally enhanced knowledge base from wikipedia, in: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, AAAI Press, 3161–3165 (2013)
16. Leal, J.P.: Path Patterns Visualization in Semantic Graphs, in: 7th Symposium on Languages, Applications and Technologies, SLATE 2018, OASIcs, vol. 62, 15:1–15:15 (2018)
17. Lin, Z., Cao, N., Tong, H., Wang, F., Kang, U., Chau, D. H.: Demonstrating interactive multiresolution large graph exploration, in: Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on, IEEE, 1097–1100 (2013)
18. Liu, Y. et al: Graph summarization methods and applications: A survey. ACM Computing Surveys (CSUR), 51.3: 62, (2018)
19. Song, Q. et al.: Mining summaries for knowledge graph search, in IEEE Transactions on Knowledge and Data Engineering 30, no. 10, 1887–1900 (2018)
20. Zhang, K., Wang, H., Tran, Yu, D. T., Y.: Zoomrdf: semantic fisheye zooming on rdf data, in: Proceedings of the 19th international conference on World wide web, ACM, 1329–1332 (2010)
21. Zinsmaier, M., Brandes, U., Deussen, O., Strobelt, H.: Interactive level-of-detail rendering of large graphs, Visualization and Computer Graphics, IEEE Transactions on 18 (12), 2486–2495 (2012)

**José Paulo Leal** graduated in mathematics from the Faculty of Sciences of the University of Porto and earned a Ph.D. in Computer Science from the same institution. His main research interests are technology enhanced learning, web adaptability, and semantic web.

*Received: July 17, 2019; Accepted: October 6, 2019.*

Computer Science and Information Systems 17(1):253–269     https://doi.org/10.2298/CSIS190427039E

# A Mobile Crowd Sensing Framework for Suspect Investigation: An Objectivity Analysis and De-Identification Approach*

ElAlaoui ElAbdallaoui Hasna[1,2], ElFazziki Abdelaziz[1,3], Ennaji Fatima Zohra[1,4], and Sadgal Mohamed[1,5]

[1] Computing Systems Engineering Laboratory (LISI)
[1] Faculty of Sciences Semlalia, Cadi Ayyad University, Marrakech, Morocco
[2] h.elalaoui@edu.uca.ac.ma
[3] elfazziki@uca.ma
[4] f.ennaji@edu.uca.ma
[5] sadgal@uca.ma

**Abstract.** The ubiquity of mobile devices and their advanced features have increased the use of crowdsourcing in many areas, such as the mobility in the smart cities. With the advent of high-quality sensors on smartphones, online communities can easily collect and share information. These information are of great importance for the institutions, which must analyze the facts by facilitating the data collecting on crimes and criminals, for example. This paper proposes an approach to develop a crowdsensing framework allowing a wider collaboration between the citizens and the authorities. In addition, this framework takes advantage of an objectivity analysis to ensure the participants’ credibility and the information reliability, as law enforcement is often affected by unreliable and poor quality data. In addition, the proposed framework ensures the protection of users' private data through a de-identification process. Experimental results show that the proposed framework is an interesting tool to improve the quality of crowdsensing information in a government context.

**Keywords:** crowdsourcing, crowdsensing, law enforcement, objectivity analysis, de-identification.

* It is an extended version of the (8th International Conference on Model and Data Engineering MEDI 2018).

## 1. Introduction

For a long time, everyone was asked to perform organizational tasks, to generate content or simply to collect useful information. Today, the access to the Internet has enabled human crowds to be online 24/7, which has spurred the use of human intelligence in different types of problem solving [1]. Over the past decade, crowdsourcing has emerged as a powerful paradigm for engaging the crowd in complex tasks [2] that even powerful computing machines cannot perform. While participants could perform their tasks on computers, they can now do some of them on their mobile devices. The latter, being equipped with several integrated sensors (GPS, for example), make it possible to collect more relevant data in a given context. This technique of data collecting allows the emergence of the crowdsensing concept, especially mobile crowdsensing [3]. In addition to playing a leading role in different applications, crowdsensing has also attracted considerable attention in organizational practice and many institutions and government organizations have incorporated it into their decision-making policies [4]. By exploiting the new opportunities offered by the new information and communication technologies, these agencies are making good efforts to strengthen citizen participation. Several examples testify this and prove that crowdsensing played a key role in solving many problems: the best-known example is the Boston Marathon attack on April 15th, 2013 [5]. Crowdsensing has rapidly become a form of collaboration that allows governments (local, state or federal) to use citizens' skills in order to gather or evaluate information about a situation or a context related to a given event (natural disasters, crimes, wars, etc.).

Despite the breadth of the various crowdsensing applications mentioned above, a challenge of how to manage all the information provided by the crowd and ensure or verify their reliability emerges and requires effective support. Recurring questions about the concept of the participants' credibility and the information reliability can be raised. The success of a participatory activity depends on its quality, especially in critical contexts such as witnessing crimes or identifying suspects/criminals.

To address this problem, the purpose of this paper is to encourage the public to participate in activities that fall within the remit of the public authorities; especially e-participation. Recently, several platforms such as CrimeReports, WikiCrimes or CrimeMapping [6] have emerged in several countries. These platforms report, on digital maps, information about the location of crimes or suspects. Such platforms will be of undeniable added value in developing countries such as Morocco. To do this, we propose an approach to develop a framework for managing citizen collaboration in government activities. This framework will be a good support for the authorities allowing them a rational data management on the crimes and the criminals and an acceleration in the process of suspects' identification and localization. So, this framework will support:

- The collecting of information on crimes and suspects through an e-participatory infrastructure.
- The information about crimes and suspects can be visualized thanks to online user interfaces.
- The verification of the participants' credibility and the reliability of the information they provide will be done using an objectivity analysis.

– The anonymity of the participants will be set up using a de-identification process.

The implementation will be carried out using a set of tools. First, the process of de-identification is based on a k-anonymization algorithm effectively masking the identifiers of a participant. Subsequently, the objectivity analysis will be based on a K-Means clustering that brings together location information of a suspect to be processed by a reliability validation algorithm of each cluster.

The rest of the paper is structured as follows. After a literature review gathering a set of works developed for suspect identification and localization, we detail the proposed framework by presenting its structure and its implementation. Experiments and results are presented in Section 6 before concluding the paper with a discussion and a future work section.

## 2. Related Research

In this section, we review the work previously done in the context of suspect investigations. First, we discuss law enforcement and the integration of the information technologies. Then, we present some work where the crowdsensing concept was adopted and we discuss how the latter has been able to strengthen the quality of suspect investigations. Finally, we present the main methods proposed for verifying the users' credibility and the reliability of the information that the e-participants provide.

### 2.1. Law enforcement and information technologies

The identification and the prosecution of criminals have changed considerably and many tools have been developed to help the authorities find the suspects. By analyzing the literature, we found a considerable number of works made for similar purposes. For example, Mali P et al. in [7] propose a system based on the analysis of the images captured by the CCTV cameras in order to find a correspondence between suspects and the criminals mugshots registered in the authorities' databases.

With the aim of promoting generic reports while automating the process of detecting, and synthesizing information about a crime, Asquith [8] has implemented new techniques for sharing criminal information through sophisticated sensors. This is supported by Artificial intelligence algorithms and Natural Language Processing (NLP).

In a similar view, M. I. Pramanik et al. [9] see the Big Data Analytics and its applications as a potential for effective resolution of complex issues, including criminal analysis. Most law enforcement and government agencies need to think about adopting Big Data analytics techniques like Data Mining to cope with the large volume of data from a variety of data sources. This will help to develop effective strategies to prevent crime and the formation of criminal networks.

We note that these works neglect or rarely involve citizens who remain important sources of information in such scenarios. Human potential is a pillar in the suspect identification process. In other words, the application of crowdsourcing/crowdsensing has become an increasingly common practice among government authorities who see in the collaboration with citizens, an important step to take into consideration in any criminal case. In this part of the paper, we present some previous work that used the crowdsensing concept for suspect identification and/or localization. We also summarize the main contributions and techniques proposed for verifying the information provided by the crowd as part of a crowdsensing-based activity.

## 2.2. Law enforcement and crowdsourcing

Transafe [10] is a system that allows citizens of the city of Melbourne, Australia to share location-based, time-stamped crime data, as well as their perceptions of security in a given city location. This data can be used by government agencies such as the public transit companies. The system also has a user tracking and emergency calls features. However, this platform, not yet evaluated, does not give details about the crimes and/or the criminals.

In a similar perspective, CrowdSafe [11] is a crowdsensing-based system for storing and displaying spatio-temporal data of criminal incidents. CrowdSafe includes other features such as a Safety Router that guides users through the least dangerous routes. Thanks to a dashboard, this system will enable better data analysis for smarter and safer public decision-making. The authors used real data from the Washington DC metropolitan area, but they did not propose any solutions to verify the reliability of the information shared on the platform.

On the other hand, Furtado et al. describe the WikiCrimes Web application [12], a multi-agent system that allows anyone to record and/or search for criminal information directly on maps. In WikiCrimes, the information is more credible if the user supports his statement with a document (link to a video, a newspaper article, a police report, etc.). In addition, the more people confirm a fact, the more it is reliable.

Following the same vision, Hairihan Tong proposed Bian Yi [6], a system dedicated to crime mapping allowing spatio-temporal visualization. It allows a user to mark on a Google Map, the location of a crime and all the underlying information (date, description, etc.).

Despite the importance of this platform (this has been proved through an online survey), this system has some limitations. For example, when a user submits a crime report, his contact information or credentials are not requested and remain optional. The accuracy and the authenticity of the data is still a problem. In addition, the scoring system (based on increasing or decreasing the reliability score of a report) is a feature granted only to witnesses who may not perform this task for fear that their identity will be disclosed.

## 2.3. User credibility and information reliability

With the explosion and diversity of the information sources, it is difficult to determine the veracity of any information given that it is done in an open and/or anonymous and not lucrative way, which is the case for participatory activities. However, this has repercussions on the quality of the decision-making process, which has led many researchers to propose some methods of Truth Discovery or Quality Assurance [13,14]. These two concepts aroused great interest in the field of participatory management.

Li Y, Gao J, Meng C, et al. [14] have classified three methods to explain the general principle of Truth Discovery: iterative methods, optimization-based methods and probabilistic graphical model based solutions (PGM). These methods and others have been successfully applied in crowdsourcing/crowdsensing applications to build mutual trust between the entities involved such [15] [16] and [17]. In the latter [17], the authors proposed a system based on fuzzy clustering to improve the quality of human computing in crowdsourcing applications. This system has been combined with a Trusted Access Control (TBAC) strategy to decide if a participant has access to a collaborative work or not.

## 3. The Framework overview

Taking into account the information received from the crowd is one of the major challenges in the e-participative activities. However, the ultimate objective is to find the simplest and the most appropriate way to do so while preserving the confidentiality of the information, especially in critical initiatives such as the reporting of crimes or the identification of suspects.

The purpose of this work will be to enable the authorities to develop, collect, analyze and interpret the data provided by the citizens about a crime situation. In this case, the use of an objectivity analysis turns out to be a necessity to take into account in the development of a crowdsensing-based application. This two-level analysis involves verifying the credibility of the participants, in addition to validating the veracity of the information they share. In order for the participants to demonstrate commitment and motivation, and not have this fear of disclosing their identities, a system will ensure the anonymization of their private information. The framework architecture and the request analysis and validation process will be presented below.

### 3.1. The architecture

The framework architecture consists of five components and their interactions are schematized in Fig. 1.

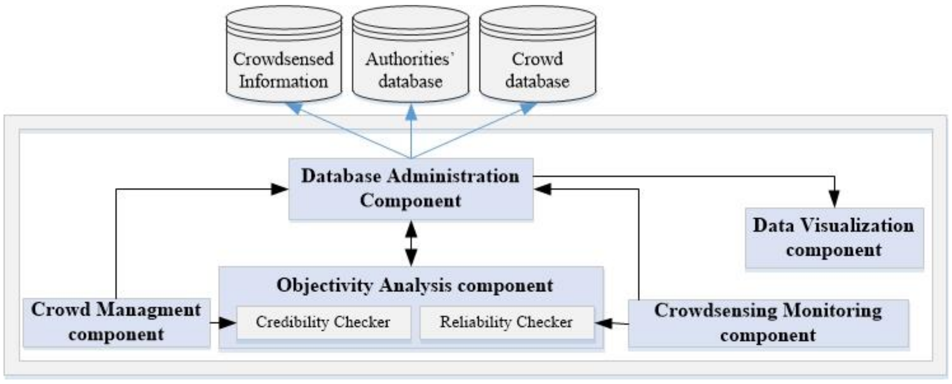

Crowdsensed Information
Authorities' database
Crowd database
Database Administration Component
Data Visualization component
Objectivity Analysis component
Credibility Checker
Reliability Checker
Crowd Managment component
Crowdsensing Monitoring component

**Fig. 1.** The framework architecture

The Database Administration Component is linked to all other components and three databases: 'Authorities database' where the profiles of criminals are stored, 'Crowd Database' which contains all the participant identification data and finally, 'Crowdsensed Information DB' which is reserved for storing all data collected by the participants. Given the heterogeneity of the data (structured and unstructured) to be stored, we adopt a document-oriented database management system and more specifically MongoDB [18].

Containing two sub-components, the objectivity analysis component supports the validation of the participants' credibility and the verification of the collected data reliability. It is linked to the Crowd Management component that manages the participants' registration and authentication, and to the CrowdSensing Monitoring component which manages the collected data such as crime details, suspect locations and visuals (photos, videos, etc.).

All information about crimes and/or suspects are retrieved by the Data Visualization component and presented on digital maps that can be viewed by the authorities or the public through online user interfaces.

## 3.2. Request analysis and validation process

First, we would like to point out that the proposed framework is generic and is suitable for any type of crime or criminal incident that can be witnessed by a citizen (theft, violence and/or armed threat, physical aggression, etc.). Fig. 2 illustrates the process to be followed by a participant (a citizen) from the issuance of his request to its approval.

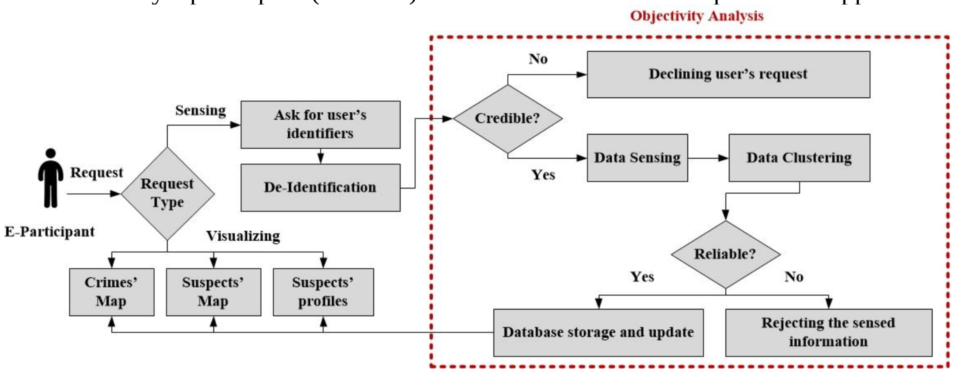


**Fig. 2.** The request analysis and validation

The participant information submission is carried out by an authentication to verify the user credibility. If the framework judges unfavorably his credibility, his request is automatically declined, if not, he can participate in the crime management by providing the information about a crime, the localization or the profiling of a suspect (image, video, etc.). The submitted information is subject to a verification before its approval.

## 4. The De-Identification Process

De-identification, also known as anonymizing data, is the process used to prevent a person's identity from being linked to information [19]. For example, data produced during research on human subjects could be de-identified to preserve the privacy of the participants. It is commonly used in the health field where de-identification is primarily focused on the protection of patient information [20,21].

The authorities have a database reserved for the personal crowdsensing information storage (national identifiers, address, etc.). This confidential information must be provided by a participant before the access enabling. Although the authorities can benefit from the relevant crowd information, they are wary of disclosing their details without a guarantee that they will not be intercepted. As a result, these information, known as Personally Identifiable Information (PII) is critical and vulnerable and needs to be de-identified. Fig. 3 below shows the process of de-identification followed after the registration or the authentication of a participant to allow him or not the access to the framework.

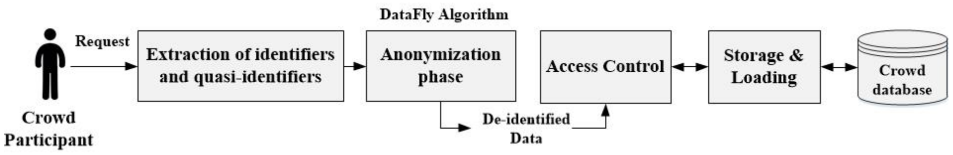


**Fig. 3**. The de-identification process

### 4.1. The extraction of identifiers and quasi-identifiers

To begin, we define the attributes that directly designate the identity of an individual (Identifiers) such as the name and the national identity card number and those who identify him indirectly (Quasi-Identifiers) such as ZIP code or the date of birth. We want to point out that a subset of some quasi-identifiers allows the identification of an individual. In other words, if the QIs are sufficiently well correlated, they can be combined to create a unique identifier. Table 1 below defines these attributes and their categories.

**Table 1**. Identifiers and Quasi-identifiers with their categories and PII

| | Category | PII |
|---|---|---|
| Identifiers | Name | Fist Name, Last Name, Surname |
| | IDs | National Identity Card Number, Social Security Number, Driver’s License Number, Passport Number |
| | Contact | Phone Number, Email Address |
| Quasi-Identifiers | Age | Birth Date (year, month, day) |
| | Gender | - |
| | Marital Status | Not Married (Single, Widowed, |

| | |
|---|---|
| | Divorced), Married |
| Contact | Country, City, Department, ZIP, Street Address |
| Job | Job, Institution or Organization |

## 4.2. The anonymization phase

After identifying and classifying PII, we apply two strategies: pseudonymization [19] to suppress (replacing with *) or hide (replacing with symbols) identifiers and make individuals de-identified. Also, we proceed to a k-anonymization process [22] and treat the data by ensuring that at least k individuals have the same combination of QI values through generalization and/or deletion techniques. This reduces the probability of disclosure risk by 1/k. For example, the k-anonymization replaces some original data in the records with new range values and retains some unchanged values. The new combination of QI values prevents the identification of the individual and avoids the destruction of the data records.

Several methods have been established for this purpose [23]. In this paper, we use the DataFly algorithm [23] known for its minimal execution time and information loss. The DataFly algorithm is detailed below. It counts the frequency on the QID set (the set of quasi-identifiers) and if the k-anonymization is not yet satisfied, it generalizes the attribute having the most distinct values until the k- anonymization is satisfied.

```
The Datafly Algorithm
Input: {T: the table with private data, k: the k-
anonymity constraint, QID ={Q1,Q2, …,Qn}: the n quasi-
identifiers, suppThreshold: a suppression threshold};

Output: GT: the generalized table

start
  Compute the frequency count (FreqC) of T using QID

  While (FreqC<= k) do

     // Verify if the table T is ready for suppression

          If (FreqC <= suppThreshold) do

             Suppress tuples with FreqC <= suppThreshold

          Else

          Search for attributes t with FreqC_t= max(FreqC)

          GT = Generalize table T using t

          End if
```

```
    Compute FreqC

End while

Return: GT
end.
```

An example of the de-identification achieved using the above-mentioned techniques is presented in the implementation section. Table 2 and Table 3 are respectively the private data table and the de-identified data table using some attributes.

**Table 2.** Identification Data Table

| | Identifiers | | | Quasi-Identifiers | | | |
|---|---|---|---|---|---|---|---|
| | Name | NIC | Phone | Age | Gender | Marital Status | Zip Code |
| 1 | Name1 | NIC1 | +212666666666 | 24 | M | Divorced | 10242 |
| 2 | Name2 | NIC2 | +212611111111 | 23 | F | Single | 10256 |
| 3 | Name3 | NIC3 | +212600000000 | 35 | M | Single | 10440 |

**Table 3**. De-Identified Data Table

| | Identifiers | | | Quasi-Identifiers | | | |
|---|---|---|---|---|---|---|---|
| | Name | NIC | Phone | Age | Gender | Marital Status | Zip Code |
| 1 | * | $*$ | +2126$$$$$$$$ | [20 :30) | M | Not Married | 102** |
| 2 | * | $*$ | +2126$$$$$$$$ | [20 :30) | F | Not Married | 102** |
| 3 | * | $*$ | +2126$$$$$$$$ | [30 :40) | M | Not Married | 104** |

## 5. The information reliability checking

In a previous work, we were interested in the suspect profiling data transfer, analysis and processing. However, in this paper, we detail the process of verifying the information provided by a crowd concerning a crime/suspect locations.

For this, a participant is required to mark on a map, the locations where he identified a crime or a suspect and the time of the identification. Thus, the information is not precise and can generate some problems in terms of precision. For example, two people may be in the same place at the same time but within a few meters. Therefore, instead of checking each location, we check the entire group of nearby locations. The first step, then, consists of grouping (clustering) these reported locations before applying the objectivity analysis.

## 5.1. Locations clustering

There are several clustering algorithms for partitioning the received location data into subgroups, or more formally into clusters. These “group” together similar observations (here localizations). To deal with this problem, we chose to use unsupervised learning methods, specifically the K-Means algorithm [24]. It aims at structuring all types of data into k groups in order to minimize a defined function. Fig. 4 illustrates this classification by showing the resulting clusters after applying the K-means algorithm.

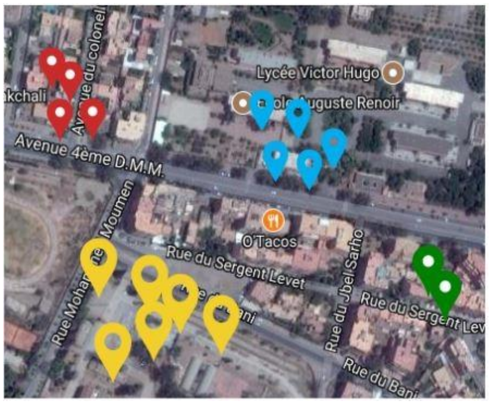


**Fig. 4.** An example of the clusters obtained after applying the K-Means algorithm

## 5.2. The objectivity analysis algorithm

The objectivity analysis proposed in this paper is based on a simple and probabilistic algorithm in order to identify the most reliable information among a crime/suspect locations reported at a given moment. By setting a moment T= t ± Δt, we define some parameters for reliability calculation as follows:

- S: the number of participants.
- L: set of information reported at the time T where L= {$l_0$, $l_1$… $l_n$} and li is the ith cluster of localizations at time T and n is the number of clusters.

```
The information reliability analysis algorithm
Input: { the information L= {$l_0$, $l_1$… $l_n$} and the number of
participants S };

Output: Identified reliable locations and their scores

Start.
For $l_0$ to $l_n$ do
```

```
 RS(l_i) = |S_li|/|S|   where ∑ Rs(l_i)= 1 and S_li is the
number of people who reported the location l_i

End For.

Return: the cluster l_i with the highest reliability score
RS
end.
```

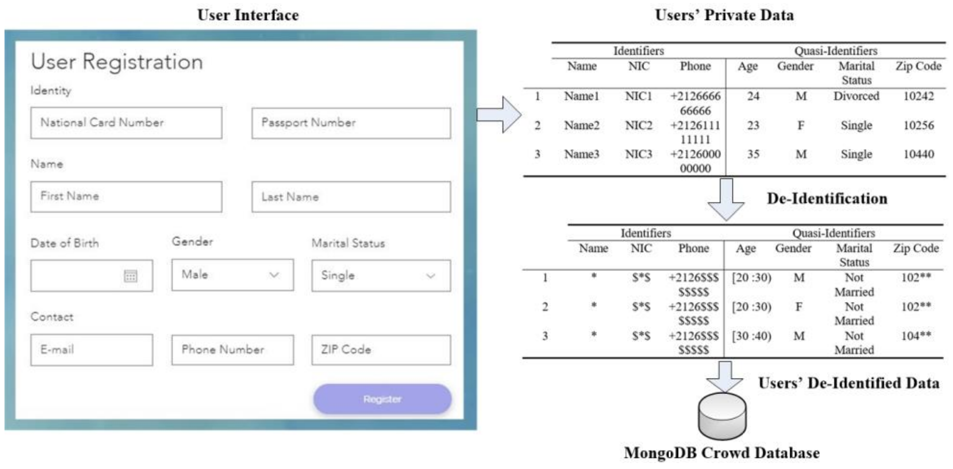
For each location $l_i$, the algorithm deduces the reliability score of the group $l_i$ as a function of the current estimation. At the end of n iterations, it returns the cluster with the highest score, which indicates the most reliable information.

## 6. The implementation

In this section, three main online user interfaces will be presented: user registration interface and two data collecting interfaces that enables the e-participants to report information about a crime or a suspect. These interfaces are adapted for both web and mobile. For each case, we also present the storage process and the databases structure.

### 6.1. User registration

Before allowing a participant to report any information, he must first register in the framework by indicating the information presented in Fig. 5. These information are anonymized using the de-identification process presented earlier before being stored in the crowd database.



| | Identifiers | | | Quasi-Identifiers | | | |
|---|---|---|---|---|---|---|---|
| | Name | NIC | Phone | Age | Gender | Marital Status | Zip Code |
| 1 | Name1 | NIC1 | +212666666666 | 24 | M | Divorced | 10242 |
| 2 | Name2 | NIC2 | +212611111111 | 23 | F | Single | 10256 |
| 3 | Name3 | NIC3 | +212600000000 | 35 | M | Single | 10440 |

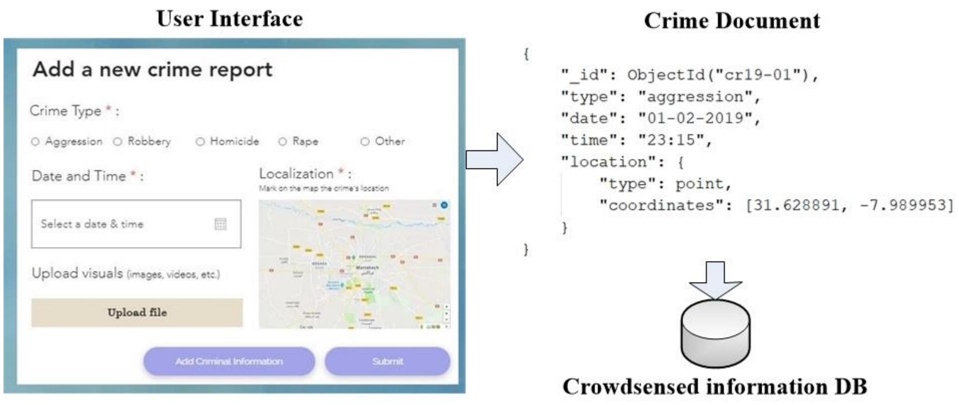
| | Identifiers | | | Quasi-Identifiers | | | |
|---|---|---|---|---|---|---|---|
| | Name | NIC | Phone | Age | Gender | Marital Status | Zip Code |
| 1 | * | $*$ | +2126$$$$$$$$ | [20 :30) | M | Not Married | 102** |
| 2 | * | $*$ | +2126$$$$$$$$ | [20 :30) | F | Not Married | 102** |
| 3 | * | $*$ | +2126$$$$$$$$ | [30 :40) | M | Not Married | 104** |

**Fig. 5.** User registration interface

### 6.2. Data collecting

The e-participants validated by the access control module are authorized to enquire information about a crime or a suspect to enable the authorities to progress in their decision-making process. These information are then structured to be suitable for storage in a document-oriented database such as MongoDB. In this section, we present the implemented user interfaces as well as the structure of the documents stored in a MongoDB database.

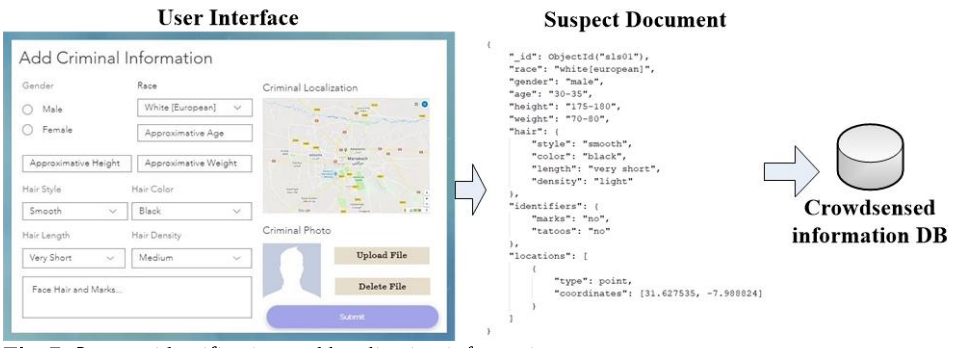
**Crime description.** Fig. **6** illustrates the web interface for loading the information about a crime and the structure of the crime document (JSON file) that is stored in the ‘Crowdsensed Information Database’ (MongoDB).



```
{
    "_id": ObjectId("cr19-01"),
    "type": "aggression",
    "date": "01-02-2019",
    "time": "23:15",
    "location": {
        "type": point,
        "coordinates": [31.628891, -7.989953]
    }
}
```

**Fig. 6.** Crime reporting

**Suspect identification and localization.** By clicking on the 'Add Suspect Information' button of the crime report form, a participant can add information to assist in identifiying or locating a suspect. Fig. 7 presents the form to which participants are redirected to share this data. This latter is structured in a JSON file before being stored in the MongoDB ‘Crowdsensed Information Database’.



```
{
    "_id": ObjectId("sls01"),
    "race": "white[european]",
    "gender": "male",
    "age": "30-35",
    "height": "175-180",
    "weight": "70-80",
    "hair": {
        "style": "smooth",
        "color": "black",
        "length": "very short",
        "density": "light"
    },
    "identifiers": {
        "marks": "no",
        "tatoos": "no"
    },
    "locations": [
        {
            "type": point,
            "coordinates": [31.627535, -7.988824]
        }
    ]
}
```

**Fig. 7.** Suspect identification and localization information

## 7. Case Study, Results and Discussion

A simulation of the proposed framework process was performed in an earlier work using the Anylogic simulator[1]. The simulation [25] helped to test the proposed framework in a virtual environment and to get a dataset to apply the objectivity analysis.

In this part, we present the results obtained after the processing and the analysis of the information reported by the e-participants. First, we will expose the results of the objectivity analysis (verification of the reliability of the reported locations). Then, the retained locations are used to generate a spatio-temporal map where all the reliable localizations are marked. We are interested in the case of suspects but this is also valid for the information on the crimes. Finally, we discuss the overall contributions of the proposed framework and its effectiveness.

### 7.1. Objectivity analysis results

The objectivity analysis and more particularly the information reliability make possible the data refinements for a better result of the suspect identifying process. Fig. 8 below is a diagram of two Matlab graphs plotting on the left all the information of a suspect location reported by the crowd and on the right those filtered and considered reliable by the OA module.

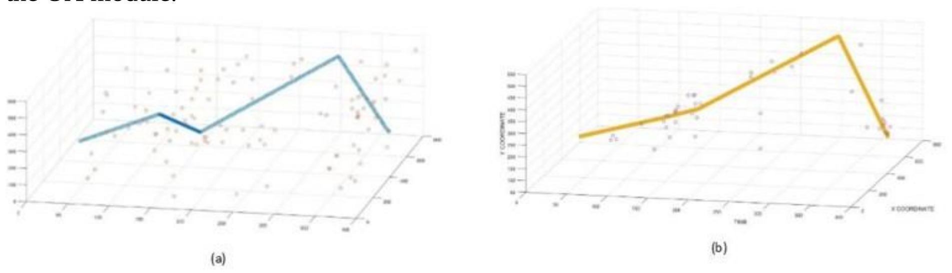


**Fig. 8.** Suspect localizations (a) before and (b) after applying the objectivity analysis

[1] www.anylogic.com

### 7.2. Suspect spatio-temporal map

Since the location information of a suspect are reported by the e-participants, the framework only retains the most reliable information (checked by the objectivity analysis component) to generate a digital map viewable only by the police officers (Fig. 9). This map allows tracking the movements of a suspect.

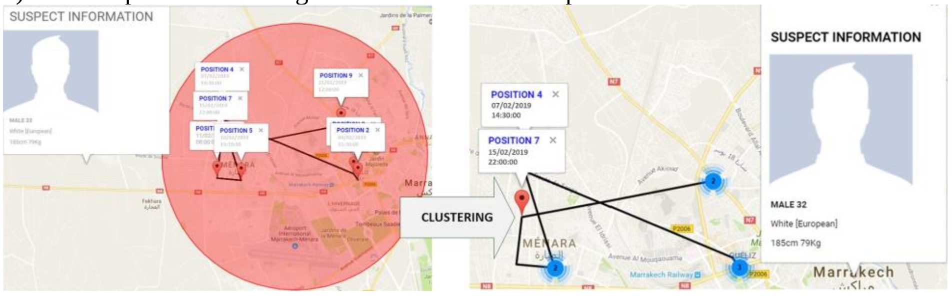


**Fig. 9.** Suspect movements spatio-temporal map

### 7.3. Discussion

The proposed framework differs from the systems discussed in the literature review in that it is based on both new technologies and the concept of crowdsensing. This combination is important for the acceleration of the investigation process. In addition, the proposed framework is not only intented for reporting crimes but also in identifying and tracking potential suspects. It also incorporates 3 important contributions: the verification of the e-participants' credibility before granting them the access to the different interfaces, the verification of the information reliability through an objectivity analysis and a de-identification process of users' private data which must encourage them to collaborate.

The experimental results obtained prove the effectiveness of the proposed framework since it allowed a significant reduction of 37.7% of the data that could hinder the investigation process or slow it down. Also, the generated spatio-temporal map is a good visual and support for the decision-makers to identify the risky areas (the dispersion of the crimes in a city) and follow the movements of a suspect.

The major limitations of this research work can be summarized in two major points: the lack of real data to test and validate the framework in question. Morocco doesn't currently make available to the public, data on crimes and/or suspects. Also, and as mentioned in the section 'Conclusion and Future Work', an integration of a reliable and secure reward system is a necessity. This will significantly increase the participants' motivation.

## 8. Conclusion and Future Work

The low rate of crime reporting in the cities to the authorities and the reduced number of the authorities' officials allocated to this task is a major impediment to the smooth process of identifying and locating suspects. Therefore, adopting the concept of crowdsourcing for the information collecting and sharing between the citizens and the authorities can alleviate this problem. Also, the anonymization of e-participants can encourage them to collaborate. On the other hand, incorporating an objectivity analysis into the process can make it more relevant.

The implementation of this complex and spatiotemporal process by means of datamining tools and the storage of information in a document-oriented database (MongoDB) make it possible to have an appropriate infrastructure for taking spatiotemporal information into account relating to the identification and location of suspects.

In order to make this infrastructure more confidential and secure, we are considering the integration of the blockchain concept as future work.

## References

1. Lease M, Alonso O. Crowdsourcing and Human Computation, Introduction [Internet]. In: *Encyclopedia of Social Network Analysis and Mining (ESNAM)*. Springer, 304–315 (2014). Available from: https://www.ischool.utexas.edu/~ml/papers/lease-esnam14.pdf.
2. Howe BJ. The Rise of Crowdsourcing. *Wired Mag.* 14(6), 1–4 (2006).
3. Guo B, Wang Z, Yu Z, *et al.* Mobile Crowd Sensing and Computing: The Review of an Emerging Human-Powered Sensing Paradigm. *ACM Comput. Surv.* [Internet]. 48(1), 1–31 (2015). Available from: http://dl.acm.org/citation.cfm?doid=2808687.2794400.
4. Cupido K, Ophoff J. A Model of Fundamental Components for an e-Government Crowdsourcing Platform. *Electron. J. e-Government*. 12(2), 141–156 (2014).
5. Brabham DC, Ribisl KM, Kirchner TR, Bernhardt JM. Crowdsourcing applications for public health. *Am. J. Prev. Med.* [Internet]. 46(2), 179–187 (2014). Available from: https://doi.org/10.1016/j.amepre.2013.10.016.
6. Tong H. A crowdsourcing based crime mapping system. (2014).
7. Mali P, Rahane V, Maskar S, Kumbhar A, Wankhade S V. Criminal Tracking System using CCTV. *Imp. J. Interdiscip. Res.* [Internet]. 2(7), 2454–1362 (2016). Available from: http://www.onlinejournal.in.
8. Asquith B james. Crime Intelligence 2.0: Reinforcing Crowdsourcing using Artificial Intelligence and Mobile Computing [Internet]. (2017). Available from: https://cloudfront.escholarship.org/dist/prd/content/qt39s3k7bw/qt39s3k7bw.pdf.
9. Pramanik MI, Lau RYK, Yue WT, Ye Y, Li C. Big data analytics for security and criminal investigations. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. (2017).
10. Hamilton M, Salim F, Cheng E, Choy SL. Transafe: A crowdsourced mobile platform for crime and safety perception management. *Int. Symp. Technol. Soc. Proc.* 2015-July (2015).
11. Shah S, Bao F, Lu C-T, Chen I-R. Crowdsafe: Crowd Sourcing of Crime Incidents and Safe Routing on Mobile Devices. In: *ACM SIGSPATIAL GIS'11.* , 521–524 (2012).
12. de Oliveira M, D'Orleans J, Caminha C, *et al.* Collective intelligence in law enforcement – The WikiCrimes system. *Inf. Sci. (Ny).* [Internet]. 180(1), 4–17 (2009). Available from: http://dx.doi.org/10.1016/j.ins.2009.08.004.
13. Pouryazdan M, Kantarci B, Soyata T, Song H. Anchor-Assisted and Vote-Based Trustworthiness Assurance in Smart City Crowdsensing. *IEEE Access*. 4, 529–541 (2016).
14. Li Y, Gao J, Meng C, *et al.* A Survey on Truth Discovery. *ACM SigKdd Explor. Newsl.* [Internet]. 17(2), 1–16 (2016). Available from: https://doi.org/10.1145/2897350.2897352.
15. Xu G, Li H, Tan C, Liu D, Dai Y, Yang K. Achieving efficient and privacy-preserving truth discovery in crowd sensing systems. *Comput. Secur.* [Internet]. 69, 114–126 (2017). Available from: https://doi.org/10.1016/j.cose.2016.11.014.

16. Huang C, Wang D, Chawla N. Towards time-sensitive truth discovery in social sensing applications. *Proc. - 2015 IEEE 12th Int. Conf. Mob. Ad Hoc Sens. Syst. MASS 2015.* , 154–162 (2015).
17. Folorunso O, Mustapha OA. A fuzzy expert system to Trust-Based Access Control in crowdsourcing environments. *Appl. Comput. Informatics* [Internet]. 11(2), 116–129 (2015). Available from: https://doi.org/10.1016/j.aci.2014.07.001.
18. Sowmya R, Suneetha K R. Data Mining with Big Data [Internet]. In: *2017 11th International Conference on Intelligent Systems and Control (ISCO).* , 246–250 (2017). Available from: http://ieeexplore.ieee.org/document/7855990/.
19. Khalil M, Ebner M. De-Identification in Learning Analytics. *J. Learn. Anal.* 3(1), 129–138 (2016).
20. Dernoncourt F, Lee JY, Uzuner O, Szolovits P. De-identification of patient notes with recurrent neural networks. *J. Am. Med. Informatics Assoc.* 24(3), 596–606 (2017).
21. Stubbs A, Kotfila C, Uzuner Ö. Automated systems for the de-identification of longitudinal clinical narratives: Overview of 2014 i2b2/UTHealth shared task Track 1. *J. Biomed. Inform.* 58, S11–S19 (2015).
22. Patil D, Mohapatra RK, Babu KS. Evaluation of generalization based K-anonymization algorithms. *Proc. 2017 3rd IEEE Int. Conf. Sensing, Signal Process. Secur. ICSSS 2017.* , 171–175 (2017).
23. Ayala-Rivera V, McDonagh P, Cerqueus T, Murphy L. A Systematic comparison and evaluation of k-Anonymization algorithms for practitioners. *Trans. Data Priv.* 7(3), 337–370 (2014).
24. Arora P, Deepali, Varshney S. Analysis of K-Means and K-Medoids Algorithm for Big Data. In: *Physics Procedia*. (2016).
25. El Alaoui El Abdallaoui H, El Fazziki A, Ennaji FZ, Sadgal M. A gamification and objectivity based approach to improve users motivation in mobile crowd sensing. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* (2018).

**ELALAOUI ELABDALLAOUI Hasna:** Is a computer science engineer, graduated from the National School of Applied Sciences Marrakesh/Morocco year 2014. After acquiring scientific and technical knowledge in the computer and information systems field; especially in designing, modeling and implementing software solutions from both the architectural and administrative perspectives, she then integrated the Computing Systems Engineering Laboratory of CADI AYYAD University since January 2016 to prepare her PhD thesis. Her research interests include e-government applications, crowdsourcing, image processing, mobile applications, etc.

**ELFAZZIKI Abdelaziz:** Received the M.S. degree from the University of Nancy, France, in 1985, and the Ph.D. degree in computer science from CADI AYYAD University in 2002. He has been with CADI AYYAD University since 1985, where he is currently a Professor of computer science. He has been responsible for the master’s degree program in information system engineering since 2006. He was the Director of the Computer Systems Engineering Laboratory between 2011 and 2015. He has co-authored several papers on agent-based image processing, and is the main Author of over 20 papers in software engineering and data analytics field. His research interests are related to software engineering, decision support, big data, data analytics, crowdsourcing, and e-government. In the MDA field, he has been involved in agent-based systems, service-oriented systems, and decision support systems.

**ENNAJI Fatima Zohra:** Is a computer science engineer, graduated from the National School of Applied Sciences Marrakesh/Morocco at 2014 and finished her PhD in 2019. She joined in 2015 the Computing Systems Engineering Laboratory of CADI AYYAD University. Her thesis includes many research interests like social media, sentiment analysis, data mining, Big Data Analytics, crowdsourcing, social CRM, etc. She participated in many international conferences and published many articles in international journals.

**SADGAL Mohamed:** is professor of computer science at Cadi Ayyad University, Morocco, and researcher on computer vision with the Vision team at the LISI Laboratory. His research interests include object recognition, image understanding, video analysis, multi-agent architectures for vision systems, 3D modelling, virtual an augmented reality, among other topics. Before Marrakech, he was in Lyon (France), working as Engineer in different computer Departments between 1988 and 1994. He obtained a PhD in 1989 from Claude Bernard University, Lyon, France.

*Received: April 27, 2018; Accepted: September 12, 2019*

Computer Science and Information Systems 17(1):271–292 https://doi.org/10.2298/CSIS190430040E

# Verification and Testing of Safety-Critical Airborne Systems: a Model-based Methodology

Mounia Elqortobi[1], Warda El-Khouly[1], Amine Rahj[1], Jamal Bentahar[1] and Rachida Dssouli[1]

[1] Concordia University, Quebec, Canada
m_elqort@ mail.concordia.ca,
warda_elkholy@yahoo.com
{amine.rahj, jamal.bentahar, rachida.dssouli}@ concordia.ca

**Abstract.** In this paper, we address the issues of safety-critical software verification and testing that are key requirements for achieving DO-178C and DO-331 regulatory compliance for airborne systems. Formal verification and testing are considered two different activities within airborne standards and they belong to two different levels in the avionics software development cycle. The objective is to integrate model-based verification and model-based testing within a single framework and to capture the benefits of their cross-fertilization. This is achieved by proposing a new methodology for the verification and testing of parallel communicating agents based on formal models. In this work, properties are extracted from requirements and formally verified at the design level, while the verified properties are propagated to the implementation level and checked via testing. The contributions of this paper are a methodology that integrates verification and testing, formal verification of some safety critical software properties, and a testing method for Modified Condition/Decision Coverage (MC/DC). The results of formal verification and testing can be used as evidence for avionics software certification.

**Keywords:** Model-based Verification, Model Checking, Communication Graph, Methodology, Model-based Testing, Partial Reachability Graph, MC/DC (Modified Condition/Decision Coverage).

Sponsored by NSERC/CRD CMC CS Canada. Project CRIAQ AVIO 604, CRDPJ 463076-14.

## 1. Introduction

Developing safety-critical software requires rigorous processes. To prevent catastrophic events, the avionics industry has introduced a rigorous certification process, described in the RTCA [1, 2] standard. The DO-178C standard [1] includes a supplement on formal methods called DO-333. In the DO-333 standard, a formal method is defined as "a formal model combined with a formal analysis". The DO-178C and its supplement have been successfully applied into the production of software systems at Dassault-Aviation and Airbus [3]. The motivation of this work is to increase software dependability by integrating formal verification techniques with testing and to capture the benefits of their cross-fertilization. In addition, formal verification and test results can be used as evidence for certification. Although model-based testing [5, 6] and verification activities [3, 4, 5] are natural approaches to the certification of avionics software, the integrated model-based engineering approach is not yet well studied in the literature, and several challenges still need to be addressed [4, 7, 12, 14].

We propose a model-driven approach that encompasses two main levels: verification/design and validation/implementation. As shown in figure 1, in the first level, we adopt model checking, a formal and fully automatic technique for model-based verification. It is a natural choice for a rigorous verification of avionics systems against desirable properties, including safety and liveness. In the second level, we transform the finite state machine (FSM) verification model [9, 10, 11] into an Extended Finite State Machine (EFSM) testing model that is an FSM-like model extended with variables [16]. We generate both local test cases for each EFSM component modeled as agent in its context of communication, and global test cases for a Communicating EFSM (CEFSM) model. The CEFSM is a composition of EFSMs. The test generation method satisfies the Modified Condition/Decision Coverage (MC/DC) criterion, all Definition-Use (DU)-paths, and ensure that the verified properties hold in the implementation. The selection of coverage criteria is based on the satisfaction of DO 178C for MC/DC and on the use of middle ground structural coverage for all DU-paths. Using a better structural coverage criterion, such as all-paths, is often impractical.

Model–based verification and model-based testing are still very active research domains [5, 6, 14, 16, 23, 24, 25]. They are considered as two distinct research areas and supported by different research communities. EFSM-based testing and Communicating EFSM have been extensively studied [18, 19, 20, 21, 23, 25]. A more recent research area is test generation based on model checking, with several publications [17, 22, 26, 27, 28, 29] discussing that topic. The principle is basically to generate counterexamples or witness traces that can be used to derive test cases. The major problems in all the published work are related to performance, the notion of test coverage or test efficiency, non-determinism, and the abstraction level of test cases, derived from counterexamples and witness traces, that need more refinement to be accurate and be utilized to test implementations [17, 29]. To the authors' best knowledge, there is no work on the methodologies that link testing and verification in the same framework.

The rest of the paper is organized as follows. In Section 2, we present an overview of the proposed approach and our case study about a landing gear system [11]. This is followed by a summary of our model-based verification system, formal modeling of our case study, and our experimental results verifying the correctness of the modeled landing gear system against desirable properties including safety and liveness in Section 3. We then present our model-based testing and show how to automatically generate test cases in Section 4. In Section 5, we cover the details of MC/DC criterion and how to integrate non executable paths. We offer our discussion, conclusions, and identify future work in Section 6.

## 2. The Proposed Methodology and Case Study

### 2.1. The Proposed Methodology

We introduce the proposed verification and testing framework in this sub-section. The methodology begins with formally modeling the safety-critical airborne system from the given informal requirement specifications, producing an FSM-like model as described in Figure 1. We assume that a correct informal specification exists. Next, it proceeds to refine and encode the obtained model using ISPL+ (an extended version of the input language of the symbolic model checker MCMAS+ introduced in [9]) to verify agent-based intelligent systems.

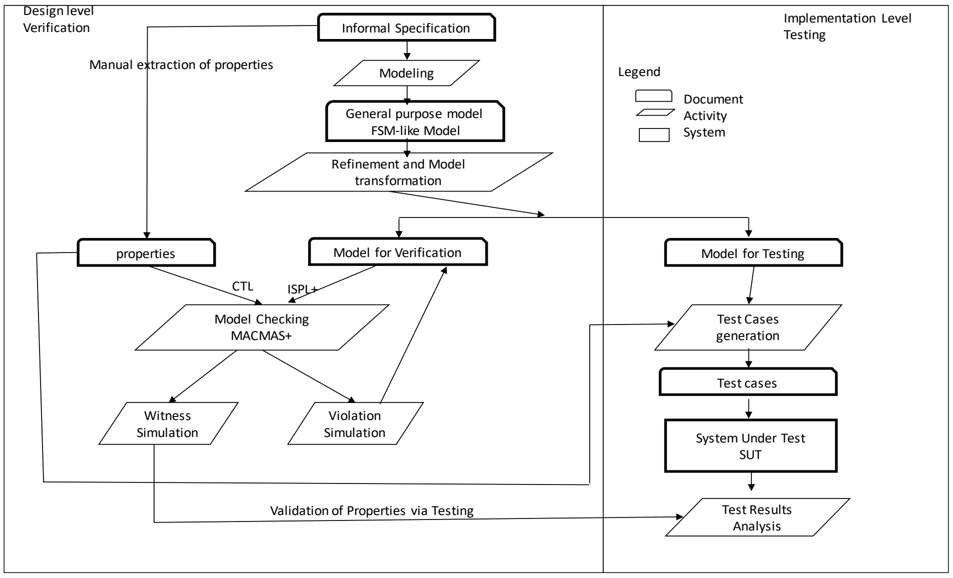


**Fig 1.** Overview of our Approach

Parallel with this step, our approach extracts and expresses the system requirements in the form of temporal properties using Computation Tree Logic (CTL) [8]. MCMAS+ automatically checks whether the model satisfies the intended properties and graphically produces witness-examples or counter-examples [12, 13]. The produced witness-examples prove the satisfaction of properties while the produced counter-examples guide designers to detect and repair design errors in the formal system model. In the validation/implementation level, our approach automatically transforms the formal models into a reduced Communicating Extended Finite State Machine (CEFSM) that uses our developed algorithms and tools to automatically generate abstract test cases. These algorithms and tools address the conformity of the implementation under test to Low-Level Requirements (LLR), instead of to high-level requirements as in existing automated test generation techniques; thereby allowing them to be more applicable and efficient for the satisfaction of avionics standards. After assigning values to the required data sets, the generated test cases are transformed into concrete ones with respect to the expressed properties. The concrete test cases are then applied to the implementation under test. The Modified Condition/Decision Coverage (MC/DC) criterion is integrated into the test generation algorithm to satisfy the requirements of the DO-178C [1, 30, 31]. Finally, our approach analyzes the obtained test results and compares them with the produced witness-examples to validate our properties via testing.

## 2.2. Case study: Landing Gear System

Our case study, a landing gear system for an aircraft, was proposed by Frédéric Boniol and Virginie Wiels in [11] as a representative scenario for complex industrial needs. The case study is very rich as it is not restricted to software and includes complex system modeling. The landing system is responsible for maneuvering landing gears and attached doors. It consists of three landing packages situated in the front, right, and left part of the aircraft. Each landing package includes a door, a landing-gear and related hydraulic cylinders. A door can be open or closed, while the gear can be retracted, extended, or maneuvered. The landing system can be controlled by a software package and can be in two modes: normal or emergency. In outgoing and retraction situations, the normal mode is the default. The emergency mode is deployed to handle failure situations. This work only considers the outgoing sequence and its normal and emergency modes. The architecture of the system consists of three parts (see Figure 2): 1) a pilot part; 2) a mechanical part that incorporates the mechanical devices and three landing packages; and 3) a digital part that includes the control unit software.

Regarding the pilot part, a pilot has a button switch at her/ his disposal with two positions: UP or DOWN. When the button switch goes from UP to DOWN, the outgoing sequence is initialized. The pilot has three lights in the cockpit that reflect the current status of the gears and doors. These lights are as follows:

- One green light, indicates that “gears are locked down”;
- One orange light, indicates that “gears are maneuvering”; and
- One red light, indicates a “landing gear system failure”.

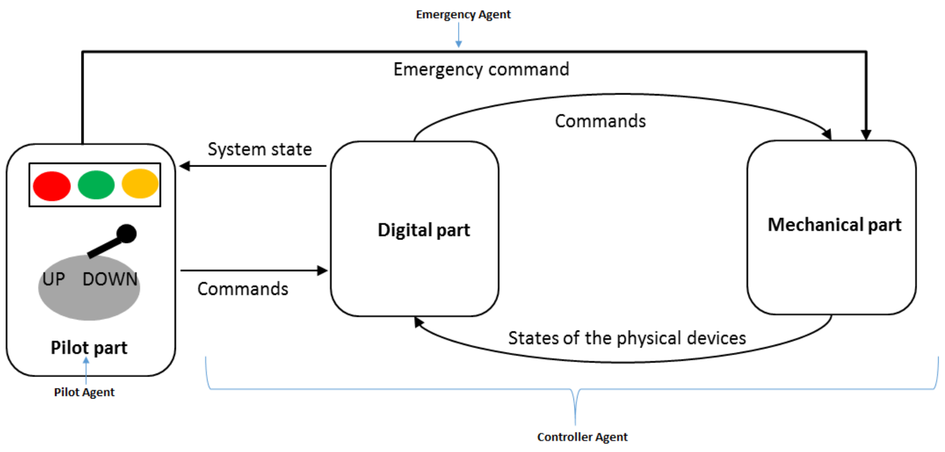


**Fig 2.** General Architecture of the Landing Gear System

Before initializing the outgoing sequence, all the landing gears are locked in their up position and all the lights are off. In case of failure (i.e., the red light is on), the pilot manually pulls the mechanical handle to deploy the emergency hydraulic system. The expected consequence of this deployment is to lock the gears in the down location. When all gears are successfully extended and all accompanying sensors are valid, the green light must be lit. Regarding the mechanical part, the motion of landing gears and doors is performed by a set of hydraulic cylinders such that the cylinder position basically corresponds to the door or landing gear location. For example, when a door is open, the corresponding cylinder is extended. The hydraulic power of these cylinders is supplied by a set of electro-valves. The digital part is in charge of sending an electrical order to activate each electro-valve. Notably, the three doors (and their gears) are controlled in parallel by the same electro-valve. The digital part plays an intermediate role between the pilot part and the mechanical part. Specifically, the software embedded in the digital part is responsible for controlling the gears and doors, detecting anomalies, and informing the pilot about the status of the system through a set of lights. It also generates commands directed to the hydraulic system to open or close the doors and extend or retract the gears with respect to the values of employed sensors and captures the pilot orders.

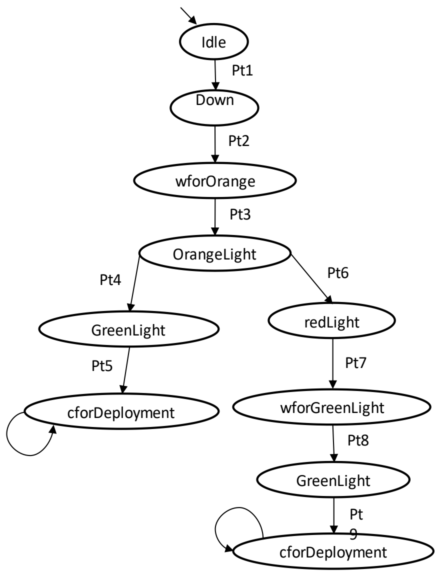
## 3. Model-based Verification

### 3.1. Modeling the Landing Gear System

In this section, we show how our model M can formally model the landing gear system. In our modeling, we specifically consider the normal and emergency modes of the landing gear system without going into low-level details regarding the mechanical devices of sensors and electro-valves. To achieve this aim, we introduce three agent machine models: $M_p$ for pilot, $M_c$ for control unit, and $M_e$ for emergency. The pilot agent machine model $M_p$ models the behavior of the pilot part and the control unit agent machine model $M_c$ models the behavior of the digital part. The emergency agent machine model $M_e$ models the behavior of the emergency system. Instead of adding another agent machine to model the behavior of the hydraulic cylinders, we depend on the status of doors and gears to directly represent the status of the employed cylinders. This is because the description above states that the doors' cylinders are extended when the doors are open, and a similar relation holds between gears and their cylinders.

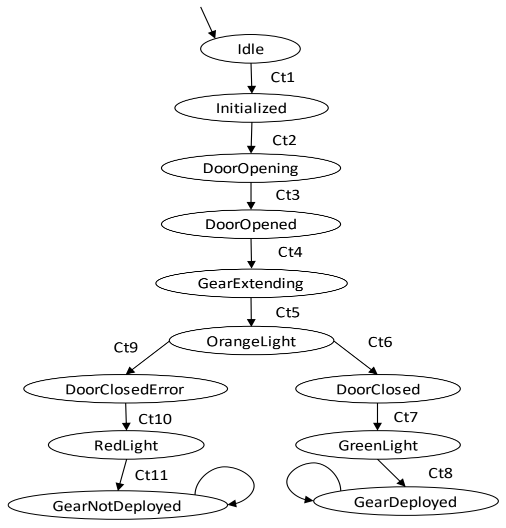
In the published case study paper, there are two types of requirements and the authors classify them as strong and weak. The weak requirements that did not consider deadline constraints/time constraints. Although we selected the weak requirements, the time constraints are abstractly represented in our model where each transition takes one-time unit as in all standard abstracted temporal models.

Figures 3, 4, and 5 show the EFSM models of the pilot, control unit, and emergency agent machines, respectively. In each figure, we introduce the input and output of each transition in a tabular form where the symbols "?" and "!" refer to the process of receiving and sending an action. The output of a transition can be directly assigned by the shared and unshared variables when there is no explicit output action. Given that, it is easy to define the Boolean predicate of each transition using the conjunction operator between its input and its output.



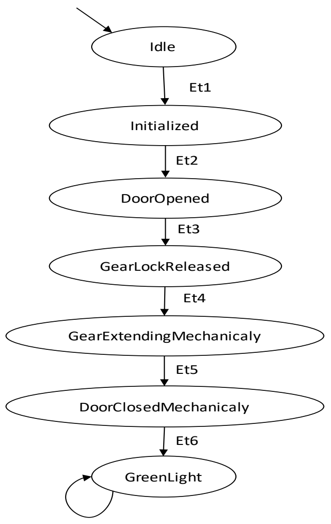
| Label | Input, output & predicate |
|---|---|
| Pt1 | ? LandingSpecs(speed, distance)<br>! PressDownButton |
| Pt2 | ? PressDownButtonAck<br>! wforOangeLight |
| Pt3 | ? OrangeLightOn<br>! OrangeLightOnAck |
| Pt4 | ? GreenLightOn<br>! GreenLightOnAck |
| Pt5 | ? ConfirmGearDeployment<br>! DeploymentStatusSccess |
| Pt5 | ? RedLightOn<br>! RedLightOnAck |
| Pt7 | ? ConfirmGearDeploymentError<br>! InitializeEmergencySystem |
| Pt8 | ? GreenLightOn$M_e$<br>! GreenLightOn$M_e$Ack |
| Pt9 | ? ConfirmGearDeployment$M_e$<br>! DeploymentStatusSuccess |

**Fig 3.** Pilot Agent Machine model, $M_p$



| Label | Input, Output & Predicate |
|---|---|
| Ct1 | ? PressDownButton<br>! PressDownButtonAck |
| Ct2 | ? ProcessReceivedCommand<br>! OpenGearDoors |
| Ct3 | ? OpenGearDoorsAck<br>! OutgoingGears |
| Ct4 | ? OutgoingGearsAck<br>! OrangeLightOn |
| Ct5 | ? OrangeLightOnAck<br>! CloseGearDoors |
| Ct6 | ? CloseGearDoorsAck<br>! DoorsCloseSuccess |
| Ct7 | ? GearsExtended<br>! GreenLightOn |
| Ct8 | ? GreenLightOnAck<br>! ControlUnitDisconnected |
| Ct9 | ? CloseGearDoorsError<br>! DoorsCloseError |
| Ct10 | ? GearsNotExtended<br>! RedLightOn |
| Ct11 | ? RedLightOnAck<br>! ControlUnitDisconnedted |

**Fig 4.** Controller Agent Machine model, $M_c$



| **Label** | **Input, Output & Predicate** |
|---|---|
| Et1 | ? InitializeEmergencySystem<br>! OpenGearDoors |
| Et2 | ? OpenGearDoorsAck<br>! OutgoingGears |
| Et3 | ? OutgoingGearsAck<br>! VerifyGearsPosition |
| Et4 | ? VerifyGearsPositionAck<br>! LockDoorsMechanicaly |
| Et5 | ? LockDoorsMecanicalyAck<br>! GreenLightOn |
| Et6 | ? GreenLightOnAck<br>! GearStatusExtended |

**Fig 5.** Emergency Agent Machine model, $M_e$

## 3.2. Validating

To perform the verification, we introduce the MCMAS tool. This is a symbolic model checker that extends MCMAS, a model checker for Multi-Agent Systems (MAS) that uses Ordered Binary Decision Diagrams (OBDD) [12, 13]. MCMAS takes two inputs: a model description for the system to be verified and a set of properties specified by different logics such as CTL and CTLC [9, 10]. The inputs of MACMAS are formatted by the ISPL language which is used to describe the communicating MAS to be checked and encode the desired specifications. The ISPL+ is a dedicated programming language for interpreted systems that formalize MASs (Fagin & Halpern, 1994). MCMAS+ automatically evaluates the truth value of the encoded specifications and produces counterexamples that can be analyzed graphically for false specifications. MCMAS can also provide witness executions for the satisfied specifications and graphical interactive simulations. For clarity, we introduce the syntax of CTL that is given by the following grammar rules:

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid EX\phi \mid EG\phi \mid E(\phi\ U\ \phi)$$ where:

1) $p \in Ap$ (the set of atomic propositions) is an atomic proposition and $E$ is the existential quantifier on paths.
2) $X, G$, and $U$ are temporal operators standing for “next”, “globally”, and “until”, respectively.
3) The Boolean operators $\neg$ and $\vee$ are defined and used in the usual way.

To validate our model M (a composition of $M_p$, $M_c$, and $M_e$) we need to perform the review and tracing activities. As a first validation activity, we must review the model with the wide range of features implemented in the MCMAS+ graphical user interface [10]. This graphical interface specifically highlights syntax errors, automatically displays content, and assists and supports text marking and formatting. After fixing all the highlighted errors, we have a clear and error-free encoding model. Tracing the activity allows us to track the behavior of the encoded model. The MCMAS+ tool offers an Explicit Interactive Mode. This tool starts with the initial state and offers all the transitions available at this state and gives the possibility to choose the transitions. After we select one of these transitions, the tool moves to the reachable state connected with the initial state by this transition and then displays the available transitions at the new state. This step allows us to evaluate whether the model is progressing as we intended. If an error is detected, we return to our encoding and update it. This process continues until we reach the end state. Then, we start again from the initial state and select another transition. Our graphical interface supports a new feature, which displays the whole model. By completing these two activities, we ensure that our encoding model exactly captures the intended behavior of the landing gear system. In fact, these two activities are key to ensuring that the model is correct; otherwise, errors in the design model could jeopardize the entire activity of the design formal verification using a model checking technique.

## 3.3. Model checking

According to the model checking technique, we must formally: 1) model the system underlying the verification process; and 2) express the requirements. The correctness of these requirements has been proven on the modeled system using MCMAS+. We have just shown how we complete the first activity. For the second activity, we used the Computation Tree Logic (CTL) [8] supported by the MCMAS+ model checker tool [12] to express the following requirements:

$$\begin{aligned}
\phi_1 &= AG(PressedDown) \rightarrow AF(GearsExtended \wedge DoorsClosed)) \\
\phi_2 &= \mathrm{EG}(\mathrm{E}(\mathrm{PressedDown}\ \mathrm{U}\ \mathrm{PressedDown} \wedge \mathrm{GearsExtended} \\
&\qquad\qquad \wedge DoorsClosed)) \\
\phi_3 &= AF\ \neg\mathrm{E}(\neg\ \mathrm{PressedDown}\ \mathrm{U}\ (\mathrm{GearsExtended} \wedge DoorsClosed)) \\
\phi_4 &= AG\ \neg(\mathrm{PressedDown} \wedge AG\ (\neg GreenLight)) \\
\phi_5 &= AF(GreenLight)
\end{aligned}$$

In [11], a set of requirements is presented with respect to the normal mode. The requirement called R11bis states that "when the command line is working (normal mode), if the landing gear command handle has been pushed DOWN and stays DOWN, then eventually the gears will be locked down and the doors will be seen closed". We expressed this requirement in the three different CTL formulae$\phi_1$, $\phi_2$ and$\phi_3$.

The first formula ($\phi_1$) can be read as follows: along all computation paths through all states, when the button is pressed down, then along all computation paths in the future the gears will be extended and the doors will be closed. The second formula ($\phi_2$) can be read as follows: there exists a computation path such that in all its states the gears will be not extended, and the doors will be not closed until the button is pressed down. The third formula ($\phi_3$) can be read as follows: along all computation paths in the future, the gears will be not extended, and the doors will be not closed if the button has never been pressed down before. The CTL formula $\phi_4$ expresses the safety requirement, which plays an important role in avoiding a bad situation. This bad situation in the fourth formula can be read as follows: the button has been pressed down and along all paths; the green light is never lit. The last CTL formula $\phi_5$ expresses the liveness requirement and can be read as follows: along all computation paths, the green light can be eventually lit. The quantifier ranging over all computation paths ("A") enables us to check the status of both normal and emergency modes. For example, the liveness formula allows us to check the status of the good thing ('green light') that will happen eventually in each mode. All these formulas are evaluated to true on the model M using MCMAS+. Therefore, our design model is error-free and it is strong, as it achieves the safety and liveness requirements required in both modes. We can also report some statistical results, such as that the execution time of verifying these formulas is 0.298 seconds and the memory consumed is 6 Megabytes.

## 4. Model-based Test Generation Approach

The goal is to generate, starting from the verification model, a set of test cases for the verified properties, apply them to the implementation under test and to then analyze the test results. The main idea is to demonstrate that the verified properties are properly propagated from the design level to the implementation level, and that they hold true within the Implementation Under Test (IUT). This demonstration requires model transformation, local and global test sequence generation, testing and test results' analysis. The approach both verifies the properties at the design level and demonstrates their validity at the implementation level using global test sequences, allowing the satisfaction of DO 178C by generating local test sequences with the required coverage criteria. In addition, we extend the set of paths to include additional paths to satisfy MC/DC.

## 4.1. Model Transformation

In model checking, a simple FSM is often used. Testing can use richer modeling techniques such as Extended Finite State Machines (EFSM). To use our test case generation techniques and tools with well-defined coverage criterion such as MC/DC [30, 31], we transform the verification model into a testing model. The notion of shared variables used in our verification model can be transformed into input parameters in the EFSM model. The interaction mode considered here is message passing. The discussion to use one model or two can take place. The solution for avoiding the use of a single model is to manually extract one model for verification and one model for testing. In this case, two different quality assurance groups should be involved, and the two models should cover the same set of requirements to satisfy the need for independency between verification and testing activities. The model transformation can show the equivalence between two models in this case.

## 4.2. Global Test Sequence Generation

Several approaches to generate global test sequences are based on or are otherwise similar to the work of Bourhfir and Cavalli [18, 19, 20, 23, 32, and 33]. We propose a test generation technique for parallel communicating agents. The generation of test sequences starts with the verification model. We first model each agent in its context and then create a list of transitions for the communication between a pair of agents. We use a transition-marking algorithm that marks every transition involved in the communication as an EFSM, along with its context. This technique generates local test sequences for each agent. Next, we compose the obtained EFSMs to build a global system M that is in fact a Communicating Extended Finite State Machines (CEFSMs) (see Figure 8).

## 4.3. Test Generation Process for the Case Study

In this case study and for the sake of readability, the EFSMs are only a partial representation of a landing gear system.

Following the DO-178C standards, the satisfaction of the MC/DC criterion is mandatory, and it is used as a criterion in this paper for test sequence generation. The MC/DC is a widely used and known coverage criterion in software avionics [30, 31].

The figure 6 describes the test generation process. To generate global test sequences, we first derive the local test sequences for each EFSM. Second, we obtain the communication graph from all EFSMs (see figure 7). Third, guided by the communication graph, we obtain the global system, or the CEFSM. Finally, from the local test sequences and the CEFSM, we generate the global test sequences. The following sections will detail the different steps of the test generation for the case study.

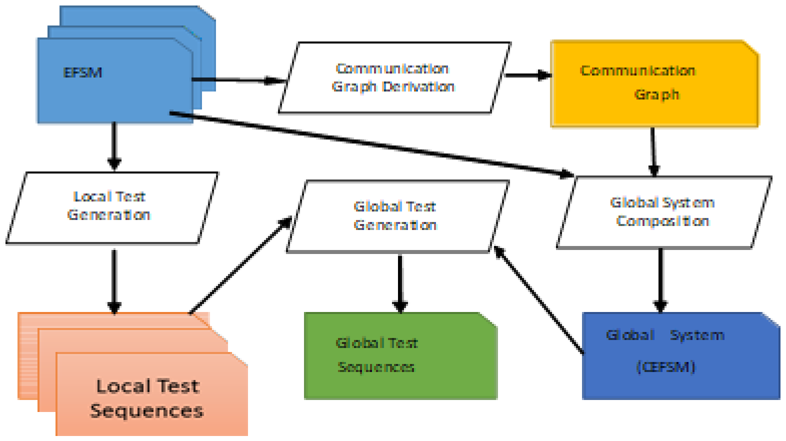


**Fig 6.** Test Generation Process

**Communication Graph**. To generate global test sequences for a global system composed of several agents, we need to abstract an EFSM agent into an abstract state and identify the communication transitions and their parameters that are used for communication. The communication graph represents the interaction between the different EFSMs (see Figure 7). For our case study, it is assumed that the communication between the machines $M_p$, $M_c$ and $M_e$ is two-way. Figure 7 visualizes the communication graph with the representation of each machine by an abstract state.

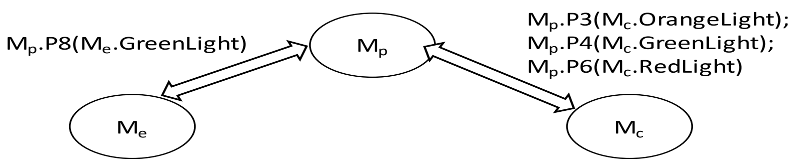


**Fig 7.** Communication graph representation

**Global Model with Communication Points.** Using the EFSMs (Figures 3, 4, and 5) and guided by the communication graph (Figure 7), we obtain (by composition) the global system model with its communication points (Figure 8). Figure 8 represents the composite system model M with its communication points, labels and transitions, and the input and output lists. We can see that the $M_c$ and $M_p$ agents start at the same time. It is in fact a parallel communicating system. The transitions representing the communication among agents are shown in orange, green, and red to represent the landing gear system lights of the same color. Similarly, a computation graph is also a composition of its constituents.

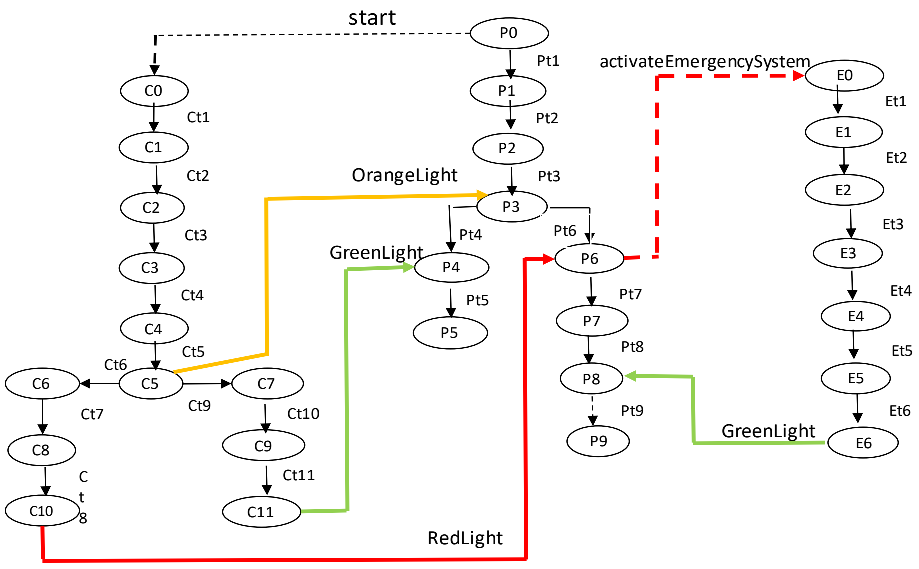


**Fig 6.** System model M: Composed of $M_p$, $M_c$, and $M_e$ with communication points

**Algorithm.** In this section, we briefly describe the test generation algorithm and its application to the case study. More specifically, we extend our algorithm to generate the test sequences that satisfy the MC/DC criterion. To generate executable test sequences, we need the final model with all the aforementioned information, the local test sequences, as well as the communication graph. The algorithm given in [18 and 31], called the generation of def-use executables, defines four different variable usages: assignment-use (A-usage), input-use (I-usage), computational-use (C-usage), and predicate-use (P-usage). These variable usages enable the links between the test sequences of each machine and help check the test sequences' executability. The algorithm provides a full set of executable and non-executable test sequences that will go through all the possible transitions existing in the system under test. We generate the paths linking two states from different machines by marking them as communication or synchronization points.

**The Generated Test Sequences.** To generate the test sequences, we first need to identify the communication variables. In the case of landing gear system, the variables are:
{Start, activateEmergencySystem, OrangeLight(on,off), GreenLight(on,off), RedLight(on,off) }
These variables indicate the possible communication between the agents. For example, if activeEmergencySystem is on, it means that the RedLight variable is also on. This is the only time the emergency system will be called upon. To identify the communication points, the input and output list for each transition is defined. The related input and output lists, as well as the predicates, are described in Figures 3, 4, and 5. They are used as inputs for the algorithm to generate the global test sequence. In general, a test case is composed of the following elements: <preamble, target, postamble>. Preamble and

Postamble might be empty. The preamble is the sequence of transitions used to reach the target transition for testing as given in Table 1.

Table 1 shows examples of the application of the algorithm using the landing gear case study. It identifies the different usage lists enabling the identification of executable test sequences. Table 2 shows an example of executable test sequences to reach specific transitions in the system model. The chosen transitions represent a case of parallelism.

**Table 1.** Example of usage lists and preamble for specific transitions of the landing gear system

| **Trans.** | **A-usage** | **I-usage** | **P-usage** |
|---|---|---|---|
| Pt2 | OrangeLight | - | - |
| Pt4 | - | GreenLight Ct11 | OreenLight on |
| Ct11 | - | GreenLight on ; OrangeLight off | GreenLight on |
| **Trans** | **Preamble** | | |
| Pt2 | Pt1 | | |
| Pt4 | Pt1, Pt2, Pt3, [Ct11] | | |
| Ct11 | Ct1, Ct2, Ct3, Ct4, Ct5, Ct9, Ct10 | | |

**Table 2.** Executable test sequences of the landing gear system

| **Transition** | **Executable test sequence** |
|---|---|
| Pt5 | Pt1, Pt2, Pt3, Ct1, Ct2, Ct3, Ct4, Ct5, Ct9, Ct10, Pt4, **Pt5** |

Table 3 presents an example of non-executable test sequences. These are non-executable because they need a preamble execution from another agent to reach the desired transition and render the sequence executable. Table 4 shows the parallelism in the executable test sequences required to make the transitions shown in Table 3 executable.

**Table 3.** Non-executable test sequence of the landing gear system

| **Transition** | **Non-executable test sequences** |
|---|---|
| Pt5 | Pt1, Pt2, Pt3, Pt4, **Pt5** |

**Table 4.** Parallelism shown for executable test sequences Pt5 of the landing gear system

| | **Executable test sequences – Pt5** | | |
|---|---|---|---|
| Mp | Pt1, Pt2, Pt3 | | P4, **Pt5** |
| Mc | Ct1, Ct2, Ct3, Ct4, Ct5 | Ct9, Ct10 | |

In the following sections, we will verify the different properties obtained from the validation phase.

### 4.4. Witness Properties' Verification

Table 5 shows specific executable test sequences for a selection of witness properties for liveness. Due to a limitation in all model checker tools in terms of generating witness-examples and counter-examples that include the universal operator "A", we used other formulas that achieve the same requirement and allow MCMAS+ to generate witness-examples.

The executable test sequences are given by the input and output information, as well as by the transitions for which that input and output information proved the witness-example to be true.

The executable test sequences represent the transition in which the witness-example holds. Hence, these are all the possible transitions forming a path needed to render a test sequence executable, up to the mentioned transition. For example, EF GreenLight holds true when a sequence executes up to transition Pt5 (refer to Table 4 for the complete executable test sequence).

**Table 5.** Executable test sequences for witness-examples for liveness properties

| Witness-example for liveness properties | Executable test sequences |
|---|---|
| EF GreenLight | Sequences leading to transitions:<br>$M_p$: Pt4 – Pt5 – Pt8;<br>$M_c$: Ct10 – Ct11<br>$M_e$: Et5 – Et6 |
| EF ( RedLight && EF GreenLight ) | Sequences leading to transitions:<br>$M_p$: Pt8 – Pt9;<br>$M_c$: none;<br>$M_e$: Et5 – Et6 |
| EF (PressedDown && EF GreenLight ) | Sequences leading to transitions:<br>$M_p$: Pt4 – Pt5 – Pt8 – Pt9<br>$M_c$: Ct10 – Ct11;<br>$M_e$: Et5 – Et6 |

### 4.5. Properties' Verification

Several properties are defined in Table 6 to verify whether the used algorithm validates the properties. The two executable test sequences shown in Table 2 were analyzed with regards to those properties. Both executable test sequences for transitions Pt5 and Pt9 verify all the properties identified so far. Table 6 confirms that all the global test sequences generated render the defined properties true.

According the algorithm used in [18], none of the executable test sequences validate the given properties. However, those that represent the full paths in the global system do validate them, being the paths generated for transitions Pt5 and Pt9. This implies that through that algorithm, only a set of test sequences can validate the different properties, and not necessarily all of them.

**Table 6.** LGS properties validated with the executable test sequences

| CTL | Status |
|---|---|
| $\phi_1 = AG(PressedDown) \rightarrow AF(GearsExtended \wedge DoorsClosed))$ | True |
| $\phi_2 = \mathrm{EG}(\mathrm{E}(\mathrm{PressedDown\ U\ PressedDown} \wedge \mathrm{GearsExtended} \wedge DoorsClosed))$ | True |
| $\phi_3 = AF\ \neg\mathrm{E}(\neg\ \mathrm{PressedDown\ U}\ (\mathrm{GearsExtended} \wedge DoorsClosed))$ | True |
| $\phi_4 = AG\ \neg(\mathrm{PressedDown} \wedge AG\ (\neg GreenLight))$ | True |
| $\phi_5 = AF(GreenLight)$ | True |

## 5. MC/DC

In order to comply with the avionics standard DO-178C, the proposed test generation algorithm needs to satisfy the modified condition/decision coverage (MC/DC) criterion. This will ensure that all possible conditions are tested. Therefore, we use a graph expansion mechanism to handle this type of coverage.

### 5.1. Handling MC/DC criterion

MC/DC is applied using binary values, and every condition will have a value of true or false. It is probable that some MC/DC test cases are not feasible within the system [31]. This means that some test cases' execution will fail.

The following requirements should be satisfied in MC/DC-based testing. For all decisions, at least once: 1) all possible outcomes are covered; 2) all possible outcomes for all conditions are covered; and 3) all conditions impacting the decision's outcomes are covered [30, 31].

In other words, all the outcomes of every decision, as well as the conditions within those decisions, should be executed at least once. By doing so, all paths regarding possible values taken by the system under test will be executed. For example, in the global system, a single decision must be made at P3 to move further to P4 or P6 as follows:

```
If (OrangeLight is on and GreenLight is off and RedLight is off )
        Return light status (RedLight or GreenLight on) from the controller;
    EndIf;
```

To satisfy the MC/DC criterion, we need to visualize a path as binary decisions and conditions. The algorithm will analyze a path with all possible conditions as binary as follows:

```
Decision → go to controller
  Conditions
      → if (OrangeLight is on/off)
      → if (GreenLight is on/off)
         → if (RedLight is on/off)
```

Another representation would be that the executable paths consist of all green transitions, and the non-executable ones are all red. The additional non-executable paths that will ultimately generate errors are partially red. For example, a path consisting of the values orange on / green off / red off will be part of a feasible path. However, going to the next state is impossible if within a path the values are orange on/green off/red on. Those additional paths exist to satisfy the MC/DC criterion. The objective is to render all the paths by considering the binary possibilities for each condition found in a decision, based on whether the orange light is on or off. However, the two other lights should be taken into consideration for conformity.

There are three conditions to consider within this decision: whether the OrangeLight is on, the GreenLight is off, and the RedLight is off. This translates to the following possibilities shown in Table 7, in which true and false are on and off, respectively:

**Table 7.** Possible binary values and possible outputs

| OrangeLight | GreenLight | RedLight | Output |
|---|---|---|---|
| True | False | False | Go to controller |
| False | True | False | Error |
| False | False | True | Error |
| True | True | False | Error |
| True | False | True | Error |
| False | True | True | Error |
| True | True | True | Error |
| False | False | False | Idle |

There is a value in executing test sequences from MC/DC criterion that result in an error, as it ensures that a test sequence will fail. As such, we also cover the possible of faulty signals being sent to the pilot, the controller, and the emergency agent. The errors are the result of a status or a state that is not naturally feasible by the system. To generate test sequences for MC/DC criterion [30, 31], we need to identify a way to consider the binary sequence and condense it into one single segment. This will enable the generation of MC/DC test sequences using model-checking. For example, we could add information in the input and output values for transition Pt3 by adding the different possibilities covered through MC/DC criterion and use that information to generate the required test sequences.

## 5.2. Test Generation Algorithm Satisfying MC/DC Criterion

The proposed test generation algorithm generates feasible test sequences. To satisfy the MC/DC criterion, the test generation algorithm must be modified and all of the decision branches need to be tested. For each binary decision, two paths will be generated for each simple condition involved in that decision. To integrate this coverage criterion, we need to pin-point in the algorithm the parts necessary to identify all DU-paths. For each element in the preamble list, we add a binary set of possibilities to satisfy the MC/DC criterion. This binary set will represent the possibilities for each information influencing a decision [30, 31].

There are several ways to approach this issue, for example: 1) create a standalone procedure executed at the end, that will have access to all the paths created initially, and generate additional ones to satisfy MC/DC criterion; 2) integrate MC/DC coverage while the paths are being generated to cover all feasible and non-executable paths related to a decision; or 3) analyze the non-executable paths and choose the ones satisfying the MC/DC criterion, using a hybrid approach based on approach number 2. The third approach is the one we selected. The algorithm that generates local test sequences is sketched as follows:

(1) Transform the EFSM to data flow graph G using graph rewriting.
(2) Expand the graph by an expansion mechanism; use state decomposition and graph splitting to handle MC/DC coverage criterion
(3) Select input values for each input parameter that can affect the control flow.
(4) Generate executable DU-paths according to data flow graph G and remove redundant paths. Append the state identification sequence and post amble (return to the initial state) to each DU-path to form a complete test path.
(5) Check test path executability; if non-executable, use cycle analysis to make it executable, discard if non-executable. This is done during the generation of a path.
(6) Verify if there are uncovered transitions, add test paths to cover them.

Handling MC/DC criterion in the Extended FSM Test Generation algorithm is explained in the following five steps.

**Step 1**: Define a second variable of binary values called vMCDC. This variable will take the values that will conform to the coverage's criterion. This variable will be used solely for MC/DC criterion satisfaction for test case generation.

**Step 2**: In the test generation algorithm, add all possible values for the identified input parameters that satisfy the MC/DC criterion and that are not already covered by the algorithm in its original state. Next, call a procedure that will analyze the discarded paths to ensure that they would not be involved in any MC/DC. Step (4) is used to analyze non-executable paths.

**Algorithm EFTG (Extended FSM Test Generation)**

(1) Read an EFSM specification;
(2) Generate the dataflow graph G from the EFSM specification;
(3) Choose a value for each input parameter influencing the control flow, augment the scope to consider the possible values for MC/DC;
(4) If the path is still non-executable, conduct the Analyze-discarded-path(P) procedure.

**Step 3**: Create procedure Analyze-discarded-path(P). This procedure will use the binary variable vMCDC and evaluate the information of path P to determine if it should be removed or not.

**Analyze-discarded-path(P)**

(1) Define binary values table with accepted values for green-orange-red states;
(2) For each variable, in every transition in the discarded path, compare the values with the binary table for green-orange-red-gear;
(3) If the values conform to the table, discard the non-executable path;

(4) If they do not conform, add this path to the MC/DC list of conformances (use the same logic for executable paths and flag them for MC/DC satisfiability).

**Step 4**: In the procedure executable-DU-path-generation, we add another loop to take into consideration vMCDC to identify the paths between transitions.

**Procedure Executable-DU-Path-Generation (flowgraph G)**

```
(1) Take in the MC/DC variables from the vMCDC variables;
(2) Generate all possible paths (call to Find-All-Paths (T,U, vMCDC)
    for each variable that has an A-use in T, and each transition U that has a P-
    use or a C-use.
```

**Step 5**: we replace the procedure handle-executability in order to not discard non-executable paths and call it procedure handle-executability-MCDC. If a path is non executable, it will not be removed. This is rather complicated as the algorithm is sound in making sure that all non-executable paths are confirmed twice as non-executable, and are then discarded. Another possibility is to add a condition that allows us to identify from which variable a path has been defined. If it was from a vMCDC variable, then we will not remove the non-executable path. Satisfying MC/DC criterion will result in adding several non-executable paths. This step is needed to ensure that erroneous paths are handled correctly, which will control both the satisfaction of the properties and the alternatives triggered by glitches or possible malfunctions.

## 6. Discussion and conclusion

Business case studies play a fundamental role in the progress and development of formal methods and help prospective users and designers demonstrate the application of different formal methods to model, verify, and test concrete, complex systems. In addition, they help to compare different formal techniques in terms of performance and ease of use. Relevant proposals have been put forward to model and formally analyze the landing gear system, a complex real-life case study published in [11]. Specifically, these proposals have suggested:

- Formal modeling methods including the Event-B methods [34, 35, 36, 37, 42], abstract state machines (ASM) [38, 43], and the Fiacre formal language [39, 40];
- Verification techniques including a proof theory [34, 35, 37, 43] and model checking [35, 39, 41, 40, 43]; and
- A test case generation technique [41], a run-time monitoring approach [40, 41], and a simulation technique [44].

These proposals provide only a partial solution with a unique objective, either modeling, verification, or testing. Moreover, these proposals do not consider all the industrial requirements of the case study. For example, a formal verification using model checking is used mainly to verify properties at the design level; the verified properties may not be propagated to the implementation stage. Therefore, testing of these properties is still needed. Although model-based testing and verification activities, as shown in DO-178C and DO-333, are natural approaches to the certification of avionics software, combining formal verification and testing in a single framework is still in its infancy and needs further investigation.

The proposed methodology and its application show that the integration of software quality assurance activities is needed to achieve software certification in the airborne industry. There are more challenges to overcome to be able to automate all the activities of the methodology. The avionics software has several types of inputs such as data from various actuators, and high volume of outputs. Both data input selection and trace analysis constitute real challenges and need more research and innovation to address them properly. Some hybrid modeling of the diversity of data input is needed. The oracle problem needs more data mining and intelligence for analyzing and for correlating outputs and searches in artifacts, such as requirement specifications, logs and test architectures. More efficient algorithms will also advance work in this field.

**Acknowledgement.** We would like to thank the reviewers and Dr. M. El Menshawy for their useful comments. This work was funded in part by NSERC/CRD CMC CS Canada and Project CRIAQ AVIO 604, CRDPJ 463076-14.

## References

[1] http://www.rtca.org. RTCA/DO-178C (2011) "Software Considerations in Airborne Systems and Equipment Certification", December 13, DO-332 Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A, DO-331 Model-Based Development and Verification Supplement to DO-178C and DO-278A, DO-333 Formal Methods Supplement to DO-178C and DO-278A

[2] Zoughbi, G., Briand, L., Labiche, Y.: Modeling safety and airworthiness (RTCA DO-178B) information: conceptual model and UML profile, Journal of Software & Systems Modeling, Volume 10, Issue 3, pp. 337-367, 2011

[3] Moy, M., Ledinot, E., Delseny, H., Wiels, V., Monate, B.: Testing or Formal Verification: DO-178C Alternatives and Industrial Experience, Journal of IEEE Software, Volume 30, Issue 3, pp. 50-57, 2013

[4] John Rushby, “New Challenges in Certification for Aircraft Software”, Proceedings of the 9th ACM International Conference on Embedded Software, pp. 211-218, 2011, www.csl.sri.com/users/rushby/papers/emsoft11.pdf

[5] Peleska, J., Siegel, M.: Test Automation of Safety-Critical Reactive Systems. South African Computer Jounal 19, pp. 53-77. (1997):

[6] Jan Peleska (2013): Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. MBT 2013: 3-28

[7] Gotzhein, R., Khendek, F.: Compositional Testing of Communication Systems. LNCS 3964, pp. 227 – 244 (2006) Gotzhein, R., Khendek, F.: Compositional Testing of Communication Systems. LNCS 3964, pp. 227 – 244 (2006)

[8] Clarke, E., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Massachusetts (1999)

[9] El-Kholy, W., Bentahar, J., El-Menshawy, M., Qu, H., Dssouli, R.: Conditional commitments: Reasoning and model checking. ACM Trans. on Soft. Eng. and Metho. 24(2), 9:1–9:49 (2014)

[10] El-Kholy, W., El-Menshawy, M., Bentahar, J., Qu, H., Dssouli, R.: Formal specification and automatic verification of conditional commitments. IEEE Intelligent Systems 30(2), 36–44 (2015)

[11] Boniol, F., Wiels, V.: The landing gear system case study. In: Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D. (eds.) Proceeding of 4th International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z.

Communications in Computer and Information Science, vol. 433, pp. 1–18. Springer (2014)

[12] Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A model checker for the verification of multiagent systems. CAV. LNCS, vol. 5643, pp. 682–688. Springer (2009)

[13] Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A model checker for the verification of multiagent systems. J. International Journal on Software Tools for Technology Transfer, 2017, Vol.19, N°1

[14] Berrada, I., Castanet R., Felix P.: Testing Communicating Systems: a Model, a Methodology, and a Tool. LNCS 3502, pp. 111–128 (2005)

[15] Kalaji, A.S., Hierons, R.M., and Swift, S.: Generating feasible transition paths for testing from an extended finite state machine(EFSM), International Conference on Software Testing Verification and Validation, ICST, pp.230–239, 2009.

[16] Clarke, E., Wing, J.: Formal Methods: State of the Art and Future Directions. ACM Computing Surveys, Vol. 28, No. 4 (1996)

[17] Fraser, G., Wotawa, F., Ammann, P.: Testing with model checkers: a survey. Softw. Test., Verif. Reliab. 19(3): 215-261 (2009)

[18] Bourhfir, C., Aboulhamid, E., Dssouli, R., Rico, N.: A test case generation approach for conformance testing of SDL systems. Computer Communications, vol.24, no.3-4, pp.319–333, 2001

[19] Bourhfir, C., Aboulhamid, E., Dssouli, R., Rico, N.: Automatic executable test case generation for extended finite state machine protocols. IWTCS (1997)

[20] Bourhfir, C., Aboulhamid, E., Dssouli, R., Rico, N.: A guided incremental test case generation procedure for conformance testing for CEFSM specified protocols. IWTCS (1998)

[21] Aichernig B. K., Delgado, C. D.: From Faults Via Test Purposes to Test Cases: On the Fault-Based Testing of Concurrent Systems. LNCS 3922, pp. 324–338 (2006)

[22] Paul, E. Black.: Modeling and Marshaling: Making Tests From Model Checker Counterexamples. In Proc. of the 19th Digital Avionics Systems Conference, pages 1.B.3–1–1.B.3–6 vol.1, 2000.

[23] Yin, X., Jiangyuan, Y., Wang, Z., Shi, X., Wu, J.: Modeling and Testing of Network Protocols with Parallel State Machines, IEICE Transactions on Information and Systems E98. D(12) :2091-2104, 2015

[24] Utting, M., Pretschner, A., Legeard, B.: A taxonomy on model-based testing. University of Waikato, Hamilton, New Zealand (2006)

[25] Bochman, G. V., Khendek, F.: Test Selection Based on Finite State Models. IEEE Transactions on Software Engineering ( 1991)

[26] Clarke, E., Veith, H.: Counterexamples revisited: Principles, algorithms, applications. In Verification: Theory and Practice, volume 2772 of Lecture Notes in Computer Science, pages 208–224, 2004.

[27] Clarke, E., Grumberg, O., McMillan, K. L., Zhao, X.: Efficient generation of counterexamples and witnesses in symbolic model checking. In Proceedings of the 32st Conference on Design Automation (DAC), pages 427–432. ACM Press, 1995

[28] Jéron, T., Morel, M.: Test generation derived from model-checking. In CAV ’99: Proceedings of the 11th International Conference on Computer Aided Verification, pages 108–121, London, UK, 1999. Springer-Verlag. ISBN 3-540-66202-2

[29] Fraser, G., Wotawa, F., Ammann, P.: Issues in using model checkers for test case generation. Journal of Systems and Software 82(9): 1403-1418 (2009)

[30] Ackermann, C.: MC/DC in a nutshell, Fraunhofer CESE, Maryland USA, 2006

[31] Prestschner, A.: Compositional generation of MC/DC integration test suites, Elsevier Science B.V, 2003

[32] Jiangyuan, Y., Wang, Z., Yin, X., Shi, X., Wu, J.: Reachability Graph Based Hierarchical Test Generation for Network Protocols Modeled as Parallel Finite State Machines. 2013 22nd International Conference on Computer Communication and Networks (ICCCN), 1-9 (2013)

[33] Besse, C., Cavalli, A., Kim, M., Zadi, F. : Automated generation of interoperability tests. Testing of Communicating Systems XIV, 169-184, 2002

[34] Su, W., Abrial, J-R. Aircraft landing gear system: Approaches with Event-B to the modeling of an industrial system. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D. editors, ABZ: The Landing Gear Case Study, volume 433, pages 19–35, 2014.

[35] Hansen, D., Ladenberger, L., Wiegard, H., Bendisposto, J. and Michael Leuschel. Validation of the ABZ landing gear system using ProB. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 66–79, 2014.

[36] Mammar, A., Laleau, R.: Modeling a landing gear system in Event-B. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 80–94, 2014.

[37] Méry M., Kumar Singh, N.: Modeling an aircraft landing system in Event-B. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 154–159, 2014.

[38] Kossak, F., Landing gear system: An ASM-based solution for the ABZ case study. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 142–147, 2014.

[39] Dhaussy, Ph., Teodorov C.: Context-aware verification of a landing gear system. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 52–65, 2014.

[40] Berthomieu, B., Dal Zilio, S., Fronc, L.: Model-checking real-time properties of an aircraft landing gear system using Fiacre. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 110–125, 2014.

[41] Arcaini, P., Gargantini, A., Riccobene, E.: Offline model-based testing and runtime monitoring of the sensor voting module. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 95–109, 2014.

[42] Banach, R.: The landing gear case study in hybrid Event-B. In Frédéric Boniol, Virginie Wiels, Yamine Ait Ameur, and Klaus-Dieter Schewe, editors, ABZ: The Landing Gear Case Study, volume 433, pages 126–141, 2014.

[43] Arcaini, P., Gargantini, A., Riccobene, E.: Modeling and analyzing using ASMs: The landing gear system case study. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 36–51, 2014.

[44] Savicks, V., Butler, B., Colley, J.: Co-simulation environment for Rodin: Landing gear case study. In Boniol, F., Virginie Wiels, Ameur, Y.A., Schewe, K.D., editors, ABZ: The Landing Gear Case Study, volume 433, pages 148–153, 2014.

**Mounia Elqortobi** is a Ph.D. degree student in Information Systems Engineering, Concordia University. She received her M. Eng. in QSE and bachelor's degree in CSE from Concordia University (2015, 2010).

**Warda EI-Kholy** received her PhD degree in Information Systems Engineering from Concordia University, she is a lecturer in Menofia University, Egypt.

**Amine Rahj** is a master's degree student in Quality Systems Engineering, Concordia University. He received his bachelor's degree (2015) from INPT, Morocco.

**Jamal Bentahar** is a Full Professor with Concordia Institute for Information Systems Engineering, Concordia University, Canada. His research interests include services computing, applied game theory, computational logics, model checking, multiagent systems, and software engineering.

**Rachida Dssouli** is Full professor and founding Director of Concordia Institute for Information Systems Engineering (CIISE), Concordia University. Her research area is in Testing and Verification.

*Received: April 30, 2019; Accepted: September 17, 2019*

Computer Science and Information Systems 17(1):293–313 https://doi.org/10.2298/CSIS190430041C

# Business Process Specification, Verification, and Deployment in a Mono-Cloud, Multi-Edge Context

Saoussen Cheikhrouhou[1], Slim Kallel[1], Ikbel Guidara[2], and Zakaria Maamar[3]

[1] ReDCAD
University of Sfax
Sfax, Tunisia
saoussen.cheikhrouhou@redcad.tn
slim.kallel@redcad.tn

[2] LIRIS
Claude Bernard Lyon 1 University
Lyon, France
ikbel.guidara@liris.cnrs.fr

[3] College of Technological Innovation
Zayed University
Dubai, U.A.E
zakaria.maamar@zu.ac.ae

**Abstract.** Despite the prevalence of cloud and edge computing, ensuring the satisfaction of time-constrained business processes, remains challenging. Indeed, some cloud/edge-based resources might not be available when needed leading to delaying the execution of these processes' tasks and/or the transfer of these processes' data. This paper presents an approach for specifying, verifying, and deploying time-constrained business processes in a mono-cloud, multi-edge context. First, the specification and verification of processes happen at design-time and run-time to ensure that these processes' tasks and data are continuously placed in a way that would mitigate the violation of time constraints. This mitigation might require moving tasks and/or data from one host to another to reduce time latency, for example. A host could be either a cloud, an edge, or any. Finally, the deployment of processes using a real case-study allowed to confirm the benefits of the early specification and verification of these processes in mitigating time constraints violations.

**Keywords:** Business process, Cloud, Edge, Time constraint, Violation.

## 1. Introduction

Until recently cloud computing has been praised for both offering *elastic* (on-demand) resources and adopting *pay-as-you-go* model [20]. These 2 characteristics made cloud computing extremely popular among Information and Communication Technologies (ICT) practitioners who deployed software applications around the concept of Everything-as-a-Service (*aaS) that cloud computing embraces. Unfortunately, with the continuous advances in ICT and organizations' changing needs, cloud computing has shown some signs of "fatigue" when for instance, real-time applications call for almost zero time-latency. Transferring data to distant clouds is a potential source of delay and opens doors to unwanted interceptions. Luckily edge computing has been introduced to address some clouds' concerns like latency and security. According to Maamar et al. [16], edge and cloud are expected to work hand-in-hand and not compete.

In conjunction with embracing the latest ICT, all organizations capitalize on their Business Processes (BP) to remain competitive, cut costs, and sustain their growth. Referred to as *know-how*, *"...a business process consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations."* [22]. A BP consists of tasks connected together with respect to a particular process model that BP engineers define at design-time. At run-time, the process's tasks are assigned to persons/machines for manual/automated execution. Different works suggest fragmenting BPs and deploying them over the clouds to tap into their elasticity and pay-as-you-go benefits [23]. However, deploying fragmented BPs over the cloud [14] could fail when constraints like temporal are not satisfied resulting into financial penalties, for example.

In line with the works on cloud-driven BP fragmentation, we presented in [5] an approach to formally specify and verify cloud resources allocation to BPs using Timed Petri-Net (TPN). Our objective was to ensure that this allocation does not violate any temporal constraints on BPs. We also presented how BP correctness is formally verified with respect to such constraints. In this paper we extend this approach by fragmenting and deploying free-of-violations time-constrained BPs in a mono-cloud and multi-edge context. We resort to cloud-edge collaboration by verifying at both design-time and run-time where data should be placed (cloud, edge, or either) and where tasks should run as well (cloud, edge, or either). Satisfying temporal constraints in the context of cloud is thoroughly discussed in the literature [4, 12, 13, 19]. However, this remains unexplored in the context of cloud/edge, which constitutes one of our main contributions in this paper.

The remainder of this paper is organized as follows. Section 2 introduces a motivating example. Section 3 briefly presents the approach for verification of cloud- and edge-based resource allocation at both design-time and run-time. We detail the design-time stage in Section 4 and the run-time stage in Section 5. Section 6 gives implementation details. Section 7 discusses related work. Finally, concluding remarks and future work are drawn and presented, respectively, in Section 8.

## 2. Motivating example

Timely responses to customers' requests are a key competitive advantage in any economy. Many organizations tap into *lead time* to sustain this advantage despite the existence of many factors that they cannot, sometimes, control like unannounced strikes and bad weather. Consequences of late delivery are multiple ranging from financial penalties to angry customers and market-share shrinkage. Amazon.com perfectly illustrates how this giant online-retailer achieves its targets by offering competitive prices despite limited lead times. Amazon.com embraces many ICT gadgets like drones in conjunction with customer care best-practices like return policies. On December $7^{th}$, 2016, Amazon.com announced its first drone-based air delivery in Cambridge, UK with a shipment lasting 13 minutes from purchase to drop-off[4]. This would not have happened if Amazon.com

[4] www.engadget.com/2016/12/14/amazon-completes-its-first-drone-powered-delivery. www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

does not operate warehouses in different parts of the world, Cambridge in this case. The objective is to be “close” to customers to avoid potential delivery delays. Collecting data on shipping BPs for the sake of tracking could also benefit from the concept of “closeness” that edge computing promotes. Indeed, sending these data to remote cloud servers for processing could limit Amazon.com’s promptness. By the time the data arrives to these servers, it could become obsolete, be subject to unauthorized collection, and have consumed unnecessary bandwidth. This becomes acute in the case of drone delivery when instant response to unforeseen events (e.g., wind speed) is a must. In the rest of this paper, we illustrate how a delivery BP could benefit from cloud/edge collaboration. To this end, a set of what-if analysis will be carried out to decide which parts of the BP should reside in the cloud, which parts of the BP should reside in the edge, and which parts of the BP should reside in either. This what-if analysis along with cloud/edge collaboration will be illustrated in the rest of this paper using drone-based delivery BP whose tasks are listed in Table 1.

**Table 1.** Tasks defining drone-based delivery BP

| Task | Description |
|---|---|
| $t_1$ | collect and verify delivery instructions |
| $t_2$ | pick items from robot-smart warehouse |
| $t_3$ | define drone position |
| $t_4$ | process drone position and other details |
| $t_5$ | update logbook of flying/delivering drones |
| $t_6$ | analyse received information |
| $t_7$ | send instructions to drone to avoid accidents or violations |
| $t_8$ | update drone position |
| $t_9$ | notify customer about item arrival |

## 3. Approach in a nutshell

This section introduces the main concepts and definitions related to TPNs. Afterwards, it presents the foundations of the proposed approach.

### 3.1. Time Petri-Nets

A PN is formed upon a mathematical theory that uses automated tools to offer an accurate modeling and analysis of systems’ behaviors [3]. Initially, PNs were a formal language without any reference to time or probability. However, for many critical applications, time is a must-have and designers should consider it when analyzing the correct behavior and performance of these applications. TPNs integrate clocks and temporal constraints into transitions to help describe and analyze properly time-dependent systems. TPNs associate a firing time interval [a,b] with each transition, where a and b are rational numbers such that $0 \leq a \leq b$ and $a \neq \infty$. Times a and b for t are relative to the moment at which t was enabled; a and b are referred to as earliest-firing-time and latest-firing-time of t, respectively. Formally, a TPN is a tuple Y = (P, Tr, Pre, Post, $M_0$, IS) where (P, Tr, Pre, Post, $M_0$) is a PN and IS: Tr→ $Q^* * (Q^* \cup \{\infty\})$ is a static interval function that associates each Tr with a time interval IS (Tr)= [min,max] so, that, $Q^*$ is the set of positive rational numbers. Readers are referred to [3] for more details about TPNs.

## 3.2. Foundations

Our approach puts forward recommendations about the "ideal" placement of a BP's tasks and data in a mono-cloud, multi-edge context. Should the recommendations violate any time constraint at design- and/or run-time, then the violations should have a limited impact on completing the BP. By limited we mean an acceptable overtime with respect to some thresholds that BP engineers could set, e.g., 10 extra minutes are still acceptable. Our approach spans over BP design-time (Fig. 1) and run-time (Fig. 2) having each a set of dedicated stages.

At design-time, 4 stages namely *specification*, *placement*, *transformation*, and *verification*, take place.

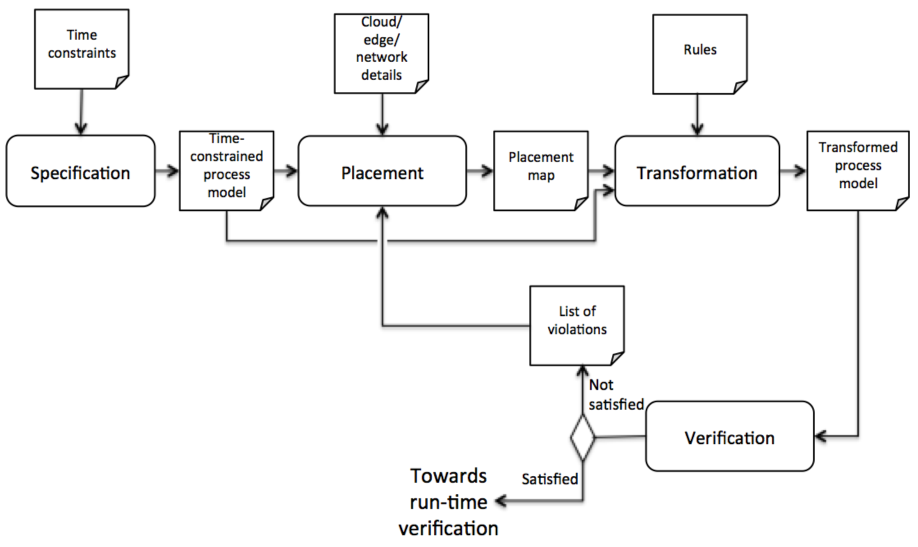


**Fig. 1.** Stages of the approach at design-time

The *specification* stage consists of modeling the BP enriched with time constraints on tasks and data. The *placement* stage takes as input the time-constrained process model and available cloud and edge devices to assign this BP's tasks and data to these devices. This assignment is reported into what we call a placement map. The *transformation stage* uses some in-house rules to convert the placement map into *a transformed process model* that is, in fact, a TPN. Our rules address Business Process Model and Notation (BPMN) constructs, temporal constraints, and time related to transferring data from cloud to edge and *vice-versa*. Finally, the *verification* stage checks if the *TPN process model* contains any time violations that needs to be fixed by going back to the *placement* stage. Otherwise, we proceed with the run-time verification.

At run-time, 3 stages namely *execution*, *ongoing verification*, and *ongoing placement* take place. A BP execution engine will run BP instances. A *log* is also used during run-time to capture the BP's instance execution progress and outcomes.

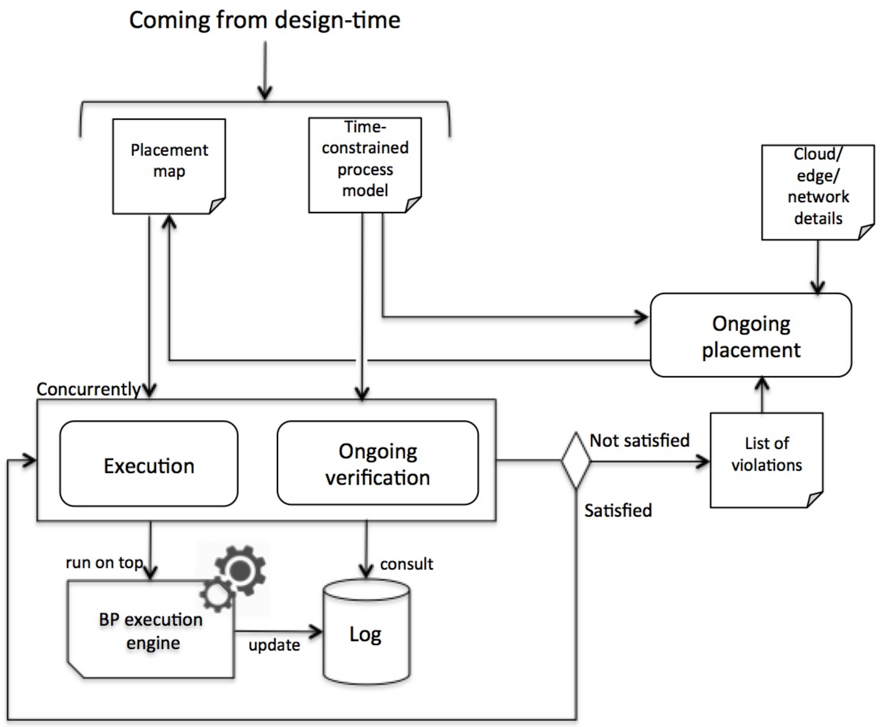


**Fig. 2.** Stages of the approach at run-time

The *execution* stage instantiates the BP's process model so, that, process instances are run over an execution engine. Concurrently to the *execution* stage, the under-execution BP instances are subject to verification to ensure that no time-constraints are violated. This *ongoing verification* stage could produce a *list of violations* so, that, corrective actions are taken to avoid additional violations that could impact the rest of tasks and/or data, and hence, more penalties. The corrective actions are identified thanks to the *ongoing placement* stage that will put forward additional recommendations about potential changes related to where future tasks and/or data should be re-placed. The implementation of these recommendations will be again reflected on the placement map.

## 4. Design-time specification and verification

Since the definition of a BP's temporal constraints over tasks and data during the *specification* stage may lead to deadlocks or unmet deadlines, designers could resort to formal languages to ensure this definition correctness. In this section, we detail further the design-time *transformation* and *verification* stages.

### 4.1. Specification stage

The engineer designs the BP's model using any modeling language, although we recommend BPMN. Then, he defines the time constraints on this BP's tasks [7, 8] and data.

1. Temporal constraints on tasks. First, we consider *intra-task* temporal constraint of type execution *duration*. Let s($t$) (resp., e($t$)) be the starting (resp., the ending) time of a task $t$. Let $d$ be a relative time value representing the duration of this task. *Duration* constraint is defined as per Equation 1:

$$\textit{Duration}(\text{t,d}) \overset{\text{def}}{=} e(t) - s(t) \leq d \tag{1}$$

   Afterwards, we consider *inter-task* temporal constraints of type execution *dependency* that could be:
   - Start-to-Finish (SF): $t_j$ can not finish until $t_i$ has started within a given time interval.
   - Start-to-Start (SS): $t_j$ can not begin before $t_i$ starts within a time interval.
   - Finish-to-Start (FS): $t_j$ can not begin before $t_i$ ends within a time interval. As per Equation 2, $t_j$ should start its execution no later than $MaxD$ time units and no earlier than $MinD$ time units after $t_i$ ends.

$$TD(FS, t_i, t_j, MinD, MaxD) \overset{\text{def}}{=} MinD \leq s(t_j) - e(t_i) \leq MaxD \tag{2}$$

   - Finish-to-Finish (FF): $t_j$ can not finish until $t_i$ has finished within a time interval.
2. Temporal constraints on data. We consider freshness to define the allowable time for a data to remain valid (adapted from [16]) when it is exchanged between tasks that could be running in separate hosts (e.g., any data received 2 seconds from its expected arrival time is not valid). Let $t_i$ be the task producing data $data_{ij}$ that task $t_j$ will consume. Formally, data freshness is defined as per Equation 3:

$$\begin{aligned} FreshData(data_{ij}, t_i, t_j, MaxValue) \overset{\text{def}}{=} \textit{produce}(t_i, data_{ij}) \\ \&\textit{consume}(t_j, data_{ij})\&s(t_j) - e(t_i) \leq MaxValue \end{aligned} \tag{3}$$

Let's recall the case study. The engineer can propose the temporal constraints on BP's tasks and data as per Table 2. For example, FS temporal dependency is specified between $t_4$ and $t_8$. In addition, the data that $t_4$ produces needs to be consumed by $t_5$ in less than 5 seconds.

**Table 2.** Case study's temporal constraints on tasks and data

| task | duration | temporal dependency | data freshness |
|---|---|---|---|
| $t_1$ | 10 milliseconds | | |
| $t_2$ | 10 milliseconds | TD(FS,$t_1$,$t_2$,1 millisecond,5 milliseconds) | |
| $t_3$ | 12 minutes | | |
| $t_4$ | 10 milliseconds | | |
| $t_5$ | 10 milliseconds | | FreshData($t_4$,$t_5$,5 seconds) |
| $t_6$ | 10 milliseconds | | |
| $t_7$ | 10 milliseconds | | FreshData($t_6$,$t_7$,5 seconds) |
| $t_8$ | 10 milliseconds | TD(FS,$t_4$,$t_8$,1 millisecond,8 milliseconds) | FreshData($t_7$,$t_8$,5 seconds) |
| $t_9$ | 10 milliseconds | | |

## 4.2. Placement stage

The engineer proceeds with a first assignment of the BP's tasks and data to cloud/edges. This assignment results into a *placement map* that considers how critical the tasks are (i.e., must execute), how dependent the tasks are, what time constraints are imposed on tasks and data (e.g., duration and freshness), and any other time-related details (e.g., data transfer between hosts). How good this first assignment will be checked out during the *verification* stage (e.g., no time constraints' violation). However it is worth noting that the *placement* stage could be triggered again if the *verification* stage reports any temporal violation. These violations are manually analyzed by BP engineers to identify their causes. Consequently, the engineer can come along with some corrective actions that would address these violations like reassigning some tasks/data from cloud to edge and *vice versa*. It is highly recommended to address any violation prior to executing the BP.

Table 3 presents an initial placement that the BP engineer could come up with. For example, "send instructions to drone to avoid accidents or violations" task is deployed on the edge device to be each time near to the drone while flying since the decision of changing position of drone should be made as quickly as possible. Indeed, The use of edge nodes is to reduce the overall response time and traffic to distant clouds. Contrarily, sending instructions to edge nodes will be forwarded to " update logbook of flying/delivering drones" task, which is deployed on the cloud, where high volumes of data can be processed.

**Table 3.** Initial placement of drone-based delivery BP's tasks

| task | placement |
|---|---|
| $t_1$, $t_2$, $t_5$, $t_6$, $t_9$ | cloud |
| $t_4$, $t_7$ | edge |

## 4.3. Transformation stage

This stage relies on a set of *rules* defined in compliance with model-driven engineering principles. The objective is to convert a time-constrained BPMN process into a TPN model.

It all begins by transforming BPMN basic elements like start/end events, tasks, and gateways into TPN. Readers are referred to [6] for a complete description of the transformation rules. We focus on rules related to BP task placement. For instance, a task with an execution duration will be transformed into 1 place and 2 transitions labeled with clocks depending on the task's duration (Fig. 3 (a)). The duration is set by the engineer depending on whether the task will run on either the cloud or the edge as per the *placement map*. Fig. 3 also includes the transformation of some temporal dependencies between tasks such as SS, FS, and FF.
Let us now focus on the transformation rules for data placement. For a given data $data_i$, it can be produced by a task $t_i$ that runs on a certain host ($h_c$ to refer to cloud) and will be consumed by another task $t_j$ that runs on a different host ($h_y$ to refer to edge). In this case, $data_i$ should be transferred from one host to another and hence, *latency* time ($l^{d_i}_{h_x:h_y}$) needs to be considered. Data latency is transformed into 2 places and 1 transition as per Fig. 3 (b).

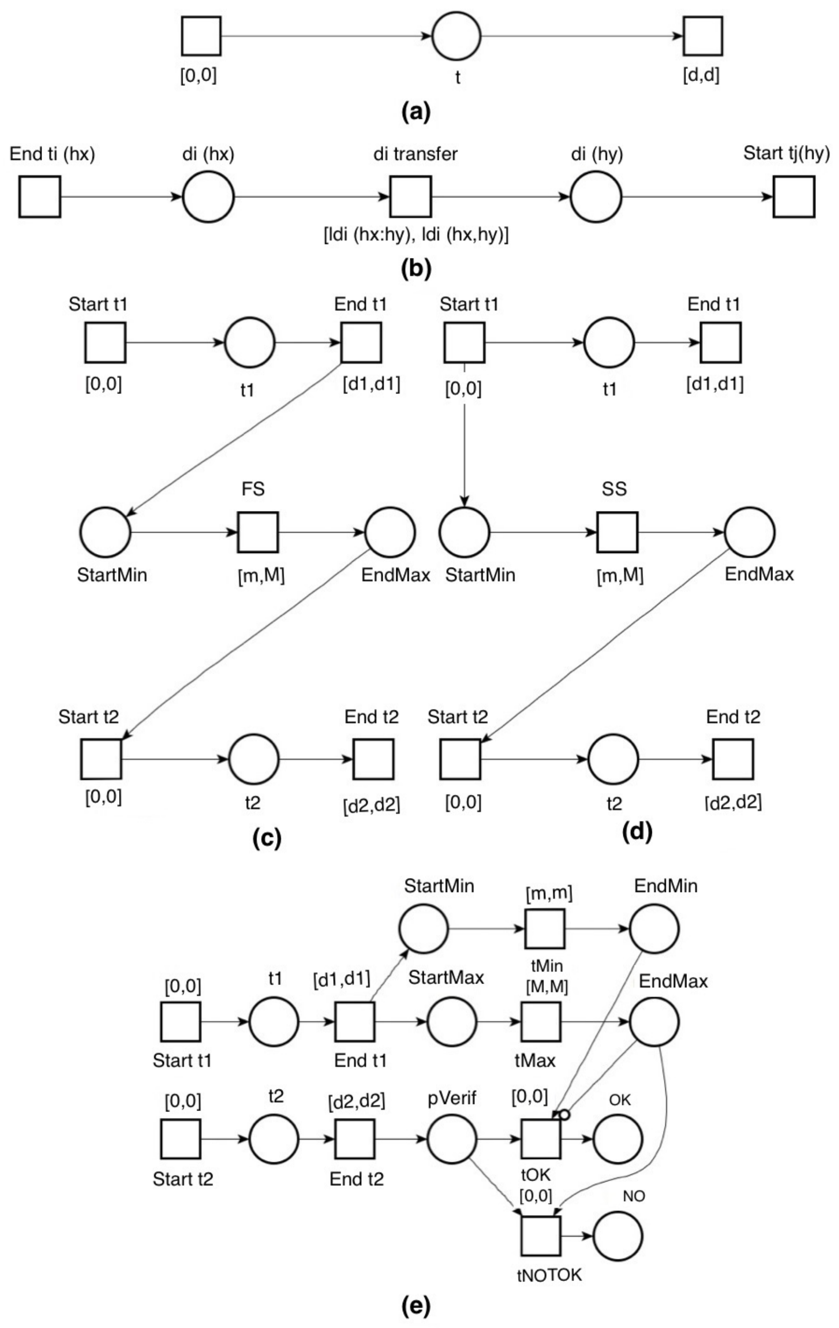


**Fig. 3.** Illustration of some transformation rules

### 4.4. Verification stage

Depending on a BP complexity, software testing and/or manual checks could turn out insufficient and hence, time constraints violations could still arise. To address this concern, we propose model checking as an accurate and exhaustive verification alternative [2]. The BP engineer checks the correctness of the time-constrained, TPN-based process model using a model checker like TIme petri Net Analyzer (TINA) [3].

If the check reports any temporal violation, a *list of violations* (could be based on counter-examples) is reported and the designer is referred back to the *placement* stage. At this stage, a threshold could be put to limit the number of times the designer has to initiate this stage. Otherwise, the designer proceeds with executing the BP. Violations at this stage could refer to deadlocks due to temporal inconsistencies (e.g., a task minimum duration exceeding the maximum delay of initiating a dependent task) and missed deadlines (e.g., process ending in 5h but the designer has set 4h as a maximum).

Concretely, we formally verify a TPN-based process model with respect to the following properties: deadlock freshness, process deadlines, and data freshness. These properties are written in State/Event Linear time Temporal Logic (S/E LTL [17]).

- $\Diamond$ (- dead) : to check that a process is free of deadlocks.
- $\Diamond$ (- notdeadline_process) : to verify if a deadline has been met. This means that notdeadline_process place (associated with an observer for the deadline property) is false throughout the whole path leading to this place.
- $\Diamond$ (- notfresh_data) : to verify if the freshness time $f^{data_i}$ related to $data_i$ has been met. This means that notfresh_data place (associated with an observer for the *freshness* property) is false throughout the whole path leading to this place.

The verification of the latter properties on the generated TPN of the case study reports that the process is deadlock free, and meets the deadline (18 minutes), and all data respect their freshness constraints.

Despite the virtues of design-time model-checking that could guarantee certain free-of-time violations, many run-time events and actions could happen triggering such violations. For instance, expected duration times could suddenly change due to power outage and hence, raising questions about the validity of design-time model-checking. To this end, we propose run-time verification to monitor process execution. In other words, verification at design-time is "preventive" rather than "curative", which backs our run-time verification.

## 5. Run-time verification

During execution, current time values like duration and freshness may change resulting into gaps with the estimated values and thus, can violate time constraints (e.g., a high latency can make data obsolete and a late data arrival can delay a task execution). Thus, BP instances need to be continuously monitored to ensure the satisfaction of their time constraints at run-time. In this section, we detail further run-time's different stages (Fig. 2).

### 5.1. Execution stage

A BP instance runs on top of an execution engine that assigns tasks to persons/machines, transfers data between tasks, stores data, etc. Both the *time-constrained process model* and the *placement map* constitute inputs to the *execution* stage that continuously updates the *log repository* that contains details about instances execution like instance id, exchanged data, and execution outcome (success or failure).

### 5.2. Ongoing verification stage

Because of the dynamic nature of environments in which BP instances are executed, we adopt thresholds that would give engineers some leeway (i.e., extra time) prior to raising the violation flag. In project management [15], this leeway is known as slack time. We tap into our previous work on service execution [11] to define thresholds with respect to a constraint satisfaction model that captures both task duration and data transfer that impacts data freshness (Constraints (4) to (13)). Our objective is to recommend a maximum threshold for task duration while guaranteeing data freshness.

$$\text{maximize } Duration(t_j) \tag{4}$$

$$Agg_{t_i \in \mathcal{T}}(Duration(t_i)) \leq deadline, \forall t_i \in \mathcal{T} \tag{5}$$

$$e(t_i) \leq s(t_k), \forall t_k \in \mathcal{T}, t_i \in \mathcal{P}d(t_k) \tag{6}$$

$$Duration(t_i) = EstimatedDuration(t_i), \forall t_i \in \mathcal{T}, i \neq j \tag{7}$$

$$s(t_i) + Duration(t_i) = e(t_i), \forall t_i \in \mathcal{T} \tag{8}$$

$$e(t_i) + MinD \leq s(t_k),\ \forall\, TD(FS, t_i, t_k, MinD, MaxD) \in \mathcal{TD} \tag{9}$$

$$s(t_k) \leq e(t_i) + MaxD,\ \forall\, TD(FS, t_i, t_k, MinD, MaxD) \in \mathcal{TD} \tag{10}$$

$$e(t_i) + Transfer(d_i) \leq s(t_k),\ \forall\, DD(d_i, t_i, t_k, h_i^s, h_k^s) \in \mathcal{DD} \tag{11}$$

$$Duration(t_j) \in [EstimatedDuration(t_j), deadline] \tag{12}$$

$$st_i, et_i \in [0, deadline], \forall t_i \in \mathcal{T} \tag{13}$$

The maximum threshold of each task $t_j$ is equal to its maximum allowed duration value (i.e., $Duration(t_j)$). Constraint (5) guarantees that the global duration constraint is satisfied. The aggregation function *Agg* depends on the distinguish characteristics of quality attributes (i.e., additive, average, multiplicative, and max-Operator) and the structure of the BP (i.e., structural patterns involved such as sequence, parallel, choice, and loop). Here, the duration is considered as max-operator quality attribute. Hence, the aggregation function is as follows:

- $Agg_{t_i \in \mathcal{T}}(Duration(t_i)) = \sum_{i=1}^{n} Duration(t_i)$ for sequence patterns where $n$ is the number of tasks in the sequence pattern.
- $Agg_{t_i \in \mathcal{T}}(Duration(t_i)) = max_{i=1}^{n}\{Duration(t_i)\}$ for parallel patterns where $n$ is the number of tasks in the parallel pattern.
- $Agg_{t_i \in \mathcal{T}}(Duration(t_i)) = Duration(t_k)$ for choice patterns where $t_k$ is the selected task in the choice pattern.

- $Agg_{t_i \in \mathcal{T}}(Duration(t_i)) = \alpha_i Duration(t_i)$ for loop patterns where $\alpha_i$ is the number of loops of the task $t_i$.

Constraint (6) deals with dependencies between tasks where $\mathcal{P}d(t_k)$ denotes the set of immediate predecessors of the task $t_k$ and $s(t_i)$ and $e(t_i)$ denote the start time and end time of the task $t_i$, respectively. The duration of each task $t_i$ is equal to the estimated duration specified at design time (Constraint (7)). Moreover, the end time of each task $t_i$ is equal to the sum of its start time and its duration (Constraint (8)). To deal with temporal dependencies between tasks, we use Constraints (9) and (10) where $\mathcal{TD}$ is the set of temporal dependency. For simplicity, we consider only finish-to-start dependencies. Constraint (11) guarantees data freshness where $\mathcal{DD}$ is the set of data dependencies between tasks. $DD(data_i, t_i, t_k, h_i^s, h_k^s)$ denotes the data dependency between the task that produces the data $data_i$ (i.e., task $t_i$) and the task that consumes the data $data_i$ (i.e., task $t_k$) when $t_i$ is executed in the host $h_i^s$ and $t_k$ is executed in the host $h_k^s$. The time required to the transfer of the data $data_i$ from $t_i$ to $t_k$ is denoted by $Transfer(data_i)$. The domain of the maximum duration threshold and the start and the end time are presented in Constraints (12) and (13), respectively.

The maximum threshold of each task $t_i$ is denoted by $T_i^M$. During execution, if one of the maximum thresholds is exceeded, the global duration is violated and thus, corrective actions should be taken which will be discussed in the ongoing placement stage. We note that maximum thresholds have to be recomputed after each violation.

In conjunction with the maximum thresholds, we identify a set of intermediary thresholds for all tasks. They are used to trigger placement actions prior to a global constraint violation. Each time a deviation exceeds an intermediary threshold, the placement of tasks is adjusted so, that, possible violations in the remaining non-executed tasks can be either reduced or prevented. The aim is to avoid delaying the placement until a violation of a global constraint occurs on the one hand, and, on the other hand avoid triggering placement actions each time a violation is observed, which can decrease the efficiency of the proposed approach. The intermediary threshold of each task $t_i$ is denoted by $T_i^I$ that is the average between the estimated duration value (before the execution) and the maximum threshold of the same task.

## 5.3. Ongoing placement

To ensure a continuous execution of the different BP instances while guaranteeing the satisfaction of all constraints, the BP instances need to continuously react to varying conditions during execution. We present hereafter the ongoing placement of tasks and data each time a deviation of an intermediary threshold or a violation occurs. To enhance the efficiency of the placement, we identify a set of alternative hosts for each task. Thus, a local placement can be easily applied to change the placement of tasks/data and guarantee the satisfaction of the different constraints. In case of a deviation/violation during execution, we propose to change the host of one or more tasks using the alternative hosts. We note that alternative hosts are updated and re-identified during execution each time a change occurs in the BP instances.

**Alternative hosts** Prior to execution, we define a set of alternative hosts for each task $t_i \in \mathcal{T}$ denoted by $H_{alti}$. Alternative hosts should satisfy the maximum thresholds of their corresponding tasks. Hence, an alternative host $h_j \in H_{alti}$ enables to execute task $t_i$ to start and end its execution without exceeding the maximum temporal thresholds (i.e., $Duration(t_i) \leq T_i^M$) and guarantee the freshness of data. In fact, since we check the satisfaction of the transfer time when computing maximum thresholds (Constraint (11) in the model from Constraints (4) to (13)), all hosts that satisfy the maximum thresholds will guarantee the satisfaction of the task duration and data freshness at the same time. A host $h_j$ that satisfies the maximum thresholds of its corresponding task is denoted by ($h_j$ sat $T_i^M$). Contrarily, $\neg$($h_j$ sat $T_i^M$) denotes that the maximum threshold is not satisfied by the host $h_j$. In this latter case, the host will not be considered in the set of alternative hosts of its corresponding task. We note that the set of alternative hosts is updated each time a deviation or a violation occurs to take into account the new values of the duration of the already executed tasks and the new values of the maximum thresholds. We rank the set of alternative hosts based on the duration of the execution of the tasks in each host. Hence, the host that guarantees the minimum execution duration will be ranked first and so on. We denote by $Duration(t_i)^{h_j}$ the execution duration of the task $t_i$ when it is executed in host $h_j$.

**Changing hosts** We denote by $PM^* = \{h_1^s, ..., h_i^s, ..., h_n^s\}$ the placement map with $h_i^s$ is the selected host for the task $t_i$. By ongoing placement denoted by $PM_{new}^*$, we refer to changes in the placement map.

Algorithm 1 handles changes in the ongoing placement map. If the execution time of task $t_i$, while being executed in host $h_i$ deviates but does not exceed the intermediary threshold ($T_i^I$), then this will not affect affect the placement map (lines 4 to 6). If the deviation is between the intermediary and maximum thresholds (line 7), then we proceed with first, the maximum thresholds and the set of alternative hosts $H_{alti}$ are updated for each non-executed task considering the values of the already executed tasks (lines 8 and 9) where $T_{ne}$ denotes the set of non-executed tasks. We note that when updating the thresholds, the execution duration of the already executed tasks are considered in the constraint satisfaction model from (4) to (13) (Section 5.2). Moreover, the set of alternative hosts is updated by identifying the new alternative hosts based on the new values of maximum thresholds. Then, if the first alternative host guarantees a better execution duration than the initial selected host, it will be considered in the ongoing placement map (lines 10 to 12) where $h_i^1$ denotes the best host for the task $t_i$ when considering the already executed tasks. The aim of this step is to avoid the accumulation of deviations during execution in order to prevent possible violations. If a violation exceeds the maximum threshold (line 15), then, the ongoing execution is no more satisfactory. In this case, we update the maximum thresholds and alternative hosts for the non executed tasks (line 17). If there is at least one host in the set of alternative hosts for a non-executed task, the selected host of this task will be substituted by the first alternative host (lines 18 to 23).

If the ongoing placement is modified, all thresholds and alternative hosts for all non-executed tasks will be updated (lines 26 to 30).

**Algorithm 1** Ongoing placement

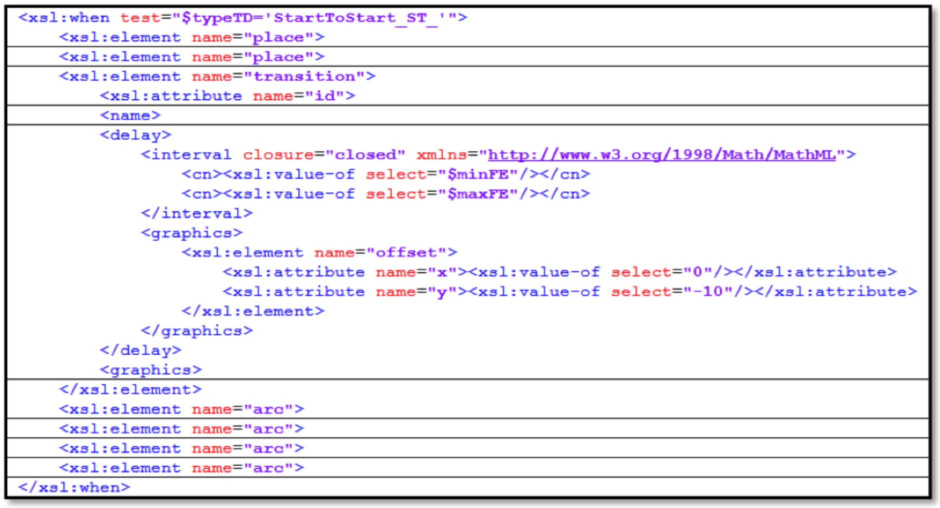
```
1:  Input: Monitoring results of task $t_i$, the placement map $PM^*$
2:  Output: The new placement map $PM^*_{new}$
3:  $PM^*_{new} = \emptyset$
4:  if ($h_i^s$ sat $T_i^I$) then
5:      $PM^*_{new} = PM^*$
6:  end if
7:  if $\neg$($h_i^s$ sat $T_i^I$) and ($h_i^s$ sat $T_i^M$) then
8:      for each $t_i \in T_{ne}$ do
9:          update($T_i^M, H_{alti}$)
10:         if $Duration(t_i)^{h_i^1} < Duration(t_i)^{h_i^s}$ then
11:             $PM^*_{new} = PM^* \setminus \{h_i^s\} \cup \{h_i^1\}$
12:         end if
13:     end for
14: end if
15: if $\neg$($h_i^s$ sat $T_i^M$) then
16:     while $PM^*_{new} = \emptyset$ and $t_i \in T_{ne}$ do
17:         update($T_i^M, H_{alti}$)
18:         if $H_{alti} \neq \emptyset$ then
19:             $PM^*_{new} = PM^* \setminus \{h_i^s\} \cup \{h_i^1\}$
20:             break;
21:         else
22:             move to $T_{i+1}$
23:         end if
24:     end while
25: end if
26: if $PM^*_{new} \neq \emptyset$ then
27:     for each $t_i \in T_{ne}$ do
28:         update($T_i^I, T_i^M, H_{alti}$)
29:     end for
30: end if
```

## 6. Implementation

This section discusses the implementation work that was carried out in terms of experiments and performance evaluation. In compliance with how our approach is designed, we discuss the implementation at design-time and then run-time.

### 6.1. Design-time implementation

We extended the work we presented in [13] that resulted into the development of an Eclipse plug-in. Using this plug-in, a designer represents a BP's 2 things: needs of resources (cloud and/or edge resources) and time-constrained activities. Next, a source model is generated as an XML document. In our work, we focused on implementing rules that transform BPs into TPN. This is done using an XSLT file containing the transformation rules. Fig. 4 exhibits an XSLT excerpt that transforms a SS temporal dependency into 2 places and 1 transition with a delay of minFE and maxFE.



```
<xsl:when test="$typeTD='StartToStart_ST_'">
    <xsl:element name="place">
    <xsl:element name="place">
    <xsl:element name="transition">
        <xsl:attribute name="id">
        <name>
        <delay>
            <interval closure="closed" xmlns="http://www.w3.org/1998/Math/MathML">
                <cn><xsl:value-of select="$minFE"/></cn>
                <cn><xsl:value-of select="$maxFE"/></cn>
            </interval>
            <graphics>
                <xsl:element name="offset">
                    <xsl:attribute name="x"><xsl:value-of select="0"/></xsl:attribute>
                    <xsl:attribute name="y"><xsl:value-of select="-10"/></xsl:attribute>
                </xsl:element>
            </graphics>
        </delay>
        <graphics>
    </xsl:element>
    <xsl:element name="arc">
    <xsl:element name="arc">
    <xsl:element name="arc">
    <xsl:element name="arc">
</xsl:when>
```

**Fig. 4.** Transformation rule of a SS temporal dependency as XSLT

The result of the transformation is an XML document that describes the generated TPN. An example of this TPN based on the drone delivery is given in Fig. 5. *data1Edge* and *data1Cloud* places and *Tlatency1* transition labeled by the time interval [latencyEC,latencyEC] specify data transfer time between t_4 (running on edge) and t_5 (running on cloud). We consider "latencyEC" equals to 200 milliseconds as latency edge to cloud. *StartMax2* and *EndMax2* places and *FS2* transition labeled by the time interval [m2,M2] with m2= 1 millisecond and M2= 8 milliseconds specify a temporal dependency constraint between t_4 and t_8.

Finally, we formally verify the matching between the activities, temporal constraints, and resource temporal constraints. The generated TPN are the inputs for the TINA model checker.

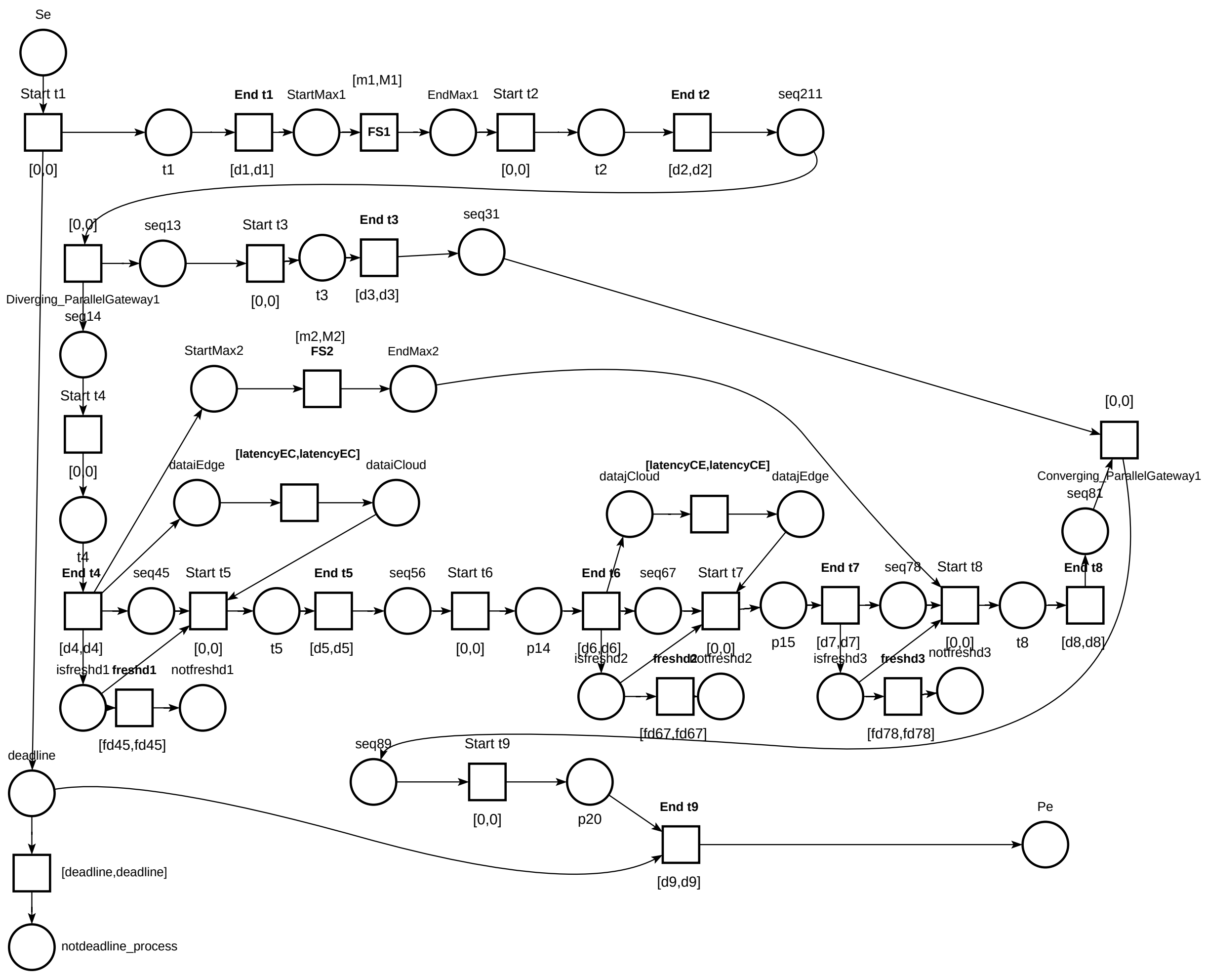


**Fig. 5.** Generated TPN of the drone delivery BP

## 6.2. Run-time implementation

We investigated how our approach behaves at run-time. First, we evaluate the success rate (I suggest to remove this. This is not the definition of success rate. i.e., converging "quickly" to optimal solution) and computation time. Thus, constraints (4) to (13) and algorithm 1 were used to test a BP of 9 tasks generated randomly. As candidate hosts, we used a mono-cloud with multiple VMs and 10 edges respectively. Constraint satisfaction models are implemented using the constraint solver Choco[5].

First, the success rate is compared to First In First Out (FIFO), in which, the first come host is first assigned to task without taking into account the host that has the best duration and the replacement of hosts is delayed until a global violation occurs which can affect the

[5] http://www.emn.fr/z-info/choco-solver/

execution of tasks contrarily to our approach which allows enhancing the selected hosts during execution as soon as a deviation occurs.

Fig. 6 depicts the success rate in response to the number of deviations in process tasks which are generated randomly at run-time. All deviations are assumed to be less than the maximum thresholds (see subsection 5.2). The positions of deviations are generated randomly. Experimental results show that our approach has a higher success rate in comparison to FIFO approach. Indeed, it reacts to changes as soon as they occur which increases the likelihood to find a solution. In contrast to FIFO, it might be the case where no solution is found after a violation which can be caused by multiple deviations.

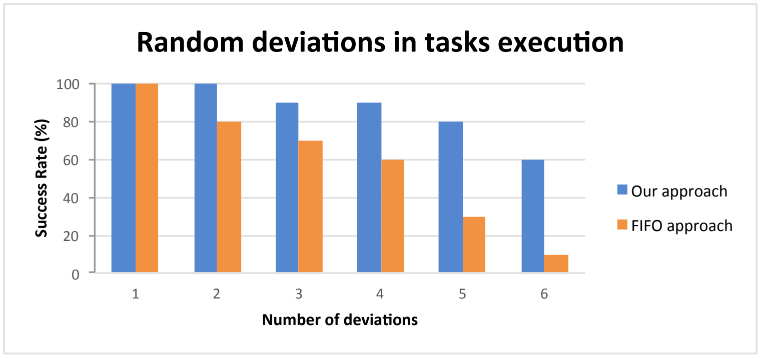


**Fig. 6.** Success rate versus number of deviations

Second, we calculate the computation time of our approach. It takes between 7 and 200 milliseconds to find a solution. These time values are taken while considering random task violations exceeding the maximum thresholds and thus, hosts changing is mandatory to guarantee the satisfaction of all process constraints. Indeed, solutions can be found by changing hosts using the alternative hosts (see subsection 5.3). In addition, new placement actions are taken as soon as deviation occurs in parallel to the execution and does not cause the interruption of the execution. Results show that the computation time of our approach is “negligible” compared to the deadline of the expected process. Results also show that the computation time increases proportionally to the number of deviations in the process.

## 7. Related work

Our related work consists of 2 parts. The first part is about BP formal specification. The second part is about BP allocation into clouds. Many works in the literature address the issue of defining BP formal specification. First, Dijkman et al. in [9] propose a formal BPMN semantics defined in terms of a transformation to standard PN. The transformation has been implemented as a tool that generates Petri Net Markup Language (PNML) code. But, the authors do not consider any temporal dimension in their analysis. Rachdi et al. [19], propose an approach that takes into account time concepts in BPMN processes. They present a formal semantics of BPMN defined in terms of transformation to TPN but without taking into consideration of temporal constraints as in our work nor the notion of resources. Cheikhrouhou et al.[7, 8] address the problem of formal specification and verification of temporal constraints of activities using timed automata. But, resources were not considered. Hachicha et al [12] extend of the BPMN meta-model to optimally manage cloud resources. They formalize the resources consumed using a shared knowledge base. Therefore, the authors propose a semantic framework for BPs enriched by cloud resources. However, the temporal perspective for BPs is out of reach.

Several works have addressed the specification and formal verification of cloud resources in BPMN. Boubaker et al.[4] validate the consistency of the allocation of cloud resources using Event- B. The latter is used to formally specify cloud resource allocation policies in business process models and to verify its accuracy based on user requirements and resource properties. However, in this work, BPs are not enriched by time constraints. Ben Halima et al. [13] formally specify temporal constraints on pricing strategies for cloud resources, especially virtual machines, and on BPMN activities. This specification is translated into timed automata to formally verify the correspondence between the time constraints of the business process and the cloud resources. But, this work does not deal with constraints on process data nor support automatic BPMN mapping to timed automata, which can lead to errors during the transformation. Several searches extend BPMN with time constraints and cloud resource perspectives and use formal verification. Watahiki et al. [21] extend BPMN to handle time constraints. They also provide an automatic mapping of extended BPMN to timed automata. This approach aims to verify certain characteristics, such as deadlock. However, the scope of this article is limited to a small subset of BPMN elements. In addition, the extension proposed in this work gives specific temporal constraints to a single task of the business process model and does not take into account time constraints related to a set of activities such as temporal dependency. There is previous research that aims to check whether the selected cloud resource meets the time constraints of business processes. Du et al.[10] propose to dynamically verify the temporal constraints of multiple simultaneous business processes with resources. However, the work does neither deal with data flow and their temporal constraints, nor with edge resources. While almost works in the litterature focus only on control flow verification, process data flow modeling is of similar importance. In [1], the approach generates a PN process model that captures the control flow along with data aspects of BPMN process models. The approach detects data-flow errors in BPMN 2.0 process models, such as missing or unused data and possible deadlocks in the PN model. However, the approach does not deal with temporal constraints on process data. The approach in [18] focuses on the resource allocation problem in fog computing based on Priced Timed Petri nets (PTPN). Provided with a group of pre-allocated resources, the designer can choose the satisfying resources autonomously while considering both the price and the cost to execute a process's tasks as the credibility evaluation of both users and fog resources. From one hand, the constructed PTPN models of process tasks does not deal with temporal constraints of the process such as deadline nor with cloud resources and the delay caused by data transfer from one host to another. Furthermore, the PTPNs were used as a formal background for a proposed algorithm that predicts task completion time. Thus, no formal verification is proposed and simulation results are presented. To the best of our knowledge, there is no research attempts to verify process models while addressing both cloud and edge resource allocation, data flow aspects, and their temporal constraints. Such verification is scarce at both design-time and run-time.

## 8. Conclusion

This paper presented an approach to specify, verify, and deploy BPs in a mono-cloud, multi-edge context. These BPs are bound to time constraints whose satisfaction requires placing their tasks and data in the appropriate hosts, whether cloud, edge, or either. This placement is continuous because of the dynamic environment in which BPs are expected to execute. Indeed, communication networks could become jammed and some computation resources could become unavailable. Either reason could lead to delays in executing tasks and/or transferring data. Delays raise time violations, which themselves mean penalties of all types, financial, market share loss, etc. Our specification, verification, and deployment approach happens at both BP design-time and BP run-time involving different stages such as specification, ongoing placement, verification, and execution. One of the run-time stages, ongoing placement, included a set of thresholds that give BP engineers some leeway (i.e., extra time) prior to raising any violation flag. In term of future work, we would like to extend the proposed approach to deal with several simultaneous changes in a BP's tasks and data placement and propose strategies to handle potential conflicts between corrective actions. Furthermore, we aim to further compare our approach to other approaches that suggest backup solutions.

## References

1. Ahmed Awad, Gero Decker, and Niels Lohmann. Diagnosing and repairing data anomalies in process models. In Stefanie Rinderle-Ma, Shazia Sadiq, and Frank Leymann, editors, *Business Process Management Workshops*, pages 5–16. Springer Berlin Heidelberg, 2010.
2. Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.
3. Bernard Berthomieu and François Vernadat. Time petri nets analysis with TINA. In *Proceedings of the Third International Conference on the Quantitative Evaluation of Systems (QEST*.
4. Souha Boubaker, Walid Gaaloul, Mohamed Graiet, and Nejib Ben Hadj-Alouane. Event-b based approach for verifying cloud resource allocation in business process. In *Proceedings of the 2015 IEEE International Conference on Services Computing, SCC*, pages 538–545, 2015.
5. Saoussen Cheikhrouhou, Nesrine Chabouh, Slim Kallel, and Zakaria Maamar. Formal specification and verification of cloud resource allocation using timed petri-nets. In *Proceedings of the New Trends in Model and Data Engineering - MEDI 2018 International Workshops, DETECT, MEDI4SG, IWCFS, REMEDY, Marrakesh, Morocco, October 24-26, 2018*, pages 40–49, 2018.
6. Saoussen Cheikhrouhou, Nesrine Chabouh, Slim Kallel, and Zakaria Maamar. Transformation of timed BPMN busines processes and cloud resources into timed Petri-Nets. Technical report, http://www.redcad.tn/projects/bpmn2tpn/technicalreport-0618.pdf, 2018.
7. Saoussen Cheikhrouhou, Slim Kallel, Nawal Guermouche, and Mohamed Jmaiel. Toward a time-centric modeling of business processes in BPMN 2.0. In *The 15th International Conference on Information Integration and Web-based Applications & Services, IIWAS*, page 154, 2013.
8. Saoussen Cheikhrouhou, Slim Kallel, Nawal Guermouche, and Mohamed Jmaiel. The temporal perspective in business process modeling: a survey and research challenges. *Service Oriented Computing and Applications*, 9(1):75–85, 2015.

9. Remco M Dijkman, Marlon Dumas, and Chun Ouyang. Formal semantics and analysis of bpmn process models using petri nets. *Queensland University of Technology, Tech. Rep*, 2007.
10. YanHua Du, PengCheng Xiong, YuShun Fan, and Xitong Li. Dynamic checking and solution to temporal violations in concurrent workflow processes. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(6):1166–1181, 2011.
11. I. Guidara, I. Al Jaouhari, and N. Guermouche. Dynamic selection for service composition based on temporal and qos constraints. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 267–274, June 2016.
12. Emna Hachicha and Walid Gaaloul. Towards resource-aware business process development in the cloud. In *Proceedings of the 29th IEEE International Conference on Advanced Information Networking and Applications, AINA*, pages 761–768, 2015.
13. Rania Ben Halima, Imen Zouaghi, Slim Kallel, Walid Gaaloul, and Mohamed Jmaiel. Formal verification of temporal constraints in business processes and allocated cloud resources. In *Proceedings of the 32nd IEEE International Conference on Advanced Information Networking and Applications, AINA*, 2018.
14. Slim Kallel, Zakaria Maamar, Mohamed Sellami, Noura Faci, Ahmed Ben Arab, Walid Gaaloul, and Thar Baker. Restriction-based Fragmentation of Business Processes over the Cloud. *Concurrency and Computation: Practice and Experience*, 2019.
15. Olivier Lambrechts, Erik Demeulemeester, and Willy Herroelen. Time slack-based techniques for robust project scheduling subject to resource uncertainty. *Annals OR*, 186(1):443–464, 2011.
16. Z. Maamar, B. Thar, N. Faci, E. Ugljanin, M. Al Khafajiy, and V. Burégio. Towards a Seamless Coordination of Cloud and Fog: Illustration through the Internet-of-Things. In *Proceedings of the 34th ACM/SIGAPP Symposium On Applied Computing (SAC'2019)*, Limassol, Cyprus, 2019.
17. Madhavan Mukund. Linear-time temporal logic and bchi automata, 1997.
18. Lina Ni, Jinquan Zhang, Changjun Jiang, Chungang Yan, and Kan Yu. Resource allocation strategy in fog computing based on priced timed petri nets. *IEEE Internet of Things Journal*, 4(5):1216–1228, Oct 2017.
19. Anass Rachdi, Abdeslam En-Nouaary, and Mohamed Dahchour. Liveness and reachability analysis of BPMN process models. *CIT*, 24(2):195–207, 2016.
20. Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 228–235. IEEE, 2010.
21. Kenji Watahiki, Fuyuki Ishikawa, and Kunihiko Hiraishi. Formal verification of business processes with temporal and resource constraints. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Anchorage, Alaska, USA, October 9-12, 2011*, pages 1173–1180, 2011.
22. Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.
23. Xun Xu. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1):75 – 86, 2012.

**Saoussen Cheikhrouhou** obtained her diploma of engineering in 2009 and master degree in computer science in 2010 and her Ph.D. in 2015 from National School of Engineering of Sfax (University of Sfax, Tunisia). She joined the Faculty of Economics and Management (Tunisia) as Associate Professor of Computer Science in 2015. Her research interests include the business process management field and Time-aware Business Processes. More

details are available on her home page:
http://www.redcad.org/members/saoussen.cheikhrouhou/

**Slim Kallel** obtained his diploma of engineering and masters degree in computer science from National Engineering School of Sfax (University of Sfax, Tunisia) in 2005 and his Ph.D. from Darmstadt University of Technology (Germany) in 2011. He joined the University of Sfax as Assistant Professor of Computer Science in 2009. He became an Associate Professor in 2012. His work focused on the specification and the implementation of Time-aware business process, and adaptive systems.

**Ikbel Guidara** is Assistant Professor at Claude Bernard University of Lyon 1 and member of SOC research team at LIRIS-CNRS Lyon-France. Her research interests include Service-Oriented Computing, Business Process, Quality-of-Service (QoS) driven service selection and Internet of Things.

**Zakaria Maamar** is a Professor in the College of Technological Innovation at Zayed University, Dubai, UAE. His research interests include Internet-of-Things, social computing, and business process management. Zakaria has extensively published in different peer reviewed journals and conferences, regularly serves on the program and organizing committees of several international conferences and workshops. He also serves on the editorial boards of many international journals.

*Received: April 30, 2019; Accepted: September 12, 2019.*

Computer Science and Information Systems 17(1):315–338 https://doi.org/10.2298/CSIS190501042A

# A Tool-assisted Method for the Systematic Construction of Critical Embedded Systems using Event-B

Pascal André, Christian Attiogbé and Arnaud Lanoix

LS2N CNRS UMR 6004 - University of Nantes
{firstname.lastname}@univ-nantes.fr

**Abstract.** Embedded control systems combine digital and physical components, leading to complex interactions and even complexity of their development. In [4] we proposed a method to build such complex systems in a systematic way. The overall method starts from an abstract model of the physical environment of the considered system and its controller. The method consists in a sequence of refinement steps, in the spirit of Event-B, that gradually introduces design details from an abstract level, until more concrete levels. Two main refinement processes are distinguished: one to capture the global model, the other to detail it; we provide through the method the guidelines to accompany these two refinement processes. But there were a lack of assistance tools. The designers need to be assisted by tools to guide them, to automate partially the refinements and to help in proving more easily model properties. We illustrate the method with the landing gear system case study and choosing the Event-B tool Rodin for illustration; we make it explicit the tools requirements for such a general method and, we introduce a tool support to assist the user in applying the method in combination with standard Event-B tool such as Rodin.

**Key words:** Embedded control systems; Modelling method; Event-B patterns; Tool

## 1. Introduction

Engineering complex embedded control systems requires methods and assistance tools. Without dedicated methods their analysis is painful, inefficient and time-consuming. More specifically guidelines and tools are required for formal software engineering. Unlike many other types of software, embedded systems are often developed for specific target environments (processors, vehicles, medical devices, etc.) and very often they should run for long times (even years), once they have been implemented in their so-called critical environments. Therefore, embedded systems and their construction have stringent robustness requirements; accordingly one have to develop them with the sake of reliability at runtime. There are numerous models for embedded real-time systems [9]; moreover the target environments of each embedded system do not help the construction or the expansion of dedicated tools and methods. There are many works dealing with semi-formal models extraction from requirements documents, for example in the scope of the UML notations[10]. There are also many works around the tabular requirement engineering method by Parnas[13]; but we are not aware of any results about the synthesis of Event-B formal models from informal requirments as we are doing in our work.

Considering that *i)* the requirements for reliability and correct construction of the models and the derived embedded systems are important concerns, and that *ii)* the development of these systems still lacks of methods to guide the developers, we are motivated to contribute to fill the gap between these needs and the state of the art. Many researchers underline the need of methods, techniques and tools to support formal modeling and more especially for the Event-B approach which addresses the full software development process: [8,3,12,23]. We have proposed in [4] a correct-by-construction method (named Heñcher) dedicated to the construction of critical embedded control systems. This method, based on Event-B, is intended to guide step by step the specifier or the engineer to drive its development from requirements to concrete software, defining abstract models, and refining them in a systematic way. The current article is an extension of that previous work. We extend that work in two main directions: *i)* we extend the proposed method with an assistance tool dedicated to help the users to apply the Heñcher method step by step and to build quickly the preliminary Event-B models of her/his control systems. The tool is designed as a companion tool of the already existing Event-B frameworks such as Rodin or Atelier B for which we provide input models; *ii)* the Event-B models of the case study are now totally proved using the Rodin tool.

We present in this article the main components and the background of the tool; it is designed as an extensible standalone tool that should be compatible and integrable with Event-B framework. The article is completely reshaped compared to [4] where more space was given to the method and less to its illustration. Here we emphasise the application of the method on the Landing Gear System case study.

The article is structured as follows. In Section 2 we introduce the proposed method through its main steps. In Section 3 we present a benchmark case study, the *Landing Gear System*, we illustrate the detailed application of our method on this case study, and we emphasise issues on tool requirements. Section 4 is devoted to the proposed assistance tool through its main components and the provided facilities; and its use to illustrate parts of the case study. In Section 5 we draws some conclusions of this work.

## 2. A Glimpse of the Heñcher Method

In [4], we presented a stepwise and systematic method (named Heñcher) to construct critical embedded control systems using Event-B. Complex systems can be constructed by combining (see [6,7,1]) two classical approaches: 1) horizontal refinement with *feature augmentation* where we have to build a global abstract model of a the whole system (a controller and its physical environment) and 2) *structural refinement* (making the abstract structures more and more concrete).

The high-level state space of any control system can be described by the **elicitation of the interface variables** between the digital part (the controller) and the physical part (the controlled environment) of the considered system. Fig. 1 depicts a general principle that may govern the organisation of *event-based models* of control systems. The dashed ovals are representative of the parametric events families; They should be replaced by the effective events related to the logic of a specific case study.

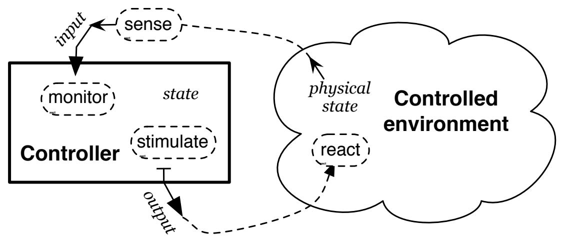


**Fig. 1.** A generic shape for event-based model of a control system

Besides, the identified physical devices to be controlled should be precisely listed. The behaviour of each one will be specified later.

We summarise here the Heñcher method (detailed in [4]); it starts from this interface and comprises six steps for guiding the modelling and analysis of the target system. These steps spawn the two classical Event-B structuring approaches. The horizontal process is made of 3 steps and the vertical process is made of at least 2 steps (steps 4 and 5).
**Step 1:** Characterising the abstract model of a considered system.

- *Step 1.1: Elicit the controller interface* made of three categories of variables describing the controlled environment (input from sensors, output to actuators, and state for monitoring). Additional *internal* variables represent information inside the controller.
- *Step 1.2: Elicit the global properties of the system*: system properties, including safety, liveness and non-functional properties.
- *Step 1.3: Start with a first abstract model* and build a first Event-B abstract model.
- *Step 1.4: List the events of the abstract model*: *sense events*, *monitor events* and *stimulate events*. These families of events are in compliance with the standard sense-decision-control of any control system's cycle. Additionally the behaviour of the physical part is described with the *reaction events family*.

**Step 2:** Extension of the previous abstract model using *feature augmentation* [6,7,1] to integrate the controlled environment on the basis of the *sense events* family.

- *Step 2.1:* Introduce the physical environment and the *reaction events* family.
- *Step 2.2:* Detail the *sense events* family.

**Step 3:** Integration of the specific properties. Considering the requirements of the system, additional specific properties are added to the global model to constrain the functioning of the system. They may be are *reachability properties* or *non-functional properties*.
**Step 4:** Structural refinement of the global abstract model. New internal B events may be added to refine the events of each family of events (*sense*, *monitor*, or *stimulate*). The state space variables may be refined with more details in the invariant.
The model is more refined with the behaviour of the physical part (made of the controlled devices); this is captured through the *reaction events* family.
**Step 5:** Decomposition into software and physical parts. We adopt the A-style decomposition [2]. The methodological guide to achieve the decomposition is as follows: the digital part is made with all the events defined in the *sense events*, the *monitor events* and the *stimulate events* families whereas the physical environment gathers all the events defined in the *reaction events* families. Each part must have an abstract view of the other.
**Step 6:** Refinement of the control software and the physical environment. In addition to classical but complete refinements in Event-B, we propose some guidelines to assist the user in applying there refinements steps.

- *Step 6.1:* Refining the control software. *Structural refinements* based on the *monitor* events family should be used to refine the controller. The involved categories of variables are the *input* variables, the *state* variables and the *output* variables.
- *Step 6.2:* Refining the controlled (physical) environment. Many cases can be considered depending on the system to be studied; either the physical devices are already available, or one has to build the physical devices from the formal models, or one has to build a part of the physical devices.

**Proposed modelling patterns** When there are sub-modules, the *input* variables may be spawned inside the sub-modules.

In the same way *output* variables may be updated by promotion from the sub-modules if any. Therefore one have to incorporate successively in the Event-B model the events to set and modify the *output* variables; they describe the result of the behaviour of the control part. State automata help to catch these behaviours; then the events of the B models encode the automata.

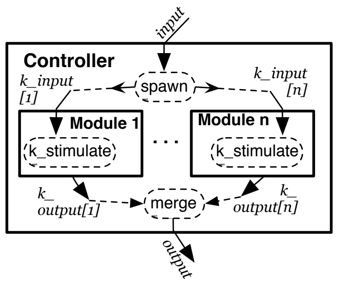


**Fig. 2.** Modules redundancy

We give now some recurrent patterns to help in modelling the control part.

*i*). *Composition of several redundant sub-modules:* when a controller is made of several redundant modules, it is straightforward to describe a generic module and use an indexing function to compose several instances of such modules (see Fig. 2).
   - *Encasing variables inside modules:* the values coming from outside one or several modules can be systematically encased inside the modules with a dedicated event that spawn the events.
   - *Promoting variables outside a module:* in a symmetric way, the values going outside a module or several modules can be systematically described using a **promotion pattern** (with a dedicated event) for merging the output variables of the internal computing modules.

*ii*). When the modules are not redundant, each one should be refined separately, but the treatment we have described for the inputs and outputs variables is the same.

Fig. 3 illustrates the Event-B patterns from the most abstract model (which describes only the interface of the controller) to the systematic decomposition into two parts: the Controller and the physical Environment.

## 3. Applying the Method to the Running Case Study

The proposed method is applied on the *Landing Gear* (LG) case study, a benchmarking example proposed at the ABZ'2014 conference to compare different formal methods in terms of expressivity, performance, and ease of use. in [5].

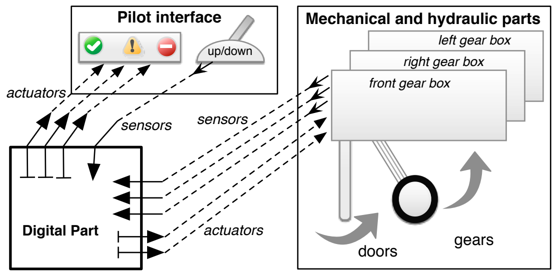


**Fig. 4.** Global architecture of the LG system

A prerequisite for reading this section is the detailed specification of this critical embedded system given A summary of the LG system is depicted in Fig. 4. The LG system is in charge of manoeuvring 3 landing boxes: front, left and right. Each landing box contains a landing gear, an associated door and the corresponding hydraulic cylinders in charge to move gears and doors.

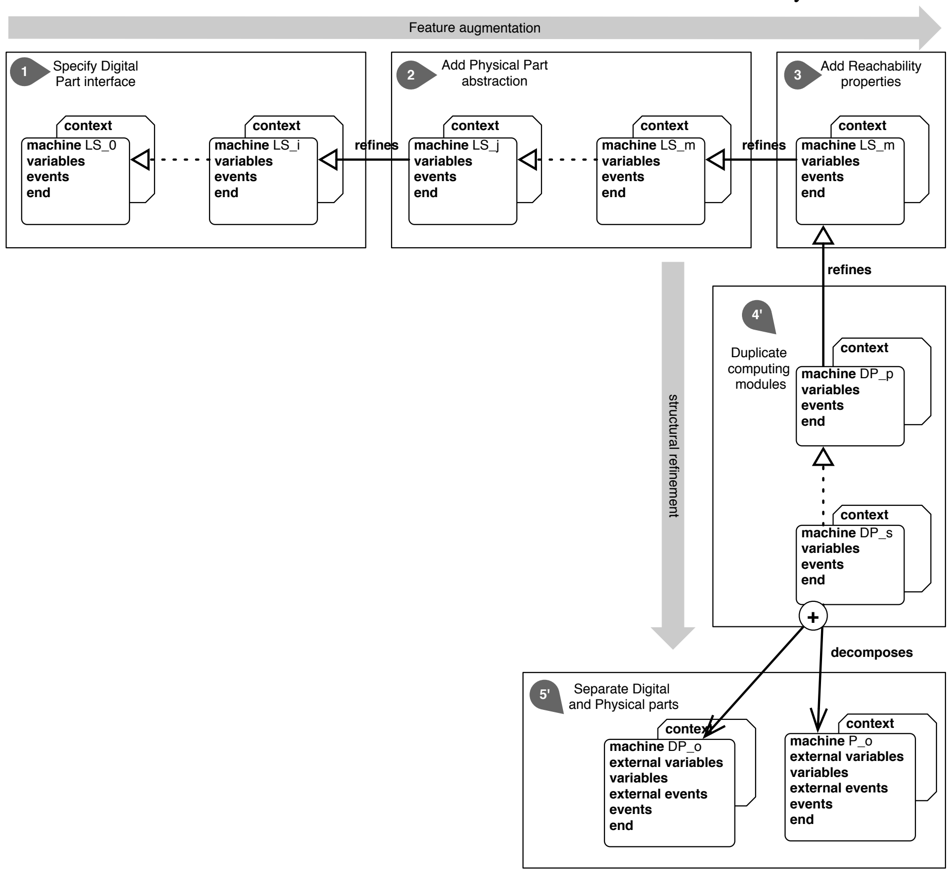


**Fig. 3.** Synoptic structure of the Heñcher method

The system is made of a controller (the *digital part*) and the controlled physical environment (i.e. the 3 landing gear boxes and a pilot interface) which interact via sensors and actuators.

The sensors provide to the digital part the information on the state of its physical part; the actuators engage the orders of the controller on the physical part. The physical devices already exist, we will not build them; the challenge deals with the digital control part only (see page 2 of [5]). We give the main elements resulting from the successive application of the steps proposed in the method (Sect. 2).

## 3.1. Horizontal Process: Building an Abstract Global Model of the System

The document [5] is helpful to identify the different variables at the interface between the digital part and the physical part.

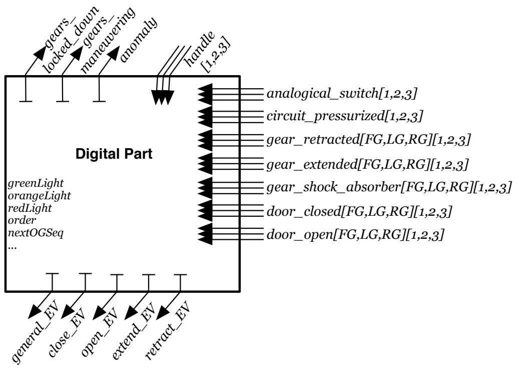


**Fig. 5.** The interface of the digital part

*Step 1: Characterising the abstract model*

*Step 1.1: Elicitation and modelling of the interface variables* The requirement document listed several triplicated *input* variables: handle, analogical switch, gear states, doors states$\cdots$
We model them with a type $TRIPLE = \{1,2,3\}$ used as an index of the function variables (see *Step 1.3*):

| | | | |
|---|---|---|---|
| $GEAR$ | $= \{FG, LG, RG\}$ | $analogical\_switch$ | $\in TRIPLE \rightarrow AnalSWSTATE$ |
| $DOOR$ | $= \{FD, RD, LD\}$ | $handle$ | $\in TRIPLE \rightarrow HSTATE$ |
| $HSTATE$ | $= \{hDown, hUp\}$ | $gear\_extended$ | $\in (TRIPLE \times GEAR) \rightarrow BOOL$ |
| $AnalSWSTATE$ | $= \{openSW, closedSW\}$ | $door\_closed$ | $\in (TRIPLE \times DOOR) \rightarrow BOOL$ |
| $\cdots$ | | $\cdots$ | |
| $handle$ | $\in TRIPLE \rightarrow HSTATE$ | $analogical\_switch$ | $\in TRIPLE \rightarrow AnalSWSTATE$ |
| $gear\_extended$ | $\in (TRIPLE \times GEAR) \rightarrow BOOL$ | $door\_closed$ | $\in (TRIPLE \times DOOR) \rightarrow BOOL$ |

The function variable $handle \in TRIPLE \rightarrow HSTATE$ captures precisely the requirement $handle_i \in \{hDown, hUp\}$ with $i \in \{1,2,3\}$. The *state* variables are the states of the gears, doors, anomalies, etc. They are modelled as follows:

$$gears\_locked\_down \in BOOL \ \wedge \ gears\_maneuvering \in BOOL \ \wedge \ anomaly \in BOOL \wedge \cdots$$

The *output* variables hold the values computed for various electro-valves:

$$general\_EV \in BOOL \ \wedge \ close\_EV \in BOOL \ \wedge \ open\_EV \in BOOL \wedge \cdots$$

The lights which indicate the position of the gears and doors to the pilot are described as *internal* variables: *greenLight*, *orangeLight*, *redLight*. These variables are bound to the output state variable *gears_locked_down* with an invariant predicate. Another *internal* variable *order* is used to record the action of the pilot on the handle.

The LG system is controlled digitally in the *normal* mode until an anomaly is detected. A permanent failure leads to an *emergency* mode where the system is controlled analogically. Accordingly the *internal* boolean variable *anomaly* is used to denote that an anomaly has been detected or not.

*Step 1.2: Elicitation of the global properties of the LG system* Most of the normal mode requirements are safety properties. Some identified ones are gathered in Table 1.

*Step 1.3: Start with a first abstract model* The first Event-B abstract model resulting from *Step 1.3*, gathers all the variables of the interface, their related invariants and initialisations. Event-B contexts are used to model the static part with the various sets and definitions that we have introduced.

| | |
|---|---|
| $R_{21}$ | We can not observe a retraction sequence (consequence of the order $hUp$) if the handle is down. Using the enumerated set $HSTATE$ which permits only one value from two for the variable $order$. |
| $R_{31}$ | The gears outgoing event occurs if doors are open locked. |
| $R_{41}$ | Opening and closing doors electro-valve are not stimulated simultaneously. |
| $R_{51}$ | It is not possible to stimulate the manoeuvring EV (opening, closure, outgoing or retraction) without stimulating the general EV. |

**Table 1.** Identified requirements

*Step 1.4: The families of events of the abstract model* A thorough analysis of the two action sequences (*outgoing sequence* and *retraction sequence*) described in the LG system helps us to capture the behaviour of the digital part and to derive the events. We use here state automata to make it clear the interaction between the different components (actions of the pilot, the controller, the orders received by the environment).

In the *sense event* family we have listed for example the event sense_gear to modify the input variable *gear_extended* listed above. In the same way, we have listed the other events sense_door, etc. Examples of events we have identified for the *control events* family are: stmlt_general_EV to stimulate the general electro valve, stmlt_door_opening, stmlt_gear_outgoing, stop_stmlt_general_EV, stop_ stmlt_gear_outgoing, etc. Each one modifies its related variable, for instance the event stop_stmlt_gear_outgoing sets the variable $extend\_EV$ to $FALSE$. Examples of events we classified in the *monitor events* family are: monitor_ anomaly, monitor_gears_locked_Down, monitor_ gears_manoeuvring. In the *reaction event* family we have Door_openDoor_cl2cu, Gear_extend, Gear_retract, ...

*Step 2: Extension of the abstract global model with the event families*

We achieve many refinement steps, by feature augmentation, to integrate gradually the variables and events related to the physical devices: the sensors, the doors and the gears.

Following *Step 2.1*, we define the behaviours of physical devices. For instance, the door behaviour is first captured with a state automata; the transitions of the automata are then described as events. For this purpose we use a transition function $doorState \in DOOR \to DSTATE$ where $DSTATE = \{ClosedLocked, ClosedUnlocked, OpenUnlocked\}$ is the enumerated set of the identified door states. The set $DOOR$ contains the three door identifiers. The function $doorState$ is a total function; this captures the requirement that all the three doors are controlled via the state transition.

The starting transition of the door behaviour is enabled by the $open\_EV$ order given by the digital part. Therefore there is a synchronisation between the digital part and the motion of the doors. We only give below the description of the starting event Door_openDoor_-cl2cu; the other necessary events are similar.

```
event Door_openDoor_cl2cu
/* Door's Behaviour (for the three doors). The first transition of the Door Automata */
where
    @g1  $open\_EV = TRUE$ // all the doors Electro Valves are on
    @g2  $ran(doorState) = \{notOpenLocked\}$
then
    @a1  $doorState := DOOR \times \{notOpenNotLocked\}$ // door is being opened
end
```

The following event describes an event of the *control event* family.

```
event stmlt_gear_outgoing
/* stimulate gear outgoing electro valve once the three doors are in the open position */
where
    @g0  $general\_EV = TRUE$
    @g1  $order = hDown$
    @g2  $ran(handle) = \{hDown\}$
    @g4  $ran(door\_open) = \{TRUE\}$
    @next  $nextOGseq = 3$
    @gano  $anomaly = FALSE$ // no anomaly detected
    @notretract  $retract\_EV = FALSE$
then
    @a1  $extend\_EV := TRUE$
    @a2  $nextOGseq := nextOGseq + sequenceStep$
end
```

The variable *nextOGseq* controls the evolution of the outgoing sequence; it indicates in the event guards, the next step in the outgoing sequence. We note that the events in the *sense event* family anticipate their real future specifications, which are related to the physical part introduced later. When we have introduced the various events families and the related variables, it becomes clear for us that we have the complete control loop. Following *Step 2* the properties (listed in *Step 1.2* above) are formalised as first order predicates, integrated into the invariant of the abstract model and, proved along the horizontal refinement. As an example, the requirement $R_{51}$ is described as follows.

$$((open\_EV = TRUE \lor close\_EV = TRUE \lor extend\_EV = TRUE \lor retract\_EV = TRUE) \\ \Rightarrow general\_EV = TRUE)$$

To sum up, the global Event-B abstract model results from a series of refinement of contexts and machines.

*Step 3: Dealing with specific properties* The properties to be proved (requirements given in pages 18-19 of the requirement document) are formalised as first order predicates integrated into the invariant of the abstract model and proved along the horizontal refinement. Most of the normal mode requirements are safety properties. Here are some of the requirements captured in our case study: $R_{22}$, $R_{32}$, $R_{42}$, $R_5$.

| | |
|---|---|
| $R_{22}$ | In a similar way we cannot observe an outgoing sequence (consequence of the order *hDown*) if the handle is up. |
| $order = hUp \Rightarrow ran(handle) \neq \{hDown\}$ | |

| $R_{32}$ | The gears retraction event occurs if doors are open locked |
|---|---|
| | $(retract\_EV = TRUE \Rightarrow ran(door\_open) = \{TRUE\})$ |

| $R_{42}$ | Outgoing and retraction gears electro-valve are not stimulated simultaneously |
|---|---|
| | $\neg(extend\_EV = TRUE \wedge retract\_EV = TRUE)$ |

| $R_{51}$ | It is not possible to stimulate the manoeuvring EV (opening, closure, outgoing or retraction) without stimulating the general EV |
|---|---|
| | $((open\_EV = TRUE \vee close\_EV = TRUE \vee extend\_EV = TRUE \vee retract\_EV = TRUE) \Rightarrow general\_EV = TRUE)$ |

In this case study, reachability is another set of the specific properties. Requirement $R_1$ for instance needs a specific treatment presented in the sequel. We will detail this point in Section 3.3.

## 3.2. Vertical Process: Building the Concrete Parts of the LG System

The vertical process includes several refinements (in *Step 4*) described below following the proposed method.

*Step 4: Structural refinements of the global abstract model*

In the requirement document, the inner structure of the digital part is made of two redundant computing modules. Structural refinement steps overcome the details of the behaviour of the digital part.

*a) Introducing the two computing modules with refinements* Both modules have the same interface (*input* and *output* variables) inherited from the abstract model of the digital part. Each interface variable of a module $k$ (where $k \in \{1,2\}$) is inherited from a variable (for instance *gear_extended*) of the digital part of the abstract model and it is denoted by $k\_gear\_extended(k)$ where $k$ is an index. An enumerated set $CompModule = \{1,2\}$ is used for the indexes. Therefore each interface variable of the computing modules is specified with the following shape:

$$k\_gear\_extended \in CompModule \to ((TRIPLE \times GEAR) \to BOOL)$$

The binding between the two modules interface variables and those of the abstract module is achieved via refinements where new variables and related events are introduced.

*b) Spawning the inputs inside the computing modules with refinements* We introduced new events (prefixed with `spawn_`) to push the value of each *input* variable (for example *handle*) at the abstract level, in the corresponding variable (for example *k_handle*) of each computing module. As the inputs of the modules should be the same, an invariant is defined in each case of variable spawning in order to guarantee the correctness of the binding between the *input* variable of the digital part and the same input of the computing modules. The following event pattern spawns the variables at the interfaces of the computing modules.

```
event spawn_handleDown // spawn handleDown within the k CompModules
where @g1   $ran(handle) = \{hDown\}$
then
    @a1   $k\_handle := \{1 \mapsto (TRIPLE \times (ran(handle))), 2 \mapsto (TRIPLE \times (ran(handle)))\}$
end
```

We have identified a reusable **specification rule**: a new event is introduced along with each new k-indexed variable. This event should copy the variable at high level (the digital part) into the indexed variables at the low level. Furthermore, the existing events, whose guards or actions involve the spawned variables, should be refined by extending their guards and actions in order to satisfy the binding between the variables and the associated k-indexed variables. One noticeable feature in this case is that when we have a non-deterministic event of abstract level (as for the value of the sensors), then in the refinement the event should be refined (not extended). This is another reusable **specification rule** we have identified.

*c) Merging the outputs of the computing modules with refinements* As depicted in Fig. 2, the k-indexed *output* variables are merged using a logical OR to set the corresponding variable at the output of the digital part. Therefore the event that sets the variable should be guarded by the availability of the merged value. As explained before, a binding invariant should be provided for each variable and the related k-indexed variable. Several refinements are used to introduce the appropriate events.

*d) Specifying the behaviour of the computing modules* The two computing modules have the same behaviour which is made of: the events that monitor the system and set accordingly the state output variables and the input variables of the digital part; and the events that give orders (control decision) to the physical part through the order output variables. This results in the k-indexed form of the events related to the three categories of the interface and internal variables.

We can stop the construction of the global model at this stage; however following the guidelines provided in the method, it remains to perform the decomposition step in the basis of the *sense*, *monitor*, *control* events families (*Step 6*). Fortunately, the decomposition modules of Rodin provide assistance for this purpose. In our case where the event families structured the model, the Abrial's style of decompostion which is based on share variables [2] is the most appropriate. Indeed, the decomposition is precisely based on the families of events: the *reaction* family should be used for a (physical) machine while *sense, monitor* and *control* families should be used for another (software) machine.

As far as *Step 5* and *Step 6* are concerned, there are two main considerations: *i)* if we want to use animation capabilities on the global model, the construction should stop after the refinements of *Step 5* without doing the decomposition of the *Step 6*; *ii)* if we do not want to use animation capabilities, the construction may be continued with the decomposition process in *Step 6.* For our illustration of the case study we experiment with both considerations. First, in order to keep animation capabilities, we end our process with the *Step 5*; the *Step 6* was not performed for the case study, but only the digital part is refined with the objective to build the software part; The variables and events which are specific to the behaviour of the physical part are not refined but we keep them in the model in order to perform animation of the global model. Second, for the experimentation of the method, we go through decomposition in *Step 6.* But in this case, we have two independent models which should evolve separately, for instance the physical part may be replaced by a hardware and the control part refined into an executable code without modifying any elements of the physical part inherited from the decomposition.

## 3.3. Handling the Required Properties

We classified the requirements listed in the case study document (see page 18-19 of the requirement document) in several categories of properties to be proved for the system.

**Safety:** Requirements $R_2$, $R_3$, $R_4$ and $R_5$ should be considered through safety properties.
**Liveness:** The requirements $R_1$ are related to liveness (reachability) properties.
**Nonfunctional:** The requirements $R_6$, $R_7$ and $R_8$ (Failure mode requirements) are related to nonfunctional properties: management of time constraints.

In the following we deal with liveness (reachability) properties.

**Introducing the reachability property (requirement $R_{21}$ and $R_{22}$)** This is a step of the horizontal refinement process (with the tag $\textcircled{3}$ in Fig. 3).
Based on the idea of Lamport's logical clocks [11], we implement a technique that captures the reachability requirement $R_1$ given in page 13 of the requirement document. For that purpose, we introduce the notion of *control cycle*; this is necessary to reason locally on relevant events. A *control cycle* is a period of time during which one can observe several events, especially a chain of events denoting an outgoing sequence or a retraction sequence; a typical control cycle is one starting with an event which denotes the *hDown* order and terminating by an event which denotes the fact that "*the gears are locked down and the doors are seen closed*"; similarly, another control cycle is started when the handle triggers an order *hUp*. A dedicated variable *endCycle* is used to control the start and the end of each control cycle.

Assume that we have observable events that occur along the time and that denote our events of interest[1]; for instance the starting of an outgoing sequence, a door closed, a gear locked in a position, etc. Each such event can be stamped with the timestamp of its occurrence, thus if we have the set of observed events we can define at least a partial ordering of these events (see Fig. 6). Given a set *obsEvents* of events and a logical

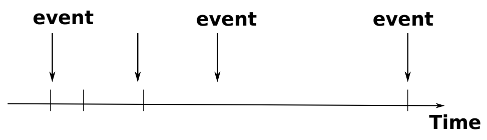


**Fig. 6.** Events and timestamps

clock modelled as a natural number, the occurrences of the events can be ordered by the timestamp given by the clock. In our case two events cannot happen at the same time. We use a partial function $ldate \in obsEvents \rightarrow\!\!\!\!\!+ \mathbb{N}$ to record the timestamps of the events. We can compare and reason on the timestamps of any events happening during a sequence and specifically within the specific event sequence called *control cycle*.

For example, in the normal mode, we observe the event "*the door is closed and the gear extended*" (named dcge) at the end of a cycle, if the event "*order DOWN is given*" (named downH) occurs and is maintained (no event upH occurs). If these events have

[1] These events are not to be confused with Event-B events.

respectively the specific timestamps $dj$ and $di$, then we can compare $di$ and $dj$ and also examine the events which happen between $di$ and $dj$. Accordingly the property $R_{1bis}$ of the requirement is expressed as follows:

$$\boxed{\begin{array}{l} \forall dj.(((dj \in \mathbb{N}) \wedge (dcge \in dom(ldate)) \wedge (dj = ldate(dcge)) \\ \qquad \wedge (endCycle = TRUE) \wedge dj < llc) \Rightarrow \\ \quad \exists di.((dd \in \mathbb{N}) \wedge (downH \in dom(ldate)) \wedge (di = ldate(downH)) \wedge (di < dj) \wedge \\ \qquad \quad \forall ii.(ii \in \mathbb{N} \wedge di \leq ii \wedge ii < dj \Rightarrow ldate \sim [\{ii\}] \neq \{upH\}))) \end{array}}$$

The above property expresses that if we reach the end of a control cycle where the door is closed and the gear extended at a given timestamp ($dj$), then we should have an order *hDown* issued at a timestamp $di$ less that $dj$ and maintained between $dj$ and $di$; the outgoing sequence is not interrupted by an order $hUp$ which would start another cycle. Consequently we have expressed the property $R_{11bis}$. Property $R_{12bis}$ can be expressed in a similar manner.

To put in practice in Event-B with Rodin, we defined the set *obsEvents* in the context of our machines, and the above property is included in the invariant of the abstract model.

## 3.4. Experimentation with Rodin and statistics

The main modelling steps of the Landing Gear System case study have been completely achieved; that is the modelling from very abstract level to more concrete ones, the refinements and the decomposition into hardware and software parts. Applying a rigorous method as we defined, was very helpful to master the complexity of the case study.

The Rodin tool is very efficient for proving the Event-B models; a very high percentage ($\sim 87\%$) of proof obligations was automatically discharged. All the remaining proof obligations are proved interactively.

| | Total | Auto | Manual | Review. | Undis. |
|---|---|---|---|---|---|
| LandingSys5 | 567 | 494 | 73 | 0 | 0 |
| **Abstract model** | | | | | |
| Landing_DP_Ctx | 0 | 0 | 0 | 0 | 0 |
| LandingSysDP_A | 109 | 106 | 3 | 0 | 0 |
| LandingSysDP_SWITCH_A | 3 | 3 | 0 | 0 | 0 |
| LandingSysDP_DOOR_A | 42 | 42 | 0 | 0 | 0 |
| LandingSysDP_DOOR_GEAR_A | 79 | 79 | 0 | 0 | 0 |
| LandingSysDP_DOOR_GEAR_TIME_A | 2 | 2 | 0 | 0 | 0 |
| **Models of the vertical refinement** | | | | | |
| LandingSysDP_DGT_R1_In | 42 | 34 | 8 | 0 | 0 |
| LandingSysDP_DGT_R2_INOUT | 56 | 40 | 16 | 0 | 0 |
| LandingSysDP_DGT_R3_DG | 234 | 188 | 46 | 0 | 0 |

**Table 2.** Statistics of PO generated and proved with Rodin

The specifications are available online[2]. The current version of the Event-B models is deliberately partial as we chose to focus on representative events instead of being exhaustive. We have used the version 3.4 of Rodin in the last experimentations; the statistics on Proof Obligations are given in Table 2.

The proofs discharged using the interactive prover are related to the structural refinement and specifically they are related to the binding invariants.

[2] hencher.ls2n.fr

Using the Rodin tool we have modelled and refined the Landing Gear System until to take account of the main requirements about software part, physical part and some of the specific properties as explained in Section 3.3. After several steps of vertical refinements we have a complete model of the Landing Gear system. For the experimentation purpose, using the Event-B decomposition technique [22], we decompose the last model of the system into two parts corresponding to the hardware part and the software or control part. The so-called A-style decomposition, based on the separation of events through different machines, and implemented as a Rodin plugin [15], was successfully used in this step.

Managing very large models requires a rigorous slicing and several small steps of refinements. This is the reason why we have introduced many refinements, but it is still not enough, the slicing can be of finer grain.

Moreover a good naming discipline is necessary at each level of the modelling. It helps for traceability and to face the complexity due to the size of the model.

As far as the ProB animation tool (integrated in Rodin) is concerned, it is very helpful to tune the Event-B models; indeed the failure in the animation gives information about the (bad) states and accordingly the wring part of the model can be rewritten.

## 3.5. Tooling Concerns

During the above experimentations, we felt need assistance at different steps for different motivations. We mention some situations where tooling would be helpful, in addition to Rodin's facilities, to apply the Heñcher method.

- $RT_1$ *Starting the process.* The first steps are often crucial when applying a method. Assistance is required to answer the users's question *How to start the process?*.
- $RT_2$ *Incremental step-by-step refinement.* The Heñcher method is tightly based on refinements. To master the development process in Event-B, the recommended approach is to proceed with small and well-defined refinement steps; that means the complexity is not in individual refinements but in the whole refinement process; but there is a lack of assistance tool to help developers. For instance the developer may be happy with a highlighting of some parts of its specifications.
- $RT_3$ *Pattern-based substitutions and automatic refinements* for composition and physical part refinement. This requirement needs a full development when the specifications are in Rodin.
- $RT_4$ *Team collaboration.* Making it easy for several people to work simultaneously on a project, versioning, decision traceability are all important concerns when dealing with big Event-B projects as the one we have studied.
- $RT_5$ *Overall development process management.* When one should stop with a development step? Is the current state sufficient to start the next step? there is a need of various metrics to evaluate the quality of ongoing specifications (completeness,...).
- $RT_6$ *Iterative process of model evolution.* Often, one wants to modify an abstract model (for example adding a variable or an event in the M0 machine), and has the modification be systematically propagated in the chain of the remaining models and refinements. The tool supporting the method should enable such a continuous model evolution.

$RT_7$ *Traceability of the refinement chain of a single event.* During the development and proof steps, a practical concern is to review the chain of refinement of a single event without all the surrounding events. An assistance tool is also needed.

$RT_8$ *Securing copy-paste operations.* It is often the case, to copy-paste similar events. This is particularly true when we have systems with redundancies of several instance of the same objects, like the gears, the doors and the sensors in our case study. Due to unavoidable human errors, lot of time is spent fighting again undischarged proof obligations. There is a need of a parameterized refactory tool that for instance rewrites an existing event by substituting some variables with others.

We undertake the development of an assistance tool to provide the main functionalities we have identified during experimentations. In the next section we introduce the basis of the design of a web companion tool we have developed to address the requirements $RT_1$ and $RT_2$.

## 4. An Assistance Tool

In this section we introduce our proposed prototype tool to assist the users of the Heñcher method. The tool is designed as a companion tool of the existing frameworks such as Rodin or Atelier B. It helps the users to apply the Heñcher method, and to build more quickly the preliminary Event-B models, which will be analysed in the dedicated existing environments.

From the provided interface of a given control system the tool generates in an incremental way, following the steps of the Heñcher method, the Event-B abstract models. The tool is designed for Event-B users (specifiers/engineers). The inputs of the tool will be provided by the users; an user-friendly graphical interface is designed for this purpose. The tool provides the development guidelines of the Heñcher method and some skeletons of Event-B models as output.

**Generation of an Event-B models from a control system interface** On the basis of the interface between a control system and its environment as presented in Section 2, a specifier should provide the interface variables, the list of controlled devices of its system, the global and specific properties required by its systems. From these elements the specifier will be assisted in building an Event-B machine $M_0$ and then a refined one $M_1$. The machine $M_1$ can be further refined until more concrete levels but one has to use the dedicated tool (Rodin for instance).

**The collected interface** Let an interface made of $\mathcal{X}_s$ a set of the input variables, $\mathcal{X}_o$ a set of the output variables, $\mathcal{X}_c$ a set of the control variables. In addition, let $\mathcal{X}_i$ be a set of internal variables of the controller; (a part of the control variables are used for the feedback, the internal variables are the part of the control variables used by the controller but which are not output as feedback). According to the system at hand, the user should define the types of each of the previous variables. That means each variable of $\mathcal{X}_s$ has its type in $\mathcal{T}_s$. Therefore for each family $X$ of variables of the interface and the related set of types $T_X$, we have a type mapping $\{(\sigma_i, \tau_i)\} \subseteq X \times T_X$.

Assume then three type mappings built by the user from its system requirements and from the variables $\mathcal{X}_s$, $\mathcal{X}_o$, $\mathcal{X}_c$ and the related sets of types $T_{X_s}$, $T_{X_o}$, $T_{X_c}$.

## 4.1. An Overview and the Design of the Tool Assistant

The flowchart in Fig. 7 gives an overview of how the tool assists the user in building the abstract model in a global process. We will then define the steps of this global process.

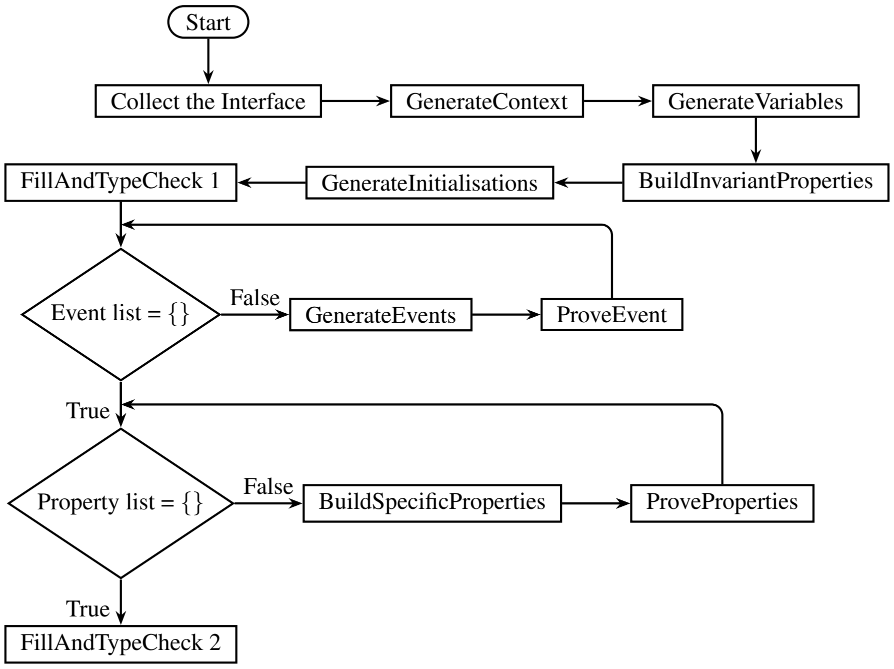


**Fig. 7.** Flowchart of the global functioning of the tool

### Stepwise building of the Event-B model

*Building the* context *of the model.* From the extracted sets of types $T_{X_s}$, $T_{X_o}$, $T_{X_c}$, a context Ctx0 made of the sets coming from $T_{X_s}$, $T_{X_o}$, $T_{X_c}$ is built. Each carrier set of Ctx0 comes from $T_{X_s} \cup T_{X_o} \cup T_{X_c}$ (see Fig. 8).

context $Ctx0$
constants
    to be completed by the specifier
sets
    `Each element of the union` $T_{X_s} \cup T_{X_o} \cup T_{X_c}$
axioms
    to be completed by the specifier
end

**Fig. 8.** Event-B context skeleton

*The initial Event-B abstract model (Step 1 of the method).* The skeleton of the Event-B abstract model ($M_0$) to build is depicted in Fig. 9.

```
Machine $M_0$
SEES Ctx0
VARIABLES
    GenerateVariables$\mathcal{X}_s \cup \mathcal{X}_o \cup \mathcal{X}_c \cup \mathcal{X}_i$
INVARIANTS
    BuildInvariantProperties$(\mathcal{X}_s, \mathcal{X}_o, \mathcal{X}_c, \mathcal{X}_i)$
    BuildSpecificProperties$(\mathcal{X}_s, \mathcal{X}_o, \mathcal{X}_c, \mathcal{X}_i, \mathcal{P}_s, \mathcal{P}_n, \mathcal{P}_l)$
INITIALISATION
    GenerateInitialisations$(\mathcal{X}_s, \mathcal{X}_o, \mathcal{X}_c, \mathcal{X}_i)$
END
```

**Fig. 9.** Event-B model skeleton

The functions GenerateVariables, BuildInvariantProperties, BuildSpecificProperties and GenerateInitialisationSubstitutions are used to compute respectively the set of variables, typing invariants, specific invariants and default initialisations for the $M_O$ model. They are elementary functions; most of them traverse a set, and encode in Event-B syntax the elements of the sets.

The function GenerateInitialisations$(\mathcal{X}_s, \mathcal{X}_o, \mathcal{X}_c, \mathcal{X}_i)$ works as follows: for each variable $v$ in $\mathcal{X}_s \cup \mathcal{X}_o \cup \mathcal{X}_c \cup \mathcal{X}_i$, if $T_v$ is a set in Ctx0 corresponding to the type of $v$ then a substitution `v ::` $T_v$ is generated. Note that these default initialisations can be modified by the users to meet its needs.

*The global properties of the system.* Considering the requirements of the given system let $\mathcal{P}_s$ be the set of the safety properties, $\mathcal{P}_n$ be the set of non-functional properties, and $\mathcal{P}_l$ be the set of liveness properties. Each property should be formalised by the user and incorporated with the assistance tool in the model under construction.

## Assistance in horizontal refinement steps

*Construction of the events of the abstract model (Step 2 of the method).* The current Event-B abstract model is now extended (that means feature augmentation) with the previous family of events (sensing events, monitoring events, control events).

Let $\mathcal{E}_s$ be the set of sensing events, $\mathcal{E}_m$ be the set of monitoring events, $\mathcal{E}_c$ be the set of control events, $\mathcal{E}_a$ be a set of reaction events.

The algorithm of GenerateEvents$(\mathcal{E}_s, \mathcal{E}_m, \mathcal{E}_c)$ is listed in Fig. 11s:

```
Machine $M_0$
. . .
EVENTS
    GenerateEvents$(\mathcal{E}_s, \mathcal{E}_m, \mathcal{E}_c)$
END
```

**Fig. 10.** Event-B model $M_0$

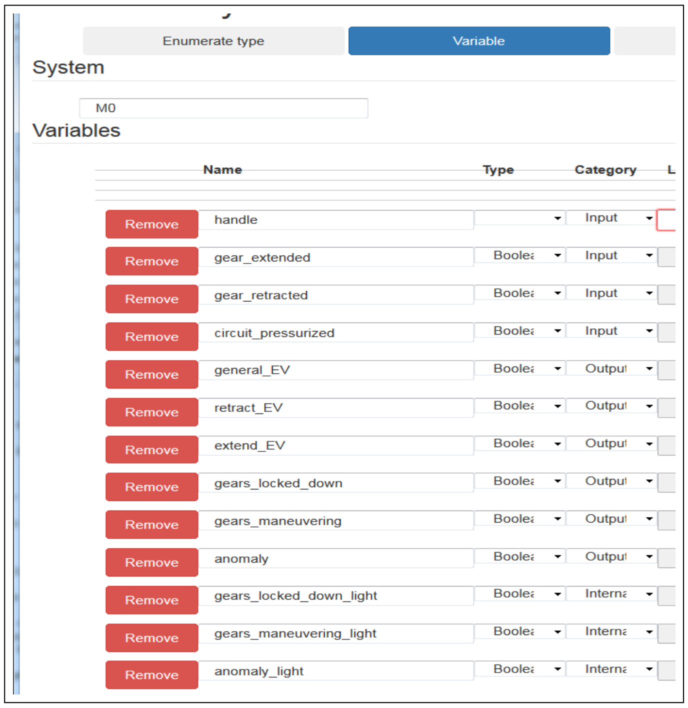
```
while $(\mathcal{E}_s \neq \varnothing) \wedge (\mathcal{E}_m \neq \varnothing) \wedge (\mathcal{E}_c \neq \varnothing)$ do
    Select an event $e$ from $\mathcal{E}_s$ and update $\mathcal{E}_s$ $(E_s = E_s - \{e\})$
    or
    Select an event $e$ from $\mathcal{E}_m$ and update $\mathcal{E}_m$
    Select an event $e$ from $\mathcal{E}_c$ and update $\mathcal{E}_c$
    $M_0 = AddEvent(M_0, BuildEvent(e))$
end while
```

**Fig. 11.** Events construction adding algorithm

The function $BuildEvent(e)$ generates a skeleton for each event name $e$. This skeleton should be filled by the user.

The Event-B abstract model resulting from this stage should now be extended with the specific properties identified by the user in the system requirements (they have to be formalised by the user).

We build a web application to implement the starting sequence of the process depicted in Fig. 7. Its application to a part of the case study is shown in Fig. 12 for the description of variables. Other informations on the tool can be found on the dedicated website[3].



System

M0

Variables

|  | Name | Type | Category | L |
|---|---|---|---|---|
| Remove | handle |  | Input |  |
| Remove | gear_extended | Boolea | Input |  |
| Remove | gear_retracted | Boolea | Input |  |
| Remove | circuit_pressurized | Boolea | Input |  |
| Remove | general_EV | Boolea | Output |  |
| Remove | retract_EV | Boolea | Output |  |
| Remove | extend_EV | Boolea | Output |  |
| Remove | gears_locked_down | Boolea | Output |  |
| Remove | gears_maneuvering | Boolea | Output |  |
| Remove | anomaly | Boolea | Output |  |
| Remove | gears_locked_down_light | Boolea | Interna |  |
| Remove | gears_maneuvering_light | Boolea | Interna |  |
| Remove | anomaly_light | Boolea | Interna |  |

**Fig. 12.** Interface of the web assistant to help in describing the abstract state

Adding properties to the model is a refinement process (Step 3 of the method). The informal properties named and described by the user have to be formalised and integrated in the abstract model. The assistance here consists in selecting for formalisation, each property from those listed in $\mathcal{P}_s$by the user (see Fig. 13).

```
while ($\mathcal{P}_s \neq \varnothing$) do
    Select a property $p$ from $\mathcal{P}_s$
    $\mathcal{P}_s = P_s - \{p\}$) // update $\mathcal{P}_s$
    $M_0 = AddProperty(M_0, Formalise(p))$
    Prove $M_0$
end while
```

**Fig. 13.** Properties adding algorithm

The function $Formalise(p)$ enables one to edit (through a popup for instance) each involved property. In the same way, the other specific (reachability and non functional) properties listed in $\mathcal{P}_n$ are integrated into the refined model $M_1$ using the algorithm described in Fig. 14. Assistance is provided to the user for selecting the specific properties to be integrated into the abstract model.

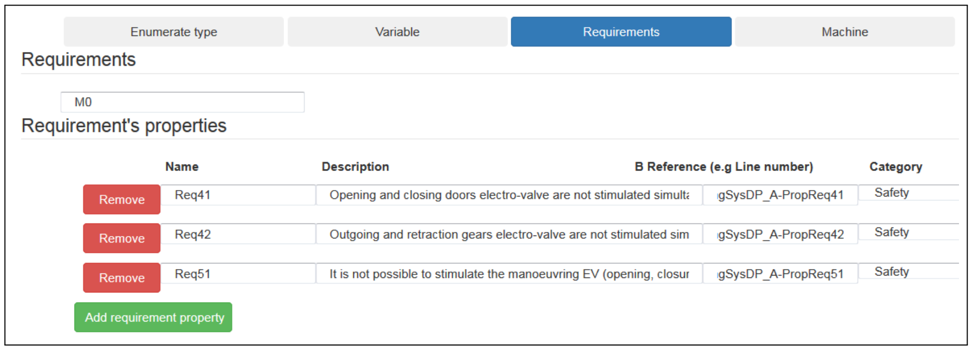
[3] hencher.ls2n.fr

```
$M_1 = M_0$ // initially M0 is copied; then updated.
while ($\mathcal{P}_n \neq \varnothing$) do
    Select a property $p$ from $\mathcal{P}_n$ and update $\mathcal{P}_n$
    $M_1 = Add_p(M_1, formalise(p))$
end while
```

**Fig. 14.** Additional non-functional properties

The function *Formalise(p)* is as previously defined. $\mathcal{P}_n$ is the set of non-functional properties.

Fig. 15 illustrates three of the requirements as listed in the web interface of the assistance tool; they appear in the Rodin snapshot depicted in Fig. 16.



Requirements

M0

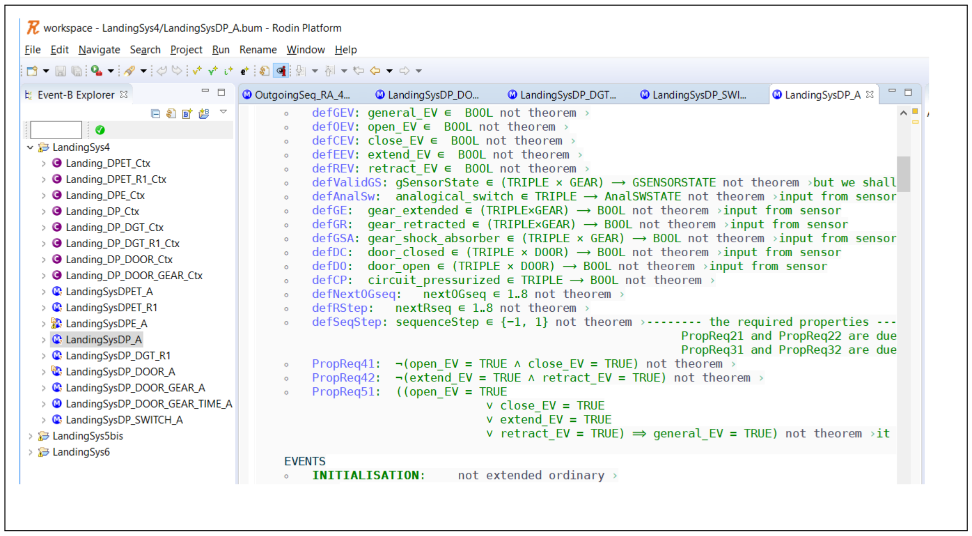
Requirement's properties

| | Name | Description | B Reference (e.g Line number) | Category |
|---|---|---|---|---|
| Remove | Req41 | Opening and closing doors electro-valve are not stimulated simulta | gSysDP_A-PropReq41 | Safety |
| Remove | Req42 | Outgoing and retraction gears electro-valve are not stimulated sim | gSysDP_A-PropReq42 | Safety |
| Remove | Req51 | It is not possible to stimulate the manoeuvring EV (opening, closur | gSysDP_A-PropReq51 | Safety |

Add requirement property

**Fig. 15.** Informal required properties listed in the assistance tool

To improve traceability between, it is better to edit directly the property in Rodin and to use its label to tag the property in the list of the informal properties.



```
  ∘  defGEV: general_EV ∈  BOOL not theorem ›
  ∘  defOEV: open_EV ∈  BOOL not theorem ›
  ∘  defCEV: close_EV ∈  BOOL not theorem ›
  ∘  defEEV: extend_EV ∈  BOOL not theorem ›
  ∘  defREV: retract_EV ∈  BOOL not theorem ›
  ∘  defValidGS: gSensorState ∈ (TRIPLE × GEAR) ⟶ GSENSORSTATE not theorem ›but we shall
  ∘  defAnalSw:  analogical_switch ∈ TRIPLE ⟶ AnalSWSTATE not theorem ›input from sensor
  ∘  defGE:  gear_extended ∈ (TRIPLE×GEAR) ⟶ BOOL not theorem ›input from sensor
  ∘  defGR:  gear_retracted ∈ (TRIPLE×GEAR) ⟶ BOOL not theorem ›input from sensor
  ∘  defGSA: gear_shock_absorber ∈ (TRIPLE × GEAR) ⟶ BOOL not theorem ›input from sensor
  ∘  defDC:  door_closed ∈ (TRIPLE × DOOR) ⟶ BOOL not theorem ›input from sensor
  ∘  defDO:  door_open ∈ (TRIPLE × DOOR) ⟶ BOOL not theorem ›input from sensor
  ∘  defCP:  circuit_pressurized ∈ TRIPLE ⟶ BOOL not theorem ›
  ∘  defNextOGseq:   nextOGseq ∈ 1‥8 not theorem ›
  ∘  defRStep:   nextRseq ∈ 1‥8 not theorem ›
  ∘  defSeqStep: sequenceStep ∈ {−1, 1} not theorem ›-------- the required properties ---
                                                      PropReq21 and PropReq22 are due
                                                      PropReq31 and PropReq32 are due
  ∘  PropReq41:  ¬(open_EV = TRUE ∧ close_EV = TRUE) not theorem ›
  ∘  PropReq42:  ¬(extend_EV = TRUE ∧ retract_EV = TRUE) not theorem ›
  ∘  PropReq51:  ((open_EV = TRUE
                          ∨ close_EV = TRUE
                          ∨ extend_EV = TRUE
                          ∨ retract_EV = TRUE) ⇒ general_EV = TRUE) not theorem ›it

EVENTS
  ∘  INITIALISATION:    not extended ordinary ›
```

**Fig. 16.** Required properties formalised in Rodin interface

The remaining steps of the method, that is the refinement of the control software (**Step 6.1** of the method) and the refinement of the controlled environment (**Step 6.2**) of the methods should be achieved within Rodin (or a dedicated Event-B framework). However we propose some assistance as described in the following section.

**Assistance in the vertical refinement process (*Step 4, 5*)** There is no specific tool for the *Step 4. Refining the global abstract model*. This steps consists in describing the behaviour of the devices involved in the case study. This can be done by using automata or the appropriate models; by encoding these models in Event-B (roughly, the transitions of the automata are encoded as Event-B events).

*Decomposition into software and physical parts (Step 5).* The methodological guide to achieve the decomposition is as follows: the digital part is made with all the events defined in the *sense events* ($\mathcal{E}_s$), the *monitor events* ($\mathcal{E}_m$) and the *stimulate events* ($\mathcal{E}_c$) families whereas the physical environment gathers all the events defined in the *reaction events* ($\mathcal{E}_a$) families.

Accordingly we systematically provide the user with the lists of the events that she/he must select for the decomposition process. The events of the control part are computed as: $\mathcal{E}_{soft} = \mathcal{E}_s \cup \mathcal{E}_m \cup \mathcal{E}_c$; those of the physical part are computed as: $\mathcal{E}_{phys} = \mathcal{E}_a$. These two lists are then used as the input of the decomposition plugin of Rodin.

The decomposition in Event-B consists in selecting and separating the desired events into two machines. In our case the model $M_{control}$ corresponding to the control part is the projection of $M_1$ on $\mathcal{E}_{soft}$. Similarly the model $M_{phys}$ corresponding to the physical part is the projection of $M_1$ on the list $\mathcal{E}_{phys}$ (See Fig. 17).

The Rodin tool already provides a decomposition plugin [15] that performs the projection of the provided machine according to the selected events. Therefore our assistance tool provides to its users, the lists of events that he/she should select when using the decomposition plugin of Rodin.

Machine $M_{soft}$
. . .
EVENTS
    events from $\mathcal{E}_{soft}$
END

Machine $M_{phys}$
. . .
EVENTS
    events from $\mathcal{E}_{phys}$
END

**Fig. 17.** Model decomposition skeleton

At this stage our tool provides much assistance to begin with the modelling in Event-B using Rodin; but many other tool facilities are provided with the Rodin to help its users. In the following we show how some of these tools can be used at various stages of our proposed method to satisfy the tool requirements identified in Sect. 3.5.

## 4.2. Tool Requirements Handled by Existing Rodin Plugins

We have experimented with some Rodin plugins to put our method in practice. But there are many other plugins available for Rodin to extend and complete the features of the

Rodin. We have studied them, and in the following we report on how some of these existing plugins can help one to apply the Heñcher method by resolving some of the previously mentioned requirements (see Section 3.5). Therefore for each of the tool requirements we advice the available or candidate plugins.

*$RT_2$ Incremental step-by-step refinement.* Our tool already provides a preliminary assistance in this direction by considering the families of events to be used in each refinement step.But more remain to be done to gain more flexibility within Event-B; there is a prototype plugin on *Group refinement* [4] which targets the simplification of the refinement links between abstract machines and their refinements; the idea is to relax the constraints on introducing dummy variables and their related housekeeping events that are there only to satisfy the refinement relation.

*$RT_3$ Pattern based substitutions and automatic refinements.* Some plugins propose to add (de-)composition features into Event-B/Rodin. The *Feature composition* plugin [17] enables one to merge Event-B machines and their seen contexts with the facilities to highlight multiple declarations of variables or events and to resolve conflicting elements.

*$RT_4$ Collaboration: working in parallel, versioning, decision traceability.* There exists a plugin called "team-based development" [19] which allows Event-B models to be stored in a SVN repository. That plugin allows to share the Event-B specifications but not the proof effort. The *Modularisation* plugin [5] may also help here by allowing separate development of part of the a global system as sub-modules and then by combining them. Indeed, this plugin enables one to weave together modules composing a model so that they work on the same global problem.

*$R_5$ Overall development process management.* An experimental plugin named *Model critic* [14] could be used to evaluate models using informal heuristics of what is typically a bad practice in model construction. That is a first step to evaluate quality of Event-B specifications, but it is not enough, to use it into an industrial process.

*$RT_6$ Iterative process of model evolution.* We identified several plugins that can help in fulfilling this requirement. The *Refactory* plugin [18] provides functionalities to rename the declaration and all the occurrences of an element of an Event-B model without modifying its proof state. There is a need for applying the same principle for more complex operations; for instance the insertion of event and its basic refinement through the chain of refinements, the detection and replacement of all the occurrences of an event in a model and trough its refinements, but with the smallest impact on the proof effort. The *Transformation patterns* plugin [20] could be used to automatise the different steps of the Heñcher method. From a given Event-B machine, we can produce a new one by applying some "transformations" : adding new variable, new event, new invariant, finding and using some event's guards, ask the user to enter some missing information, etc. No information is given about the necessary proof effort when using this plugin. The *Design pattern* plugin [16] is dedicated to reuse former development (as a pattern) in a new development by

[4] http://wiki.event-b.org/index.php/Group_refinement_plugin
[5] wiki.event-b.org/index.php/Modularisation_Plug-in

matching the corresponding variables and events. This results in a refined machine which embeds the former development. The proofs of the used pattern would be reused too. The correctness of the matching is only syntactically checked.

*$RT_7$ Traceability of the refinement chain of an event.* We have not yet identified any tool or plugin to help for this concern.

*$RT_8$ Securing copy-paste operations.* We have not yet identified any tool or plugin to help for this concern.

**Issues on tool development and maintenance** We are aware of the effort to be done to face the recurrent issues on tool development and maintenance. Most of the mentioned plugins are experimental, and insufficiently documented; they are not all based on the same (up-to-date) versions of Rodin. Many of them are difficult to install because of the (incompatible) required dependencies.

Accordingly, the *open source* policy is a solution to share the development and maintenance effort. We envision this solution for the tools we are developing and for the effort to be devoted to the remaining identified requirements.

## 5. Conclusion

In this paper we focused on the application of the Heñcher method to the Landing Gear Case study and paid much attention to the necessary tool support for the method. This led to the detailed presentation of the design of a companion tool of our method. Then we showed how the tool was used together with the Rodin tool and plugins to experiment with the case study. This work extends substantially our previous work presented in [4] where the proposed method Heñcher was introduced. We proposed the method Heñcher to guide step by step the construction of embedded control systems with Event-B. We built on the well-known structure of control systems and on the experiments of several case studies where the Event-B was used and where some methodological guidelines was provided [7,6,21]. We provided preliminary assessment in [4]; in this paper the case study is dealt with more details and we covered all the steps of the proposed method. The Landing Gear case study is representative of large systems involving the control and the interaction between software and physical parts. Before going into the presentation of the assistance tool that we have developed, we emphasised the motivations and the requirements for the needed tools.

Among the identified tool requirements, we detailed a tool to assist the specifiers in starting the modelling process, and to assist them during the development process with Rodin. Yet our Heñcher assistance tool is a prototype developed as a standalone web application. We experimented with Rodin but, it can be used not only for Rodin.

We have studied other available Rodin plugins that can help in putting into practice or to implement the remaining identified tool requirements.

Apart from the development of the tools related to facilitating the reuse of existing plugins (template reuse, refactoring, injection of events, etc), the short-term perspectives of our work are to continue the development of the Heñcher tools and their experimentation with other large scale case studies in combination with the Rodin platform. Our experiment with the Landing Gear case study which has the main features of cyber-physical systems, will be exploited to deal with more case studies of this category. Indeed there are other challenges to tackle in this area, such as handling real-time properties, dealing with the construction of systems built on the basis of digital twins concepts.

The long-term perspective is the provision of a generic development pattern related to embedded system which will make it more easier to describe requirements and properties and get the major part of the development generated.

**Acknowledgments.** Thanks to Léo Cassiau, Geoffrey Desbrosses, Alexis Giraudet, Jean-Christophe Guérin, Asma Khelifi, Ugo Mahey and Tantely Randriamaharavomanana, Master students at University of Nantes in 2017, who worked with us on the tools.

## References

1. Abrial, J.R.: Modeling in Event-B - System and Software Engineering. Cambridge University Press (2010)
2. Abrial, J.R., Hallerstede, S.: Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B. Fundam. Inform. 77(1-2), 1–28 (2007)
3. Alkhammash, E., Butler, M.J., Fathabadi, A.S., Cîrstea, C.: Building Traceable Event-B Models from Requirements. Sci. Comput. Program. 111, 318–338 (2015), https://doi.org/10.1016/j.scico.2015.06.002
4. André, P., Attiogbé, C., Lanoix, A.: Systematic Construction of Critical Embedded Systems Using Event-B. In: Abdelwahed, E.H., Bellatreche, L., Benslimane, D., Golfarelli, M., Jean, S., Méry, D., Nakamatsu, K., Ordonez, C. (eds.) New Trends in Model and Data Engineering - MEDI 2018 International Workshops, DETECT, MEDI4SG, IWCFS, REMEDY, Marrakesh, Morocco, October 24-26, 2018, Proceedings. Communications in Computer and Information Science, vol. 929, pp. 200–216. Springer (2018), https://doi.org/10.1007/978-3-030-02852-7_18
5. Boniol, F., Wiels, V.: The landing gear system case study. In: Boniol, F., Wiels, V., Ait Ameur, Y., Schewe, K.D. (eds.) ABZ2014. CCIS, vol. 433, pp. 1–18. Springer International Publishing (2014)
6. Damchoom, K., Butler, M.J.: Applying Event and Machine Decomposition to a Flash-Based Filestore in Event-B. In: 12th Brazilian Symposium on Formal Methods, SBMF 2009. LNCS, vol. 5902, pp. 134–152. Springer (2009)
7. Damchoom, K., Butler, M.J., Abrial, J.R.: Modelling and Proof of a Tree-Structured File System in Event-B and Rodin. In: 10th International Conference on Formal Engineering Methods, ICFEM 2008. LNCS, vol. 5256, pp. 25–44. Springer (2008)
8. Hoang, T.S., Snook, C.F., Fathabadi, A.S., Butler, M.J., Ladenberger, L.: Validating and verifying the requirements and design of a haemodialysismachine using the rodin toolset. Sci. Comput. Program. 158, 122–147 (2018), https://doi.org/10.1016/j.scico.2017.11.002
9. Jard, C., Roux, O.H. (eds.): Communicating Embedded Systems: Software and Design. Wiley-ISTE (2009)
10. Karsai, G., Sztipanovits, J., Ledeczi, A., Bapty, T.: Model-integrated development of embedded software. Proceedings of the IEEE 91(1), 145–164 (Jan 2003)
11. Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System. Commun. ACM 21(7), 558–565 (1978)
12. Méry, D., Singh, N.K.: Formal Specification of Medical Systems by Proof-Based Refinement. ACM Trans. Embedded Comput. Syst. 12(1), 15 (2013)

13. Parnas, D.L., Madey, J.: Functional Documents for Computer Systems. Science of Computer Programming 25(1), 41–61 (1995), `citeseer.ist.psu.edu/parnas95functional.html`
14. Event-b rodin platform plug-ins: Model critic plug-in, `http://wiki.event-b.org/index.php/Model_Critic`, accessed: 2019-03-14
15. Event-b rodin platform plug-ins: Decomposition plug-in, `http://wiki.event-b.org/index.php/Decomposition_Plug-in_User_Guide`, accessed: 2019-03-14
16. Event-b rodin platform plug-ins: Design pattern, `http://wiki.event-b.org/index.php/Pattern`, accessed: 2019-03-18
17. Event-b rodin platform plug-ins: Feature composition plug-in, `http://wiki.event-b.org/index.php/Feature_Composition_Plug-in`, accessed: 2019-03-14
18. Event-b rodin platform plug-ins: Refactoring framework, `http://wiki.event-b.org/index.php/Refactoring_Framework`, accessed: 2019-03-14
19. Event-b rodin platform plug-ins: Team-based development, `http://wiki.event-b.org/index.php/Team-based_development`, accessed: 2019-03-11
20. Event-b rodin platform plug-ins: Transformation patterns, `http://wiki.event-b.org/index.php/Transformation_patterns`, accessed: 2019-03-18
21. Satpathy, M., Ramesh, S., Snook, C.F., Singh, N.K., Butler, M.J.: A Pixed Approach to Rigorous Development of Control Designs. In: 2013 IEEE International Symposium on Computer-Aided Control System Design, CACSD 2013, Hyderabad, India, August 28-30, 2013. pp. 7–12. IEEE (2013), `https://doi.org/10.1109/CACSD.2013.6663474`
22. Silva, R., Butler, M.: Shared Event Composition/Decomposition in Event-B. In: 9th International Symposium onFormal Methods for Components and Objects, FMCO 2010. LNCS, vol. 6957, pp. 122–141. Springer (2012)
23. Singh, N.K., Wang, H., Lawford, M., Maibaum, T.S.E., Wassyng, A.: Stepwise formal modelling and reasoning of insulin infusion pump requirements. In: Duffy, V.G. (ed.) Digital Human Modeling - Applications in Health, Safety, Ergonomics and Risk Management: Ergonomics and Health - 6th International Conference, DHM 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9185, pp. 387–398. Springer (2015), `https://doi.org/10.1007/978-3-319-21070-4_39`

**Pascal ANDRE** received his Ph.D. from the University of Rennes I in 1995. He joined the Faculty of Sciences of Nantes in 1996 and then, spent 4 years as assistant professor at INP-HB engineering school (Ivory Coast). He is currently Associate Professor at the University of Nantes, France. He carries out his research activities in the Reliable Architecture and Software (AeLoS) team, at the laboratory of digital sciences of Nantes (LS2N). He published a series of course books on information system design in french. His main research topics concern the use of formal methods and verification techniques for software modelling and analysis, particularly in the context of component-based systems. He also actively work on rigorous approaches for model-driven engineering and reverse engineering.

**Christian ATTIOGBÉ** received the Ph.D. degree in Computer Science from the University of Toulouse, Toulouse, France, in 1992. He joined the Faculty of Sciences of Nantes in 1994 and he is currently Professor at the University of Nantes, Nantes, France. His research interests include formal approaches for software modelling and analysis, correct-by-construction using refinement, embedded-systems and heterogeneous systems modelling. He published several peer-reviewed papers on these topics. He is the leader of the

Reliable Architecture and Software (AeLoS) team, at the laboratory of digital sciences of Nantes (LS2N) since 2007.

**Arnaud Lanoix** received his Ph.D. from the University of Franche-Comté in 2005. He spent 3 years as a post-doctorate at the LORIA lab (Nancy, University of Lorraine). He is Associate Professor since 2008 at the Université de Nantes and carries out his research activities in the Reliable Architecture and Software (AeLoS) team, at the laboratory of digital sciences of Nantes (LS2N). His main research topics concern the use of formal methods and verification techniques for software modelling and analysis, particularly in the context of component-based systems.

*Received: May 1, 2019; Accepted: September 13, 2019.*

Computer Science and Information Systems 17(1):339–356 https://doi.org/10.2298/CSIS190511043L

# Game-based learning and Gamification to improve skills in early years education [*]

Rachid Lamrani[1], El Hassan Abdelwahed[1,2]

[1]Laboratory LISI, Cadi Ayyad University, 40000 Marrakech, Morocco.
rachid.lamrani@ced.uca.ac.ma
[2]CSEHS, Mohammed VI Polytechnic University, 43150 Benguerir, Morocco.
abdelwahed@uca.ac.ma

**Abstract.** Early childhood education has become a prevalent public policy issue. It has a serious impact on the child's personality, upbringing, education, socialization, development, and academic success from the preschool period to the university and beyond. In general, traditional teaching methods usually have a fixed learning structure which disables the child to be motivated, creative and innovative. Learners receive theoretical rather than practical instructions, which discourage them from keeping and recalling concepts and information more quickly. Moreover, traditional teaching usually lacks attracting the full attention of learners which decreases their interaction, engagement and investment in the content. Thus, the development of innovative approaches offering better education is an effective way to address this problem. On the other hand, recent researches in the fields of cognitive science and educational neuroscience show that play-based learning is a promising approach to use in early childhood education. Four key success factors for learning have been identified to strengthen children's skills, namely attention, active engagement, feedback, and consolidation. Thus, the proposed approach presents a digital play-based learning approach deploying serious games augmenting the pedagogical aspect of the Montessori approach. Our purpose is to improve children's skills in their early years education through play-based learning and gamification. It aims to provide children with a rich variety of serious gaming activities and challenging experiences in an interactive environment. We developed several serious games based on Montessori pedagogical principal and the four pillars of learning. For the evaluation, we have chosen a representative sample of children from rural regions.

**Keywords:** Early childhood education, Serious games, Gamification, Educational neuroscience, Pillars of learning.

[*] Extended version of Lamrani R., Abdelwahed E.H., Chraibi S., Qassimi S., Hafidi M. (2018) Gamification and Serious Games Based Learning for Early Childhood in Rural Areas. In: Abdelwahed E. et al. (eds) New Trends in Model and Data Engineering. MEDI 2018. Communications in Computer and Information Science, vol 929. Springer, Cham.

## 1. Introduction

Several studies have reported that early childhood education has a high impact on the child's socialization, development, and academic success. It also significantly impacts the socioeconomic outcomes of individuals. Indeed, early childhood is a critical and important stage where children acquire and develop their social and cognitive skills, self-esteem, and perception of the world. Early childhood education is an important and fundamental stage of learning. It is considered as one of the most effective ways of providing future generations with necessary skills and competencies needed to succeed in future labor markets.

Cognitive science claims [13] that learning requires being attentive, engaged, receiving and consolidating information, and giving immediate feedback. However, the basic traditional education does not respect these basic concepts. For example, it is insufficient in terms of direct learning. Also, it decreases the child's motivation by enforcing a preset program and a fixed learning structure to follow. Consequently, the children are not given the choice to take the lead in choosing the content to be learned. Therefore, it decreases the child’s motivation, creativity, innovation, then their attention which implies less engagement and knowledge acquisition.

It is more difficult to motivate the education of children living in rural areas of Morocco. Young girls are more deprived of schooling than boys living in disadvantaged areas due to the difficult living conditions (the lack of basic infrastructures, schools located far-away from the residential areas, etc.), poverty and lack of knowledge in their familial environment. Our challenging project ties in with many Moroccan organizations working to combat child school dropping by motivating their early education and raising awareness in rural areas. In fact, children intuitively learn through playing and interacting with others in a stimulating environment. It maintains their motivation, increases their interactions and their choice making.

Actually, recent educational neuroscience researches show that the best way to teach children is through playing, getting their attention, their engagement, receiving feedback and consolidating their skills [12]. Early years education should provide children with a rich variety of play activities and challenging experiences in a stimulating environment. Indeed, play is an essential activity for children to enhance their creativity and learning skills. The Playful behaviour can be considered as an exploratory and knowledge-building element [1]. Besides, playing at a younger age improves our capabilities to deal with real-life situations and interacting with the real world [2]. For children, playing is the natural way to improve their future skills starting from their early age [3]. When they play, they use plenty of their senses to capture and acquire diverse information and extend their knowledge about their environment. Moreover, children will develop new skills and abilities (e.g., talking, thinking, etc.) through playing. Also, playing provides children with the opportunity to boost their attention span, learn to get along with others, cultivate their creativity and address their social, emotional and cognitive needs. But also, it develops children's main academic skills like language and mathematics. In fact, learning through playing is a pedagogical strategy that is increasingly used in education [37]. The play mechanics create interactive and fun experiences for the learning player. Indeed, playing allows children to choose their activities and establish their own ways of doing things. Consequently, it enables them to control their learning and making new challenges.

Nowadays, the evolution and development of Information and communication technologies (ICT) improve the quality of education. It enables a broadcast of the same content through internet. In our case, it will rapidly widespread the diffusion of digital pedagogical materials and allow teachers and learners to access them online especially those living in rural areas. Serious games, IoT, virtual reality, cloud computing and many other emerging technologies offer the possibility to develop innovative learning solutions like the mobile and pervasive learning systems.

Our aim is to make use of ICT emerging technologies to develop a playing-based learning approach that relies on the fundamentals of neuroscience and Montessori pedagogical principal. In the conventional learning approach, teachers and students are attached to two separate worlds (the pre-digital generation and the digital generation). This explains the children's preference for digital games than traditional ones. In this regard, our aim is to make use of digitalization and emerging technologies to augment serious games based on the pedagogical aspect of the Montessori approach.

This paper presents a methodological approach that uses serious games based on the cognitive of playing and Montessori educational method in order to offer a research-based solution that makes playtime more stimulating and educational for children.

The rest of the paper is organized as follows: the early childhood education benefits are described in Section 2. Section 3 provides explanation where educational neuroscience meets early childhood. Section 4 depicts the concept of learning through play including the Montessori educational approach. The gamification and serious games based learning for early childhood containing our proposed approach are described in Section 5. The implementation and the experimentation results of our developed serious games are described in Section 6. Finally, the conclusion and future directions are delineated in Section 7.

## 2. Early childhood education benefits

The preschool prepares young children for the elementary education. It is considered an instructive period in the grounding of concepts and constant ideas [32]. During this phase, different experiences influence outcomes across the entire course of an individual's life. Children, who grew up in a rich and supportive environment, are more likely to achieve their full potential and achieve optimal physical, cognitive, linguistic and socio-emotional development. In addition, young children mostly benefit from childhood enrichment program of intellectual activities, emotional reactions, and behaviors. Consequently, an early childhood education quality can provide essential experiences for the child's brain development, therefore, have a direct effect on his cognitive abilities and future learning capacities [6].

### 2.1. Long-lasting benefits of preschool

Nowadays, studies conducted in the United States [9, 10] and in the United Kingdom [11] have shown a positive relationship between childcare quality and child's development outcomes. These studies have assessed the quality, quantity, and the type of childcare at regular intervals. The Cost, Quality, and Outcomes study found that there is a positive relationship between preschool quality and children’s language and mathematical abilities. The results of the NICHD ECCRN study, covering over 1,300 children from 10 sites from birth, indicated that high-quality care is related to better cognitive outcomes, less impulsivity, and better social competencies at 4.5 years of age [10]. The preschool phase helps the children to strengthen the beginning of learning and success. One of the main good education practices:

- *Socialization*: It represents a serious matter to familiarize children with others and strengthen their exchanges. It also allows children to overcome their shyness and increase their social interactions.
- *Concept of Cooperation*: It enables sharing, coordinating, and adopting a comfortable learning condition.
- *Passion for Lifelong Learning*: Lessons should be given in a fun and stimulating way that cheers children up to be more attentive and engaged on their learning process.
- *Confidence and Self-Esteem*: It offers positive thinking and confidence for children.
- *Develop literacy and numeracy skills*: children learn by tuning in to stories, discussing pictures and drawing shapes. For example, they learn numeracy abilities by singing and playing music. The proficiency and numeracy abilities in the preschool affect the child’s scholastic achievement.

## 3. Where educational neuroscience meets early childhood

Neuroscientists have shown that the brain has a great capacity to adapt to the demands of its environment: Plasticity [7]. Neuroscience includes all the fundamental disciplines necessary to explore the anatomy and functioning of the nervous system, and more specifically, the brain. Over the past fifteen years, this new discipline has continued to progress in understanding the functionalities of the brain and its surroundings. In fact, while brain sciences deal with the processes underlying learning, education aims to apply them in real life and more particularly in school life. However, although there are obvious bridges between these two disciplines, neuroscience is in the process of being used in the education realms. Neuroscience synergizes with other disciplines, have broadened our understanding of the brain in a way that is highly relevant to educational practices [12]. Cognitive science has identified at least four key factors as pillars of learning processes and pedagogical strategies [13] [14]. Actually, good learning involves attention, active engagement; feedback and consolidation (see Fig.1).

Attention mainly modulates brain activity. It is the gateway for learning that enables children to be focused on choosing and processing relevant information. In the play-based learning approach, the materials should avoid children to stay stuck and be distracted from their primary tasks. The main objective is to capture and draw the child's attention on relevant levels through ludic, ergonomic pedagogical materials. The challenge, therefore, is to focus the attention of children during their learning by inhibiting undesirable behaviours.

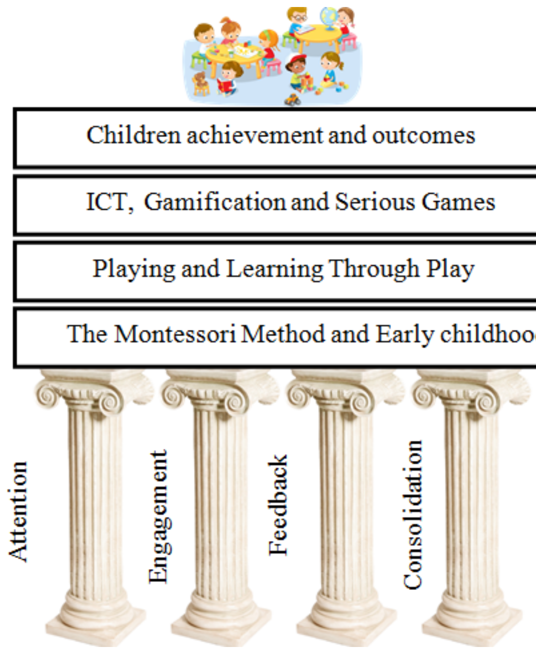

Children achievement and outcomes
ICT, Gamification and Serious Games
Playing and Learning Through Play
The Montessori Method and Early childhood
Attention
Engagement
Feedback
Consolidation

**Fig. 1.** Foundations of the proposed approach.

A passive learner does not learn. The active engagement of the learner underscores his curiosity and denotes his abilities to be maximally attentive, active, and predictive. Thus, we should take into account making learning conditions reasonably challenging that are neither easy nor difficult but adequate to the learner's context. This should paradoxically lead to increase engagement and cognitive efforts, which means improved attention. As a matter of fact, preserving commitment means that the teacher must avoid giving a long lecture, but involve the children, test them frequently, guide them while allowing them to discover certain aspects by themselves, and reward systematically their curiosity rather than discourage it.

During the learning activities, the child should have the possibility to test himself the reliability of his knowledge. The feedback of information is essential and the difference with the made prediction generate an error signal that would contribute to correct and to improve the following prediction. Indeed, to err is human. Far from being a fault or a weakness, the error is inevitable but also necessary and fertile even indispensable in learning situations. Better an active child who is wrong and learns from his mistakes, than a passive child. Further, the consolidation considered as knowledge automation where the brain achieves automation. Consolidation is the act of passing from conscious treatment with an effort, to automated unconscious treatment and the challenge is to accomplish the transfer from explicit to implicit.

Correspondingly, the child learns by his emotional intelligence [15] [16], then develops a link with his teacher, which makes him learn more words and operations. As soon as it grew, he should be initiated at the intelligence logic, which must be implemented by single organizations and visual methods.

## 4. Learning through Play

### 4.1. Play in different ways

There are different types of play that correspond to each stage of a child's learning progress.
*Exploratory play*: using physical skills and sensations to learn about materials and their properties.

*Constructive play*: using objects and materials (e.g., blocks, playdough, collage materials, sand and water) enhances their creativity, recognition and solving problems.

*Creative play*: using open-ended materials such as art materials and natural materials to encourage fluency, flexibility, originality, imagination, and making novel connections.

*Socio-dramatic play*: involves interaction and verbal communication with one or more play partners regarding the play event.

*Physical locomotor play*: a range of fine or gross motor skills are practiced involving all kinds of physical movements.

*Language or word play*: incorporates rhyme, wordplay and humor.

Play in early childhood performs an important role in learning. It is significant in cognitive, psychomotor, emotional, social development, and so on. The most commonly applied pedagogies are Piaget's constructivism [2] and Vygotsky's Zone of Proximal Development [17]. For the Piaget's constructivism, a child is motivated by their curiosity to acquire their knowledge through their experiences and other's influences. Moreover, the co-operative social interaction of children with adults, promotes cognitive, and affective development. The Zone of Proximal Development (ZPD) as proposed by Vygotsky [17] enables problem solving under adult guidance or collaboration with more skilled peers. One of the main approaches using play to learn is Montessori. Actually, many applications of constructivist and learning discovery mainly use Montessori materials.

### 4.2. The Montessori educational approach

The Montessori educational method has been created by Dr. Maria Montessori in a poor neighborhood in Rome in 1907. Maria Montessori (1870–1952) was qualified among the first women to a medical doctor specialized in psychiatry and pediatrics in Italy. She worked with children with intellectual disabilities, she had an important insight that they did not require medical treatment to learn but rather an appropriate pedagogy. She achieved fostering her pupils' self-construction and learning on several stages of development by engaging with self-directed activities within a prepared environment.

She continued improving her pedagogy based on a scientific approach of experimentation and observation for 45 years. Since its initiation, the Montessori approach has attracted international interest and has spread around the world [37].

Montessori’s educational method has two important aspects: The learning materials; And the self-directed nature of children’s engagement with those materials [37, 38].

- Each piece of material covers a concept to be learned, containing a self-correction control, and having a learning process starting from the concrete to the abstract concepts.

- The learning material triggers a self-directed nature of the child's engagement with those materials to enhance the learning process under the teacher’s expert guidance. For example, the child-led manner is expressed through his self-selection and repeated engagement.

This method potentially benefits from enhancing the development of the learning process compared to the teaching of the conventional classroom [39]. Arguably, Lillard and Else-Quest [40] is the most robust evaluation of the Montessori method so far [37].

When the Montessori educational method is rigorously implemented in a school, it fosters social and academic skills equally or better than those fostered by non-Montessori schools [40]. The authors [44] conducted a study of 172 children in Montessori and conventional school classrooms, where the Montessori classrooms performed best on a wide array of social-emotional and academic dimensions. Indeed, well-implemented Montessori classrooms have superior outcomes than conventional ones [45].

### 4.3. Benefits of Play

A review of more than 40 studies found that play is significantly related to creative problem solving, co-operative behaviour, logical thinking, IQ scores, and peer group popularity. Play enhances the progress of early development from 33% to 67% by increasing adjustment, improving language and reducing social and emotional problems.

Playing occupies a considerable amount of children's daily time and energy. Some of benefits of play are listed below:

*Building imagination and ability,* during play, kids typically mimic adults and build make-believe games that widen their imagination.

*Psychological feature Growth,* free play has an effect on confidence, intelligence and communication.

*Group Interaction,* Group play develops the necessary skills improving self-control and group integration.

## 5. Gamification and Serious Games Based Learning for Early Childhood

### 5.1. Gamification and Serious Games

Gamification aims to transform systems, activities, organizations, and services to a system that uses the characteristics of game elements [33]. Gamification is a well-known technique in education, organization engagement, crowdsourcing, commerce, information retrieval, and so on [34, 35]. It is practical to improve learning processes especially the learners' motivation [36]. Gamification techniques are benefiting from advances in ICT. Applications of gamification span a wide range including healthcare, marketing, management and recruitment, as well as learning and teaching. The relation between education and gamification is on the rise by making use of learning activities as a subject to gamification.

The intention is therefore to motivate and involve learners in becoming active participants in their own learning process. In essence, the pedagogical experience is transformed into an educational challenge by the use of badges offered in case of achievement, scoreboards, progression levels and missions. All of these game elements are integrated to support the learner in achieving their goals and learning objectives. In summary, gamification uses game mechanics to transform the educational experience into an effective learning process.

A serious game is a computer application that combines with consistency, both serious aspects such as learning, or communication intent, with playful springs from the video game like collaboration, competition and strategy [26, 27]. Actually, their main use aims to improve users' skills, engagements and performances [28, 29].

The term "serious gaming" has been used since the 17th century. Serious games were introduced in the 1970s due to the efforts of pioneers like Clark Abt. The author [52] published a book entitled Serious Games where he describes the possible uses of games to learn and simulate situations. He uses the term with reference to card and board games and role-playing games.

Three main criteria, "G/P/S model", are used for serious games classification [30]:

- G: Gameplay that incorporates all the mechanisms used through the game rules, player and the game connection, challenges, and so on.
- P: Purpose represents the hidden information on the functions that go beyond the entertainment provided by the designed games.
- S: Sector, the areas of applications covered by the "Serious Game".

Relevant serious games applications have recently been developed in different domains including education, training, well-being, advertisement, cultural heritage, interpersonal communication, and healthcare. Advances in gaming technologies allow the real-time interactive visualization and simulation of realistic virtual heritage scenarios, such as reconstructions of ancient sites and virtual museums [18]. Many research contributions are directed towards taking advantage of the success of video games and using them for the benefits of the educational domain [19]. Also, there are a few research studies and projects that use serious games in the context of preschool to develop the children's abilities and academic skills in mathematics and languages [20, 21].

In fact, serious games have been used in several domains such as, Tourism [47] to add value in tourism marketing and management; Energy assumption [48] to positively influence consumers about their energy assumption. Education [46] where using games shows positive effect during the learning process by gathering a survey of players using only two learning games. The authors [49] propose the use of didactic games to improve the arbitrary memory for preschoolers. Other authors [43] introduce an intervention program that uses voices and detects gestures to teach colors and shapes to preschoolers. Another research study [50] aims to use online gamification to promote academic dissemination. Recent approaches [41, 42, 43] consider applying serious games on education, our proposal also considers the children's cognitive developments through playing. Indeed, Montessori educational fundamentals are supported by the current psychological research [45].

Based on Montessori pedagogical principal and the four pillars of learning, our aim is to develop a proposed approach that provides children serious games in rural areas.

## 5.2. Proposed approach

As it was mentioned above, one of the motivating challenges is to elaborate pertinent solutions addressing the problem of dropping out school in early childhood. In this context, we have initialized a project aiming to develop innovative solutions to deliver an accessible early childhood education. Our goal is to eliminate the inequality in the matter of education and create real opportunities for children, in particular young girls, in rural areas to have access to education.

The pedagogical method we adopted within our project is based on Montessori approach [22, 23,24]. It states that the purpose of early childhood education is to raise the motivation of children in learning.

The Montessori approach distinguishes five categories of activities and skills to develop (see Fig.2). We aim to develop some serious games enhancing the child’s learning, such, mathematics and science skills, reading and spelling and so forth.

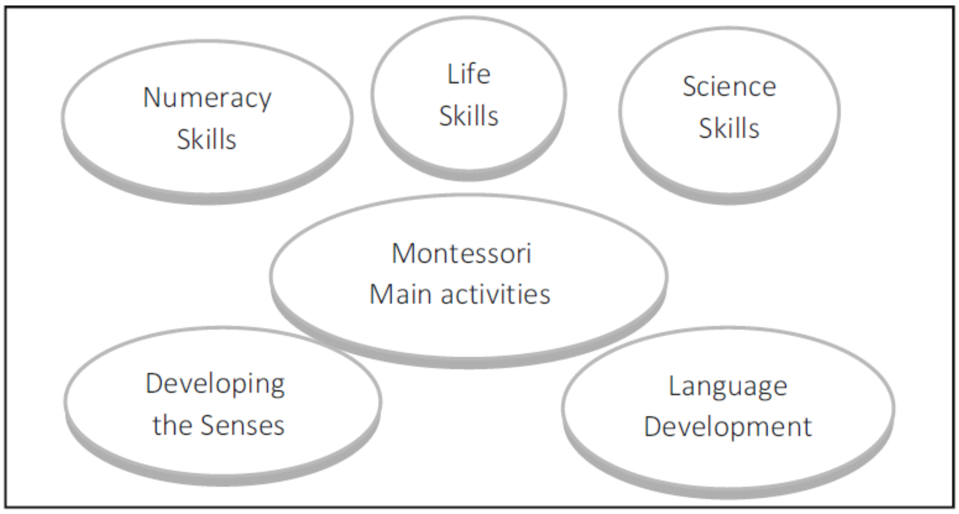


**Fig. 2.** The montessori approach main activities and skills.

This approach proposes the main learning activities and skills. During learning activities, the child should be autonomous and be mainly motivated by its natural curiosity. The fundamental principles of our approach are described as follows:

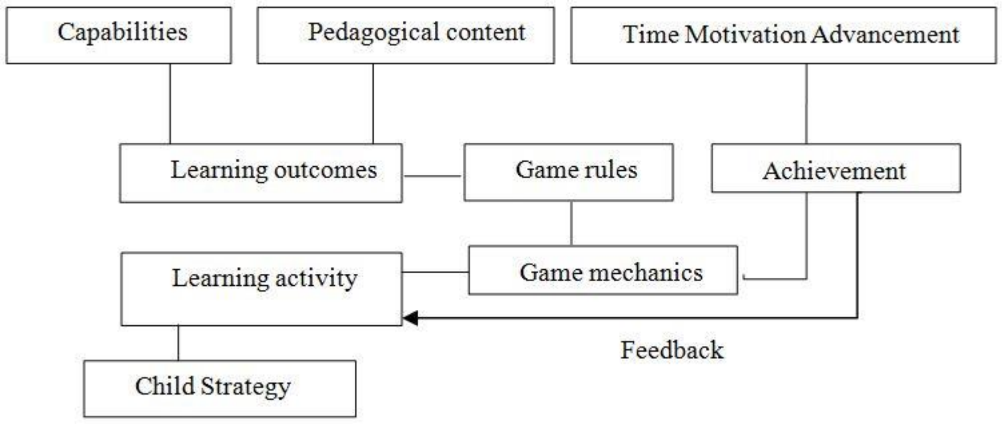


**Fig. 3.** Proposed approach structural diagram.

* The child's activity is initiated by his own will and not by his companion's instructions.

* Repetition of manipulation allows the child to respond to his curiosity and his personal pursuit.

* The most important part is to learn by doing. The child is encouraged to learn by practicing.

* The child absorbs the impressions given by his environment by offering a rich atmosphere of experiences.

Furthermore, the serious games developed in our project are aligned to the Montessori approach and with respect to the pillars of learning reflecting the educational cognitive science point of view. Indeed, errors are considered as phases of the game and do not prevent children from acquiring skills. In addition, immediate feedback ensures the quality of what they have learned, which is a very important factor for effective learning.

## 6. Implementation

### 6.1. Architecture and application

The developed serious games can be used in a group of children either in the context of online learning or blended learning that joins traditional classroom methods. They are accessible using a mobile device like a smartphone, a pad or a desktop (see Fig. 4). Presently, we have implemented more than twelve serious games integrating different Montessori Approach's main activities and skills. Below, we present some examples of developed serious games (Tables 1, 2, and 3). Our proposed play-based solution offers a pervasive learning within mixed aged children.

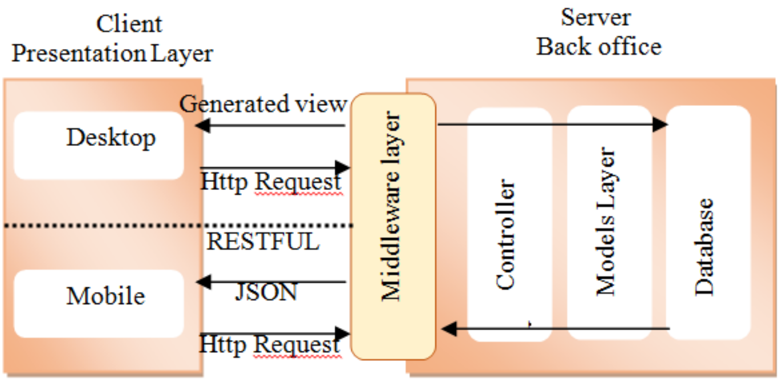


**Fig.4.** Main Architecture.

**Table 1.** Serious games concerning the language and the numeracy skills development.

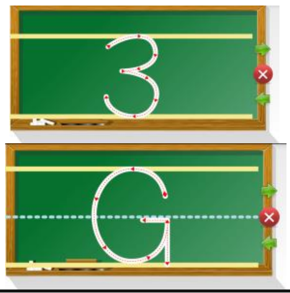
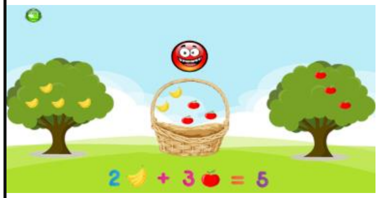
| Activities and skills : Language Development | Activities and skills : Numeracy Skills |
|---|---|
|  |  |
| Game goal and Guidelines: This game help kids recognizing a letter, numbers shapes, associate them with phonic sounds, and put their alphabet knowledge to use in fun exercises. It has the same pedagogical goal and it is aligned with the entitled games "The phonetic alphabet" and "Identifying alphabets" in Montessori . | Game goal and Guidelines: This game is about numbers and how to use them and apply some basic mathematical operations with quantities (using fruits). This game has the same pedagogical goal and it is aligned with the entitled game "Addition using numerals" in Montessori. |

**Table 2.** Serious games concerning the senses and the life skills improvement.

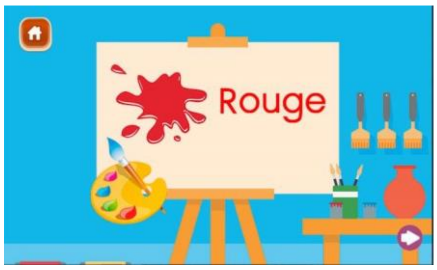
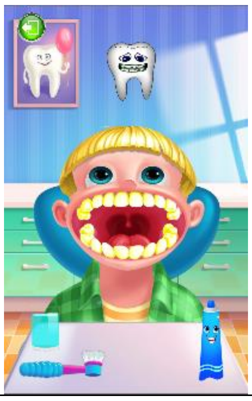
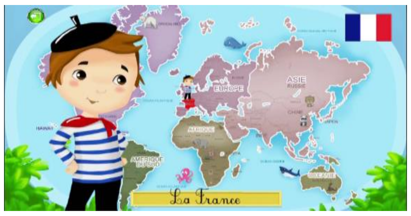
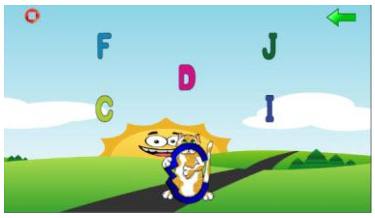
| **Activities and skills** : Developing the senses | Activities and skills : Life Skills |
|---|---|
|  |  |
| Game goal and Guidelines: This game teaches the kid the colors, their spelling and their phonetic sound. This game has the same pedagogical goal and it is aligned with the entitled game "Discovering colors" in Montessori theory. | Game goal and Guidelines: This game shows the child the importance of brushing teeth. It has the same pedagogical goal and it is aligned with the entitled game "Cleaning teeth"in Montessori theory. |

**Table 3.** Serious games concerning Science and geography skills development.

| **Activities and skills** : Introducing geography | **Activities and skills** : Introducing language |
|---|---|
|  |  |
| Game goal and Guidelines: This game teaches the child the continents, the countries and their location by making their first experience of geography as concrete and fun. This game has the same pedagogical goal and it is aligned with the entitled game "Introducing a globe and map " in Montessori theory. | **Game goal and Guidelines:** We can use this game as a starting point to introduce and practice several things at once (how to say the names of the letters and about all the sounds of each letter, etc.). Children learn by observing, listening and imitating. |

## 6.2. Evaluation

We have focused on integrating points explained bellow to develop a set of interactive serious games. The first one concerns the cognitive development by considering the children's activity to enhance their intelligence progresses.

The developed serious games promote learning through the children's experiment, exploration, learning from mistakes, then understanding and repeating. Experiencing new way of leaning triggers and activates their neurotransmitters. We have applied our developed serious games on a group of children having different ages (4 – 6) from different rural areas as shown below in Table 4.

**Table 4.** Experimental results.

| Identified factors | | average | Motivation |
|---|---|---|---|
| Numeracy Skills | • Recognizing letter, numbers shapes, and associate them with phonic sounds. | 4.2 | 3 |
| | • How to use numbers to apply some basic mathematical operations. | 4.3 | 3 |
| Life Skills | • Teaching the correct way to brush their teeth. | 4.1 | 4 |

| | | | |
|---|---|---|---|
| Child’s speech and language development | • Introducing basic French vocabulary, such as the alphabet, colours, names, and so on. | 4.2 | 3 |
| | • Correcting the alphabet spelling. | 4.4 | 2 |
| Geography development | • Teaching children the names of continents. | 4.8 | 4 |

The evaluation has been conducted evaluating 30 children (40 % girls and 60 % boys). Thereafter, we have collected the serious games’ sored tracking in order to analyze the results. At the meantime, and to obtain their reactions, a “stakeholder feedback session” has been done.

The evaluation considers the instruction of the research study [25]. We have targeted two indicators. The first is the amount of work performed (it is measured based on the number of completed games’ levels) and the second one is based on the accumulated knowledge (resulting from each feedback session). We focus on a certain learning topics; we developed our solutions including the learning activities and skills. For the evaluation, we used a scale from 1 to 4, where {1 = “boring”; 2 = “so-so”; 3 = “fun”, and 4= “awesome”} to measure the children’s learning skills and feedback to the serious games. Based on the results and tracks gathered during the "feedback session", we were impressed by the final rendering, a perfect understanding of the content in a short period.

A score is assigned at each accomplished game’s level, varying from 1 to 5, in fact, the game does not display scores but rather a reward, which is represented by an interactive cartoon character.

## 7. Conclusion and Perspectives

Play is crucial to the development of social, emotional, linguistic and intellectual child's abilities. It is a great way to discover the world and flourish. In reach of playing, the child flavour more confident, autonomous and have more pleasure to acquire new academic skills (mathematics, language, etc.) and social aptitudes (confidence, communication, etc.). All along playing and doing fun actions, he would be more motivated and curious to discover the world around him while adopting a positive attitude towards action.

In addition, play contributes to the shaping of the child's identity through creativity. Playing is exploring the world outside, having the freedom to decide. It’s the place for unique experiences, area of innovation and inventiveness. It’s also, consociates children with their imagination, parents and their surroundings.

Education through play, also known as edutainment, is a combination of education and amusement. The idea is to enlighten and empower the child by incorporating learning into various forms of diversion such as television programs, computer games, and multimedia programs or even through music. The over-the-top decade has seen a tremendous increase in digital game-based learning, with the flourishing and rising prevalence of smart phones and tablets reshaping learners' expectations. As a result, the use of digital pedagogical materials offers stimulating learning environments are becoming increasingly desirable.

Nowadays, African countries present a significant number of dropping out of school, in particular in preschool. This is due to several reasons, such as poverty, growth tuition fees as well as the lack of security. Learning starts at birth, and the first six years are for discovering and exploring. Indeed, a strong beginning in the early years provides individuals with the best and fairest chance to reach their fullest potential. Therefore, it would be essential for gathering all efforts to find innovative solutions to deliver an accessible education allowing a gapless learning. Correspondingly, the use of Serious Games by advantage of their specificities (amusement, divertissement and interactivity), in both formal and informal curriculum, presents encouraging results. It would impact classical learning methods, especially in early childhood education.

In this paper, we have presented our propositions and contributions to assure the development of the children’s early learning, in particular in rural areas. In fact, access to preschool is the main factor for individuals’ school success and thus social and economic development of the countries. For early childhood, we propose a Montessori’s Method based serious games solution. We developed several serious games according to an agile method. We have identified some factors for the assessment of our solution, and respectively, the experimentation is carried out on a specific children group from different regions, in order to evaluate the acceptance and usefulness of our approach.

What we expect in the near future, is that our approach and solution will have great flexibility and will meet the needs of a large scale of children. Another challenge that we try to face is about the youth unemployment problem in Northern Africa. To treat the scourge of youth unemployment, we project to capitalize on the outcomes of the actual project to develop a pervasive collaborative system to enhance Northern African youth entrepreneurship through gamification [31].

In future perspectives, we aim to construct an educational recommender system [51] that uses child-players feedbacks for suggesting learning pathways of serious games that fit in as much as possible with the profile and motivation of the learners. As well as working on other evaluation aspects, e.g. evaluating the children's reasoning improvement.

## References

1. Assaf, T.: La place des jeux traditionnels dans l'EPS : analyse socio-historique de 1891 à nos jours; le cas de la Gironde (2010). Available: http://www.theses.fr/2010BOR21708 (current May 2019)
2. Piaget, J.: Play, Dreams and Imitation in childhood. New York: W. W. Norton & Company. (1952)
3. Sheridan, M., Howard, J., Alderson, D.: Play in Early Childhood. Routledge, London (2011)
4. Landry, S.H.: The role of parents in early childhood learning. In: Tremblay, R.E. (ed.) Encyclopedia on Early Childhood Development (2014)
5. Erola, J., Jalonen, S., Lehti, H.: Parental education, class and income over early life course and children’s achievement. Res. Soc. Strat. Mobil. 44, 33–43 (2016)
6. Torkel, K.: The Learning Brain: Memory and Brain Development in Children. Oxford University Press (2012)

7. Kolb, B., Gibb, R.: Brain plasticity and behaviour in the developing brain. Journal of the Canadian Academy of Child and Adolescent Psychiatry, 265–276 (2011)
8. Thompson, R., Nelson, C.: Developmental science and the media: Early brain development. American Psychologist. 56, 5-15 (2001)
9. The Children of the Cost, Quality, and Outcomes Study Go To School, Available : http://fpg.unc.edu/sites/fpg.unc.edu/files/resources/reports-and-policy-briefs/NCEDL_CQO_technical_report.pdf. (current May 2019)
10. Early Child Care and Children's Development in the Primary Grades: Follow-Up Results From the NICHD Study of Early Child Care. American Educational Research Journal. 42, 537-570 (2005)
11. Sylva, K., Siraj-Blatchford, I., Taggart, B., Sammons, P., Melhuish, E., Elliot, K., Totsika, V.: Capturing quality in early childhood through environmental rating scales. Early Childhood Research Quarterly. 21, 76-92 (2006)
12. Sigman, M., Peña, M., Goldin, A., Ribeiro, S.: Neuroscience and education: prime time to build the bridge. Nat. Neurosci. 17, 497–502 (2014)
13. Dehaene, S.: Cognitive foundations of learning in school-aged children. Collège de France. Available: https://www.college-de-france.fr/site/en-stanislas-dehaene/course-2014-2015.htm. (current May 2019)
14. Jacob B. Feiler, Maureen E. Stabio: Three pillars of educational neuroscience from three decades of literature, Trends in Neuroscience and Education, Volume 13, Pages 17-25, (2018)
15. Raver, C., Garner, P., Smith, D.: The roles of emotion regulation and emotion knowledge for children's academic readiness: are the links causal? In: Planta, B., Snow, K., Cox, M. (eds.) School Readiness and the Transition to Kindergarten in the Era of Accountability, pp. 121–147. Paul H Brookes Publishing, Baltimore (2007)
16. Eggum, N., et al.: Emotion understanding, theory of mind, and prosocial orientation: relations over time in early childhood. J. Posit. Psychol. 6, 4–16 (2011)
17. Vygotsky, L. S.: Mind in society. Cambridge, MA: MIT Press, (1978)
18. Neto, J., Silva, R., Neto, J., Pereira, J., Fernandes, J.: Solis'Curse - a cultural heritage game using voice interaction with a virtual agent. In: 2011 Third International Conference on Games and Virtual Worlds for Serious Applications (2011)
19. Muratet, M., Torguet, P., Jessel, J., Viallet, F.: Towards a serious game to help students learn computer programming. Int. J. Comput. Games Technol. 2009, 1–12 (2009)
20. Nikiforidou, Z., Pange, J.: Shoes and squares: a computer-based probabilistic game for preschoolers. In: Procedia - Social and Behavioral Sciences, vol. 2, pp. 3150–3154 (2010)
21. Schuurs, U.: Serious gaming and vocabulary growth. In: De Wannemacker, S., Vander-cruysse, S., Clarebout, G. (eds.) ITEC/CIP/T 2011. CCIS, vol. 280, pp. 40–46. Springer, Heidelberg (2012)
22. Lillard, A.: Preschool children's development in classic montessori, supplemented montessori, and conventional programs. J. Sch. Psychol. 50, 379–401 (2012)
23. Alvarez, C.: Les lois naturelles de l'enfant. Les Arènes (2016)
24. Pitamic, M.: Teach Me to Do it Myself: Montessori Activities for You and Your Child. Barron's Educational Series (2004)
25. Leutenegger S., Edgington J.: A games First Approach to Teaching Introductory Programming In SIGCSE '07 : Proceedings of the 38th SIGCSE technical symposium on Computer science education, 115-118, (2007)
26. Djaouti, D.: Serious Games pour l'éducation: utiliser, créer, faire créer? Tréma 44, 51–64 (2016)
27. Wattanasoontorn, V., Boada, I., García, R., Sbert, M.: Serious games for health. Entertain. Comput. 4, 231–247 (2013)
28. Giessen, H.: Serious games effects: an overview. Procedia - Soc. Behav. Sci. 174, 2240–2244 (2015)

29. Lamrani, R., Abdelwahed, E.H.: Learning through play in pervasive context: a survey. In: IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, pp. 1–8 (2015)
30. Alvarez, J., Damien, D.: An introduction to Serious game Definitions and concepts, In Proceedings of the Serious Games & Simulation Workshop,Paris, 10-15, (2011)
31. Lamrani, R., Abdelwahed, E.H., Chraibi, S., Qassimi, S., Hafidi, M., El Amrani, A.: Serious game to enhance and promote youth entrepreneurship. In: Rocha, Á., Serrhini, M., Felgueiras, C. (eds.) Europe and MENA Cooperation Advances in Information and Communication Technologies. Advances in Intelligent Systems and Computing, vol. 520, pp. 77–85. Springer, Cham (2017).
32. Arbianingsih, Rustina, Y., Krianto, T., Ayubi, D.: Developing a health education game for preschoolers: what should we consider? Enfermería Clínica, 28, 1–4 (2018).
33. Huotari, K., & Hamari, J.: A definition for gamification: Anchoring gamification in the service marketing literature. In: Electronic Markets, 27(1), 21–31, (2017).
34. Warmelink, H., Koivisto, J., Mayer, I., Vesa, M., & Hamari, J.: Gamification of production and logistics operations: Status quo and future directions. In: Journal of Business Research. (2018).
35. Morschheuser, B., Hamari, J., Maedche, A.: Cooperation or Competition - When do people contribute more? A field experiment on gamification of crowdsourcing. In: International Journal of Human-Computer Studies (2018), doi: 10.1016/j.ijhcs.2018.10.001.
36. Chung, chih-hung & Shen, Chun-Yi & Qiu, Yu-Zhen.: Students' Acceptance of Gamification in Higher Education. In: International Journal of Game-Based Learning. (2019)
37. Chloë Marshall: Montessori education: a review of the evidence base. In: npj Science of Learning volume 2, Article number: 11 (2017)
38. Lillard, A. S.: Preschool children's development in classic Montessori, supplemented Montessori, and conventional programs. J. School Psychol. 50, 379–401 (2012).
39. Montessori, M.: The Discovery of the Child. In: Clio Press, Oxford, UK, 1912/1988.
40. Lillard, A. S. & Else-Quest, N.: Evaluating Montessori education. In: Science 313, 1893–1894 (2016).
41. Hirsh-Pasek, K., Zosh, J. M., Golinkoff, R. M., Gray, J. H., Robb, M. B., & Kaufman, J.: Putting education in "educational" apps: Lessons from the science of learning. In: Psychological Science in the Public Interest, 16(1), 3-34, (2015).
42. Johnson, T. M., Ridgers, N. D., Hulteen, R. M., Mellecker, R. R., & Barnett, L. M.: Does playing a sports active video game improve young children's ball skill competence? In: Journal of Science and Medicine in Sport. (2015).
43. Lai, N. K., Ang, T. F., Por, L. Y., & Liew, C. S.: Learning through intuitive interface: A case study on preschool learning. In: Computers & Education, 126, 443–458, (2018).
44. Lillard, A. S.: Preschool children's development in classic Montessori, supplemented Montessori, and conventional programs. In: Journal of School Psychology, 50, 379–401, (2012).
45. Lillard, A. S.: Rethinking Education: Montessori's Approach. In: Current Directions in Psychological Science, (2018).
46. Juho Hamari, David J. Shernoff, Elizabeth Rowe, Brianno Coller, Jodi Asbell-Clarke, Teon Edwards.: Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning,. In Computers in Human Behavior, Volume 54, Pages 170-179, ISSN 0747-5632, (2016).
47. Feifei Xu, Dimitrios Buhalis, Jessika Weber.: Serious games and the gamification of tourism, Tourism Management, Volume 60, Pages 244-256, ISSN 0261-5177, (2017).
48. Daniel Johnson, Ella Horton, Rory Mulcahy, Marcus Foth.: Gamification and serious games within the domain of domestic energy consumption: A systematic review, Renewable and Sustainable Energy Reviews, Volume 73, Pages 249-264, ISSN 1364-0321, (2017).

49. Elena V. Bateneva.: Arbitrary Memory Improvement in Older Preschoolers Using Didactic Games, Procedia. In: Social and Behavioral Sciences, Volume 233, Pages 259-263, ISSN 1877-0428, (2016).
50. Ming-Shiou Kuo, Tsung-Yen Chuang.: How gamification motivates visits and engagement for online academic dissemination – An empirical study. In: Computers in Human Behavior, Volume 55, Part A, Pages 16-27, ISSN 0747-5632, (2016).
51. Sara Qassimi, El Hassan Abdelwahed, Meriem Hafidi and Rachid Lamrani.: A Graph-Based Model for Tag Recommendations in Clinical Decision Support System. In: The 8th International Conference on Model and Data Engineering MEDI 2018, pp. 292-300, Marrakesh, Morocco, October 24-26, (2018).
52. Abt, Clark C.: Serious games .Viking Press New York. (1970)

**Rachid LAMRANI** holds a Master degree in Computer science from ENSET Mohammedia, Morocco. He is currently a PhD student in the Department of Computer Scienc, Laboratory of Computer Systems Engineering (LISI) at Faculty of Semlalia, University Cadi Ayyad Morocco. His current researches include Educational Technology, Serious Games, Gamification and Applications, Contextual and Ubiquitous learning, Learner engagement. He is the author and co-author of peer-reviewed scientific publications related to his research interests.

**El Hassan ABDELWAHED** holds a Ph.D. in Computer Science and Robotics from Montpellier II University France and Doctorat d'Etat in Computer Science from Cadi Ayyad University Morocco. He is currently a full Professor of computer science a Cadi Ayyad University and affiliated professor at Mohamed VI Polytechnic University in Morocco. He was the Head of the Computer Science department from 2005 to 2009 and the director of the Laboratory of Computer Systems Engineering (LISI) since 2015. His research interests include Data Science, Context-aware systems, Educational Technology, Machine Learning, Recommender systems and their applications (Industry 4.0, Disruptive innovation & Smart Education, etc.). He was program chair and a member of the program committee member of international and National Conferences and Workshops. He actively contributes in promoting research in Morocco where he supervises several Ph.D. students and organizes conferences and workshops. He is the author and co-author of peer-reviewed scientific publications related to his research interests.

*Received: May 11, 2019; Accepted: September 20, 2019*

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

004

COMPUTER Science and Information
Systems : the International journal /
Editor-in-Chief Mirjana Ivanović. – Vol. 17,
No 1 (2020) - . – Novi Sad (Trg D. Obradovića 3):
ComSIS Consortium, 2020 - (Belgrade
: Sigra star). –30 cm

Polugodišnje. – Tekst na engleskom jeziku

ISSN 1820-0214 (Print) 2406-1018 (Online) = Computer
Science and Information Systems
COBISS.SR-ID 112261644

Cover design: V. Štavljanin
Printed by: Sigra star, Belgrade