Com
SIS

Computer Science and Information Systems

Vol 7, No 3, June 2010

Computer Science and Information Systems

Published by ComSIS Consortium

Volume 7, Number 3
June 2010

ComSIS is an international journal published by the ComSIS Consortium

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

Volume 7, Number 3, 2010
Novi Sad

Computer Science and Information Systems

ISSN: 1820-0214

ComSIS Journal is sponsored by:

Ministry of Science and Technological Development of Republic of Serbia -
http://www.nauka.gov.rs/lat

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. ComSIS also welcomes surveys papers that contribute to the understanding of emerging and important fields of computer science. Regular columns of the journal cover reviews of newly published books, presentations of selected PhD and master theses, as well as information on forthcoming professional meetings. 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. The acceptance rate so far was approximately 40%. ComSIS is published semiannually.

Indexing Information

ComSIS is indexed and abstracted in the following:

- Science Citation Index Expanded (also known as SciSearch) and Journal Citation Reports / Science Edition by Thomson Reuters,
- Computer Science Bibliography, University of Trier (DBLP),
- EMBASE (Elsevier),
- Scopus (Elsevier),
- Summon (Serials Solutions),
- Selected for coverage in EBSCO bibliographic databases,
- Selected for coverage in IET bibliographic database Inspec, starting with 2009 published material,
- Selected for coverage in FIZ Karlsruhe bibliographic database io-port,
- Google Scholar,
- Center for Evaluation in Education and Science and Ministry of Science of Republic of Serbia (CEON) in cooperation with the National Library of Serbia,
- 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 one of the Editors.

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. There is no page limit on manuscripts submitted.
Manuscripts will be refereed in the manner customary with scientific journals before being accepted for publication.

Copyright and Use Agreement

All authors are requested to sign the "Transfer of Copyright" agreement before the paper may be published. The copyright transfer covers the exclusive rights to reproduce and distribute the paper, including reprints, photographic reproductions, microform, electronic form, or any other reproductions of similar nature and translations. Authors are responsible for obtaining from the copyright holder permission to reproduce the paper or any part of it, for which copyright exists.

Computer Science and Information Systems

Volume 7, Number 3, June 2010

CONTENTS

EDITORIAL

Despite the current issue being labeled with no. 3, it is actually the first regular issue of ComSIS published this year. Due to increasing interest in our journal, the two previous issues were special issues devoted to publishing extended articles selected from proceedings of international conferences. Besides regular papers, the current issue also contains six articles selected for extension and publication from proceedings of IDC 2009, the Third International Symposium on Intelligent Distributed Computing. We would like to use this opportunity to thank the guest editor of this issue, Costin Bădică, who co-initiated and co-organized IDC, for helping to include high-quality contributions to this issue of ComSIS. Also, we are pleased to announce that ComSIS has been included in the Index of Information Systems Journals, maintained by Deakin University, Australia.

The first of the regular papers in this issue is “The Integration of Learning Object Repositories and Learning Management Systems” by Krešimir Fertalj, Nataša-Hoić Božić and Hrvoje Jerković, which addresses the problem of integration of content from federated e-learning repositories into courses included in a Learning Management System (LMS). The article analyzes current repository frameworks, placing emphasis on Flexible Extensible Digital Repository Object Architecture (FEDORA), and presents a pilot application which demonstrated how the interaction between a repository and LMS can be effectively implemented.

Next, Igor Dejanović, Gordana Milosavljević, Branko Perišić and Maja Tumbas, in their paper “A Domain-Specific Language for Defining Static Structure of Database Applications,” present DOMMLite - an extensible domain-specific language (DSL) for static structure definition of database-oriented applications. DOMMLite incorporates the model-driven engineering (MDE) approach, with the language structure defined using the openArchitectureWare framework, and DSL execution semantics defined through source code generation for target platforms. The DSL is capable of generating source code for GUI forms with Create-Read-Update-Delete-Search (CRUDS) and navigation operations.

In “Prompt System Redesign: Shifting to Open Source Technology to Satisfy User Requirements,” Igor Svetel, Aleksandar Đurović and Vencislav Grabulov describes a redesign project undertaken in a short period to adapt a software system to user needs and shift it to Open Source software, as is was recognized as the technology that would enable sustainable system development. The paper chronologically describes all phases of the project and provides reasons for all decisions taken during the development process.

Motivated by the observed isolation of testing techniques within specific lifecycle phases or functional areas, “Software Testing Optimization by Advanced Quantitative Defect Management” by Ljubomir Lazić presents a set of best practice models and techniques integrated in optimized and quantitatively managed software testing process (OptimalSQM) throughout the software lifecycle. The article also discusses how the Quantitative Defect Management (QDM) Model can be enhanced to be practically useful for determining the priority of activities leading to early and cost-effective software fault detection.

Dragan Mišić, Dragan Domazet, Miroslav Trajanović, Miodrag Manić and Milan Zdravković, in their article “Concept of the Exception Handling System for Manufacturing Business Processes,” tackle the issue of exceptions in predefined workflows, which can appear in many business process scenarios, and should be handled automatically, if possible, by adapting the workflow to the new situation. The paper presents a workflow management system MD, which offers a solution to the problem of handling detected exceptions.

The paper “Reasoning With Linguistic Preferences Using NPN Logic” by Goran Devedžić, Danijela Milošević, Lozica Ivanović, Dragan Adamović and Miodrag Manić surveys the basics of negative-positive-neutral (NPN) logic and relations, and proposes an adaptive approach to causality weights assessment which involves linguistic models and fuzzy cognitive maps. Particular attention was paid to modeling possible side effects, since they were identified as having particular importance for successful decision making in real-world environments.

Finally, Živko Bojović, Emil Šećerov and Vlado Delić, in their article “QoS Testing in a Live Private IP MPLS Network with CoS Implemented,” describe testing conducted on a private IP/MPLS network of a Telecom operator during service introduction, motivated by establishing a basis for defining a stochastic traffic generator/simulator. Testing involved DiffServ and E-LSP policies for bandwidth allocation, traffic generation for creating worst-case scenarios, and measurement of QoS for individual services (voice, video, data and VPN).

On behalf of the Editorial Board and the ComSIS Consortium, we would like to thank all authors and reviewers for their high-quality contributions and efforts expended in preparing this issue of ComSIS.

Editor-in-Chief
Mirjana Ivanović

M. Ivanović

Managing Editor
Miloš Radovanović

[illegible]

GUEST EDITOR'S MESSAGE

The increasing complexity of real-world problems demands for special support for cooperative problem solving in distributed environments. Recent advances in Multi-Agent Systems, Artificial Intelligence, and Computational Intelligence set the premises for the development of a new generation of Intelligent Agent-Based Cooperative Systems. These systems are composed of autonomous, non-antagonistic and social agents equipped with suitable interaction protocols and strategies aimed at improving the social outcome of the system.

IDC'2009, the 3rd International Symposium on Intelligent Distributed Computing, was held in Cyprus, Ayia Napa, October 13-14, 2009. From 36 accepted submissions, 6 articles that address highly relevant topics for the advancement of the field of Intelligent Agent-Based Cooperative Systems were selected and invited for extension and publication in this Journal issue.

The article "Cost of Cooperation for Scheduling Meetings" by Alon Grubshtein and Amnon Meisels introduces a new measure called Cost of Cooperation (CoC) for meetings scheduling games (MSG) that is useful for motivating selfish agents to cooperate. Using CoC, authors define a new game property called "Cooperation game" according to which participants may be better off cooperating rather than playing selfishly.

The article "TEAMLOG in action: a case study in teamwork" by Barbara Dunin-Kęplicz, Rineke Verbrugge, and Michał Ślizak presents an application of authors' recent formalism termed TEAMLOG for formal modelling of teamwork in multi-agent systems, to model an agent system for cleaning ecological disasters.

The article "A Case Study on Availability of Sensor Data in Agent Cooperation" by Christian Johansson, Fredrik Wernstedt, and Paul Davidsson evaluates the effects of practical limitations in the availability and quality of sensor data onto the effectiveness of agents' cooperative behaviour in a multi-agent system for Demand Side Management on the energy market.

The article "A Layered Rule-Based Architecture for Approximate Knowledge Fusion" by Barbara Dunin-Keplicz, Linh Anh Nguyen, and Andrzej Szałas proposes a rule-based framework based on a Horn subset of propositional dynamic logic (HSPDL) for fusing approximate knowledge from distributed and heterogeneous knowledge sources.

The article "Emergent Properties for Data Distribution in a Cognitive MAS" by Andrei Olaru, Adina Magda Florea, and Cristian Graţie introduces a data management system of cooperative cognitive agents that exhibits emergent properties for data distribution and replication.

The article “Distributed Parameter Tuning for Genetic Algorithms” by David F. Barrero, Antonio González-Pardo, David Camacho, and Maria D. R-Moreno introduces a system of cooperating agents for distributed parameter tuning in learning regular expressions using Genetic Algorithms.

Acknowledgements

Great thanks are first given to ComSIS and its Editor-in-Chief, Mirjana Ivanović who kindly offered the publication of this special selection. We would also like to thank George Angelos Papadopoulos and Konstantinos Kakousis from Department of Computer Science, University of Cyprus, Cyprus who greatly helped us with hosting and organizing IDC’2009. In addition, we would like to thank all the reviewers for their restless reviewing effort and valuable feedback and all the authors who submitted papers to this issue.

Costin Bădică received the M.Sc. and Ph.D. in Computer Science from University of Craiova, Romania in 1990 and 1999, respectively. In 2006 he also received the title of Professor of Computer Science from University of Craiova. He is currently with the Department of Software Engineering, Faculty of Automatics, Computers and Electronics of the University of Craiova, Romania. During 2001 and 2002 he was Post-Doctoral Fellow with the Department of Computer Science, King’s College London, United Kingdom. His current research interests are at the intersection of Artificial Intelligence, Distributed Systems and Software Engineering, including applications of multi-agent systems, information extraction, and formal modeling of business processes. He has authored and co-authored more than 100 publications related to these topics as journal articles, book chapters and conference papers. He has prepared special journal issues and co-edited four books in Springer’s Studies in Computational Intelligence series. He co-initiated and he is co-organizing the Intelligent Distributed Computing – IDC series of conferences that is being held yearly. He also served as PC member of many international conferences.

Costin Bădică
Guest Editor

UDC 004.451.9, DOI:10.2298/CSIS081127001F

The Integration of Learning Object Repositories and Learning Management Systems

Krešimir Fertalj[1], Nataša Hoić-Božić[2], and Hrvoje Jerković[3]

[1]Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, HR-10000 Zagreb, Croatia
kresimir.fertalj@fer.hr
[2] Department of Informatics, University of Rijeka, Omladinska 14, HR-51000 Rijeka, Croatia
natasa.hoic@ri.t-com.hr
[3]Zagreb School of Economics and Management
Jordanovac 110, HR-10000 Zagreb, Croatia
hrvoje.jerkovic@zsem.hr

Abstract. The systems aimed for manipulating large number of courses and students are called Learning Management Systems (LMS). A LMS can have excellent performance implemented through advanced Web technologies but it is often accompanied by a poor or rarely used repository of institution's educational content. It has still remained. Still remains a problem how to allow users of a LMS to easily modify and integrate the content from federated e-learning repositories into their courses. This article presents an analysis of present repository frameworks and projects. FEDORA (Flexible Extensible Digital Repository Object Architecture) framework is explained as an alternative repository solution. A pilot application has been developed to demonstrate the interaction between a LMS and its repository.

Keywords: learning object repositories, learning management systems, learning objects

1. Introduction

One globally accepted way to implement e-learning is by using a Learning Management System (LMS) as all-in-one system for online education, which covers registration, administering and monitoring of users and content. LMS also provides other tools such as assessment tools, discussion forums, grading tools and so forth. Therefore, it has become an irreplaceable solution for modern universities and other educational institutions. Blackboard is the leading commercial provider of LMS solutions and Moodle is a widely accepted open source solution [1]. Generally, LMSs support exporting of learning content to other systems but collaboration with external e-learning repositories is not standardized yet. On the other hand, solutions for e-learning repositories are advancing, offering federated sophisticated searches of learning objects through a network of repositories [3]. The term "learning object" (LO) is not intended to be restrictive but refers to any digital asset which can be used to enable teaching or learning [26].

So on the one hand we have intensively used LMSs and on the other hand there are still poorly used e-learning repositories. Many projects tried to define integration between LMSs and federated repositories but none has provided a widely accepted solution.

The main goal of research presented in this paper is to propose an effective integration between LMSs and e-learning repositories. Realization of such a project could allow university staff to easily access e-learning repository, search, create, annotate, share, compose, decompose and modify content without any special knowledge of web technologies. It is also important that the repository should not function only at the institutional level but it should also have the ability to interact with other repositories as a part of federated network of repositories.

Our principle guideline for research and analysis of projects in the field is defined by the following question – are the present solutions capable or useful for the needed complete integration between LMS and LO repository (LOR) or a network of LORs?

Our research of the projects in the field goes in three main directions:

- LORs and their functionalities

Many e-learning repositories have already joined the federated networks of repositories so in this paper we shall present the main LORs and analyze their capability for integration with LMS.

- Projects focused on integration between LMSs and LORs

Projects in this field are mainly focused on creating plug-in applications that would allow interaction between LMSs and LORs.

- Projects focused on decomposition, reuse and exchange of LO

Diverse projects and existent frameworks have tried to find a way to successfully preserve data in the form of digital objects that could be easily imported, decomposed and recreated into a new object. We shall analyze these projects to see how they can be used for achieving successful manipulation over e-learning content. An analysis of all the three categories is presented in Section 2. The last part of the Section 2 also covers a possible integration of main projects in all three fields mentioned in Section 2.

In Section 3 we propose a proprietary solution by introducing FEDORA [2] repository framework based on data-service oriented architecture. We show how FEDORA can serve as a LOR by examining of the core FEDORA platform features. We explain how it is possible for any LMS to communicate with FEDORA. We also show possible manipulations over LO stored in FEDORA.

For demonstration purposes we have built a pilot application for creating lessons out of LOs stored in FEDORA LOR. In Section 4 we describe the functionality of our application. Testing and evaluation of our application is described in Section 5 and conclusion of this paper is given in Section 6.

2. Research of projects in the field

2.1. LO repositories and actual projects in field

MERLOT, PALOMA, EDNA and ARIADNE LOR [3] are some of the prominent LORs. Since many of them are using different metadata schemes to describe the content stored in the repository, research in the field of e-learning repositories is mainly focused on interoperability between LO repositories. An Application Program Interface (API) was established for querying through Simple Query Interface search (SQI) with special focus on issues related to a common metadata schema and to a common query language [5]. SQI Registry tried to establish interoperability among different LO repositories. It is basically an UDDI (Universal Description, Discovery and Integration) [12] registry, which maintains a list of SQI targets (repositories) that can be queried using SQI. PROLEARN Network of Excellence [13] is coordinating some of the practical experimentation with these specifications and many important repositories already became part of SQI Registry. The GLOBE (Global Learning Objects Brokered Exchange) project is an international effort to create federated search engine over distributed LORs for searching e-learning content [24]. Through GLOBE interface, SQI Registry repositories can be searched with standard web search. An attempt to integrate LMSs into federated repositories was made by the ARIADNE group within the GLOBE project through the development of a tool which is a plug-in for Moodle LMS [4]. The tool allows users just to search, retrieve and store a LO as a local object.

The main SCORM (Sharable Content Object Reference Model) [6] attempt aimed at providing such shareable LO environment is called CORDRA (Content Object Repository Discovery and Registration Architecture) [6]. CORDRA is a highly complex system that is still under development and has not yet been adopted by any of the educational institutions. Basic start up goal is to connect US government departments and agencies, and ADL (Advanced Distributed Learning) laboratories in order to create a network of content which would eventually become widely used.

Another attempt in the field is EduSource Communication Layer (ECL) [7]. The ECL Gateway [7] is a middleware framework that enables building bridges to other protocols easily. It is used to develop bridges to several other protocols and networks such as OAI (Open Archive Initiative) [16], SRW/SRU (Search and Retrieve Web services/ Search and Retrieve URL) [17], Resource Discovery Network in UK [18], EdNA (Australia), SMETE (USA) [19], LionShare (Gnutella based P2P network) [20] and also SQI. ECL is one of the first implementations of the IMS Digital Repositories Interoperability (IMS DRI) [7] specification. IMS DRI will allow repositories not only to share, search, and import results but also to gather records, alter each other according to new materials and submit new materials in other repositories [14].

In comparison with SQI, ECL has further broadened its goal, trying to connect practically every repository that could be found, including those in P2P networks and private users. Large number of universities and other institutions are using ECL framework. All connected networks are registered in the ECL registry and are available to ECL users. ECL registry is implemented in JUDDI (an implementation of standard UDDI) [21]. All previously named implementations are focused on interoperability among LORs but they have poor integration with existing LMS systems. The GLOBE tool for example does not allow LMS users to change a LO, store it as a new LO, subscribe to changes in a LO, nor does it provide easy integration with present content through any form of referencing service. ECL on the other hand has many of these features but each of them is tied to a certain exchange protocol upon which a given repository functions, without tools for integration with LMSs.

There is another institution that is doing a lot of research in the related field of library archives - Open Archives Initiative has started Object Reuse and Exchange (OAI-ORE) [9] which defines standards for the description and exchange of aggregations of web resources. The goal of these standards is to expose aggregations of rich content to applications that support authoring, deposit, exchange, visualization, reuse and preservation [9]. The OAI already produced a protocol very similar to SQI - Protocol for Metadata Harvesting (OAI-PMH) [9], as a mechanism for repository interoperability. It consists of Data Providers, repositories that expose structured metadata via OAI-PMH, and Service Providers that make OAI-PMH service requests to harvest or manipulate that metadata. OPI-PMH is widely used, highly valued and continually developed, which cannot be said for the network of LORs based on SQI and used by the academic community. A large number of frequently used institutional repositories and projects are based on OAI-PMH, while LO reuse through LORs is not currently in mainstream focus even among teachers [10]. While most LORs have been in operation for several years (MERLOT: 7 years, SMETE: 5 years, ARIADNE: 7 years), the amount of learning objects indexed in any one of them is small and it is comparable in number to the amount of learning objects stored in a single medium-sized LMS [10].

2.2. LMS-LOR integration

The best solution for users would be if their needs could be met in their native environment, and that is their own institutional LMS, which should allow them to access the LOR from the LMS with possibility to create, modify and integrate the content.

The main effort in this field is undertaken by the EU-funded LUISA [11] project. The goal of the project was to create a rich flexible infrastructure supporting the development and reuse of learning materials for both learners and educators, and also the integration of LOR with a Learning Content Management System (LCMS). Therefore, the whole software solution is built on Moodle, and a plug-in for Moodle is available that allows for searching the LUISA LOR, based on the ontology (encoded in RDF), which is the core of the LUISA project. The LUISA solution relies on Semantic Web Services that can apply user profile, topics and competencies to resolve the required set of LOs for the given user. Although the LUISA architecture allows for the integration and interaction with other web applications, these applications are still not developed. A set of tools should be developed to prove that LUISA's services can be integrated into any LMS environment, which has until now been achieved only for Moodle.

Swiss Virtual Campus project team tried to achieve similar integration between three types of LMS (WebCT Vista, Moodle and OLAT (Online Learning and Training LMS)). Main shortcomings of that project were that the LOR used in the project had not been designed to work as a part of federated network of LORs and the created applications did not allow modifications of LO from LMS but only an upload and integration after an object had been created and imported [25].

ARIADNE, as one of the leading institutions in the field states that "a huge number of LMSs exists today and institutions often use different kinds of LMS. Thus, ARIADNE decided to focus on its LOR and to build an API dedicated to the specific LMS." So far integration between ARIADNE Federated network of LORs and LMS is achieved only for Moodle LMS and for the less used LMS INES [27]. ARIADNE solution cannot easily be integrated into any LMS environment, besides; every subsequent version of an LMS might require modifications of application for integration with the LMS as well.

Obviously these projects have not achieved the practical goal of the LMS-LOR integration - to have a set of web service applications accessible from any LMS for authoring lessons (LOs) on the interface of LOR, or a LOR federated network.

2.3. Frameworks for decomposition and reusability of LO

Many repositories store complex and large amounts of data as single LO. Under these circumstances, LOs are losing their original meaning. Therefore, a decomposition of LOs for reusability with a proper meta description is the main goal of projects in this area.

Examples of such projects can be found in [14] where authors propose a repository of SCORM-described content packages, which are actually complete courses. On the other hand in [15] is discussed how repositories can be created by decomposing SCORM content packages in order to create customizable learning content. It is obvious that propositions are not satisfactory for users, mainly because complete courses cannot be accessed and adopted into the new LMS environment easily.

By decomposing SCORM packages, users can get only basic raw resources out of them, which then require a special authoring system for assembling that data into customized courses and another system for the proper meta description.

This problem had been recognized by the academic community and therefore a number of projects have started in order to achieve effective ontology-based decomposition and integration of LO. The main projects in this area will be discussed further in the text.

RAMLET (Resource Aggregation Model for Learning, Education and Training) [8] is a project that is focused on retrieving, interpreting, disaggregating, (re)aggregating and deploying content in LMSs from different types of structural elements available in specialized repositories. RAMLET is a conceptual model, expressed as a human readable table and a set of ontologies that specifies how this aggregation formats map to each other via a single core model.

Unfortunately, RAMLET's ontology has aggregating purposes and it is not concerned with the relationships between sections and chapters or with the interactive properties of different media types. Also RAMLET is a one way design offering no possibility for changing LO in the original repository. The RAMLET model is complete, but has not been implemented yet.

Another project that tried to cover that field is ALOCOM (Abstract Learning Object Content Model) [8], which provides a generic content model that defines a framework for LOs and their components. The ontology defines concepts representing different LO component types and their structure. This explicit definition of both content and structure enables the disaggregation of LOs into their components. In [23], the authors explain similar project in which "the use of the XML technology allows, on one hand, content structures to be defined at different levels of abstraction, and, on the other hand, content to be separated from presentation."

The authors in [24] propose the use of domain ontologies to annotate LO content as well as content structure ontologies to enable direct access to LOs' components. Basic goal in this case is that "same LO can be used in different ways and by different users, that is, it can be repurposed" [24].

It is obvious that there are many propositions and projects that are being developed by leveraging ontologies as a common knowledge representation format. As much as the development of new abilities through new ontologies is welcomed, the problem of interoperability of repositories and applications grows in case where there is no agreement on the usage of certain ontology.

ALOCOM plug-ins are available for MS Word and MS PowerPoint allowing users to search for text, graphs, pictures and other elements of decomposed MS Word and MS PowerPoint documents stored in the ARIADNE repository. However, ALOCOM ontology is not a part of the ARIADNE core tools and it is not functional on the whole SQI Registry [8]. In fact ARIADNE, so far, offers integration of SQI search with GLOBE tool only for Moodle LMS but including ALOCOM features. Apart from offering search functionality through PowerPoint, this plug-in offers an automatic generation of presentation out of objects stored in ALOCOM-based repositories [8]. It is important to note that

ALOCOM is an abstract data model, not a tool. There are plug-ins developed on top of this data model. A prototype of a standalone tool was available before the plug-ins were developed, but it was just for demonstration and testing purposes.

RAMLET, on the other hand, is focused on allowing users to search through different structural elements, enabling them to create a new content base. It can be implemented in different ways, depending on the used ontology and available metadata. If used with ALOCOM by ARIADNE, it is obvious that it will be tightly connected to the ARIADNE metadata mandatory fields as the probable starting point for developing other semantic ontology-based applications.

So it is likely that the RAMLET project will complement the work being undertaken within and around the OAI-ORE [22]. In our opinion RAMLET can also produce significant results if combined with ALOCOM in ARIADNE repositories but ALOCOM ontology is still not widely accepted nor completely implemented even in SQI federated repositories.

The main goal in our opinion should be to allow a user to retrieve LOs by searching through federated repositories, with the ability to modify those objects and compose lessons out of them. But the problem at hand is obvious - without a common nomenclature and a conceptual model to allow for the interpretation of these formats and specifications, it is difficult to create applications that can interoperate.

2.4. Possible LMS-LOR integration through existing projects

Considering the previously stated, the design of a system will be proposed that can be built upon all the projects mentioned here. LUISA is a good basis for LMS-LOR integration but its ontology should be improved through other projects in this field. For example, the parsing, categorizing and annotation of LOs could be complemented with OAI-ORE decomposition in combination with ALOCOM ontology principles.

The conclusion is that the majority of the quality content resides in LMSs. Simple Automatic Metadata Generation Interface (SAMGI) [10] project shows that many valuable metadata fields can be extracted from this content. So SAMGI can be used for metadata extraction and RAMLET could be used for the composition of LOs from different repositories. SAMGI would be a good tool to refine results of the searches, whereas the ECL protocol could be the best solution to achieve interoperability across different federated networks of repositories. RAMLET complemented with OAI-ORE, with stronger semantic web services integration points, could allow for the development of a number of distributed applications for rich user-content and user-user interaction. The key point at this moment is not the technology but the agreement on usage of a common ontology, metadata schemas and semantics in applications and repository implementations.

3. Architecture of data-service oriented digital repository

The previous chapter described a number of projects that could offer an effective solution for LMS-LOR integration but also pointed out many shortcomings.

The main problems with existing solutions are following:

- Inability to search federated LORs through any LMS.

Present solutions only partially solve this problem: the GLOBE tool enables searching only through SQI federated repositories; ALOCOM-based solution facilitates search of decomposed LOs but only with desktop application not through LMS; ECL connects almost all known types of repositories but without integration with LMS.

- Inability to modify and directly integrate content in any LMS.

Platforms like RAMLET and ALOCOM exist, but they are not properly implemented nor do they allow easy implementation of such features in any LMS.

- Lack of sophisticated authoring tools for creating lessons out of modified or stored content in federated LORs.

LUISA offered a customized LMS with interoperable platform specialized for searching LORs, but it still does not have any similar authoring tool features.

In what follows we explain how FEDORA platform can be used as a web service oriented platform, integrated with existing solutions or used as an LMS internal LOR, connected to a network of LORs. We explain the advantages of architecture like FEDORA over the repositories mentioned in the previous chapters. FEDORA will be described because it represents an ideal platform for the development of a LOR that can be easily integrated into any LMS.

3.1. The FEDORA Platform

FEDORA is project based on OAI-PMH, so in its core there is an elementary system of data and service providers. The FEDORA platform enables flexible, application oriented architecture for the management and delivery of digital content. It has a powerful digital model for storing objects, which supports different ways of manipulation, editing and establishing relations over and among objects. FEDORA is actually designed for archiving of bibliography records but our goal is to show how such a powerful platform can be used in an e-learning environment.

The architecture of a FEDORA repository is shown in Fig. 1. Users can access the LO repository via web browser, client application or a third party application which communicates with frontend web services. In this way different functionalities can be achieved like searching, modifying, and adding digital objects, linking objects, retrieving metadata or whole objects etc. An example of such application is given later in this paper. A LO in FEDORA platform consists of three main groups of data as shown in Fig. 1.

1. Metadata:

- Description metadata: used for correct description of a LO.

Metadata define the purpose, type, and possible educational usage of the respective object. Any metadata scheme can be used in FEDORA [5]. XML is the native FEDORA format for data interchange and services interactions. Any given metadata schemas and ontology references can be added to single or multiple LOs inside the repository.

- Relational metadata: definition of relations among objects, relations

between object and external content and definition of parameters needed for application manipulation.

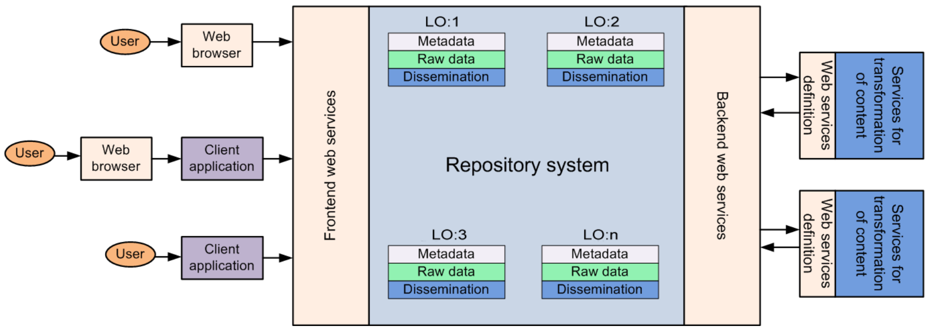


Fig. 1. Architecture of the system for digital objects storage

Relational metadata is actually a great basis for sophisticated ontology implementation. Any relations among objects can be stored even if an object is on another server, so projects like SAMGI can easily complement FEDORA for the purpose of harvesting metadata from a LMS.

2. Raw data: all stored data supported by a LO repository. Raw data is the term used for all Multipurpose Internet Mail Extensions (MIME) [10] types of data or data without metadata attached to it.

3. Dissemination data: used for calling remote or local application for management of objects stored in LO repository. Examples of applications that require dissemination are, for instance, application for picture manipulation and for automatic transformation of certain types of content into the PDF format. Since most of the applications used by repository are web applications, dissemination data has to contain the source and format definitions of the data that needs to be transformed, and also the location and parameters needed for calling the web service over the REST [6] method or SOAP [7] protocol.

Web services on the frontend of the system are used for communication with other services and applications and for calling backend web services for transformation of content. Backend service also serves for interaction with other repositories, for maintaining and indexing of objects.

An advanced example of content manipulation by third party web service is shown in Fig. 2. [2]. The client requests HTML file of a poem stored in a LO repository. The repository may contain only an XML version of the requested file, but the LO which contains the XML data can also contain dissemination data for calling a web service.

The called web service can be an internal web application, which is usually the case, but in FEDORA it can also be an external web application from different web server like in the Fig. 2. In FEDORA, a LO in the repository does not even have to contain dissemination data but it can refer to another LO that contains the needed dissemination data for the required type of transformation (i.e., XSL file which defines transformation, in this case from XML to HTML). It is obvious that this kind of architecture leaves a lot of space for integration of the repository into any kind of system without the need for a specific platform application or plug-ins like it is the case with GLOBE tool or ALOCOM.

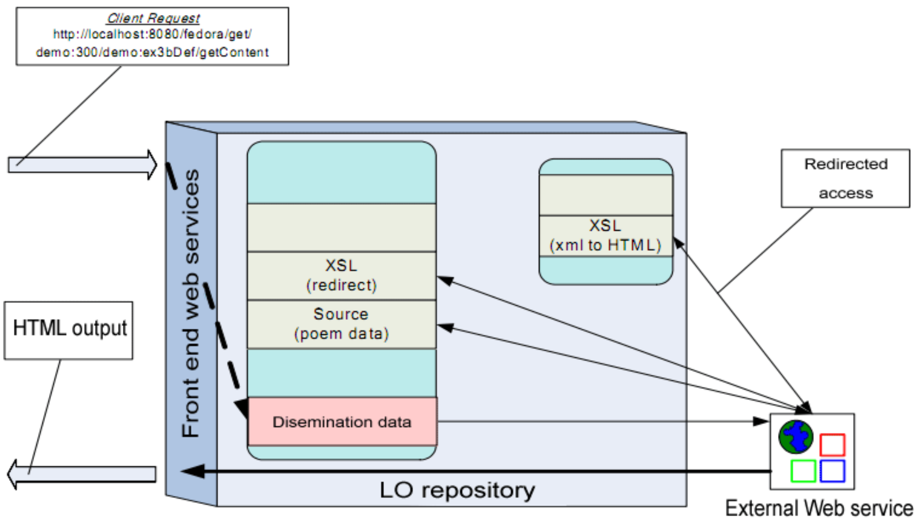


Fig. 2. Calling external Web service for LO transformations

Any given ontology, semantics and application can be applied in FEDORA. Each LO is actually encapsulated inside the object. Semantic value of the LO can grow over time because it can store information from every LMS, repository and system that has accessed or modified it. FEDORA has a versioning system that allows for saving different versions of the same LO.

From the given information, it is obvious that LOs inside FEDORA are totally platform independent units. Any information needed for a LO to be manipulated and retrieved is stored in the LO itself. LO stores in itself ontology and metadata information but also data needed by the web services for exchange, metadata harvesting or manipulation of data stored in the LO. Therefore looking from technical perspective, the LO is an independent unit which contains all the data needed for communication with other systems and programs.

3.2. Communication of an LMS and a LOR

FEDORA can interact with any LMS. Fig. 3. shows how two widely used LMSs, Moodle and WebCT, can interact with the same FEDORA LOR. Moodle and WebCT are taken just as an example. A number of other LMSs can be connected in this way to the same digital repository. The exact number depends on the characteristics of the network, server, repository and workload, which is beyond the scope of this paper.

In Fig. 3, FEDORA LOR can be replaced by a FEDORA network of repositories which can exchange metadata.

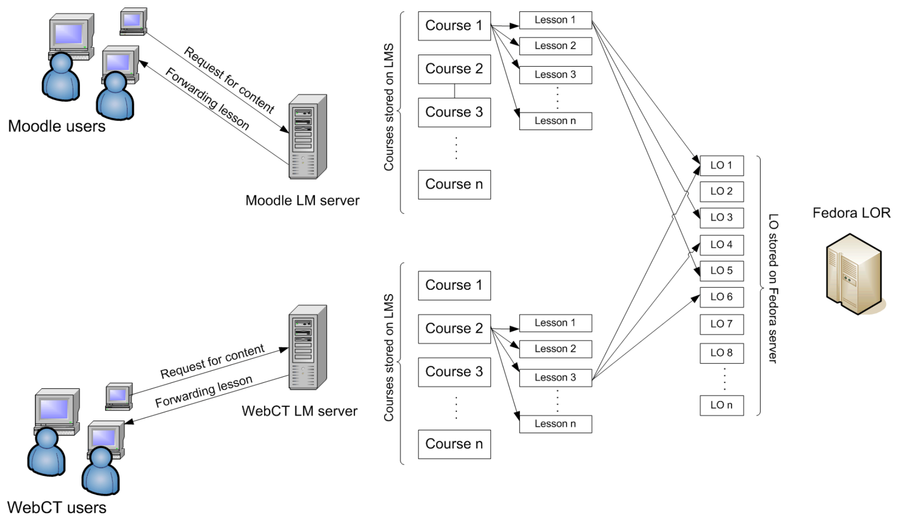


Fig. 3. Example of communication between LMSs and a digital repository

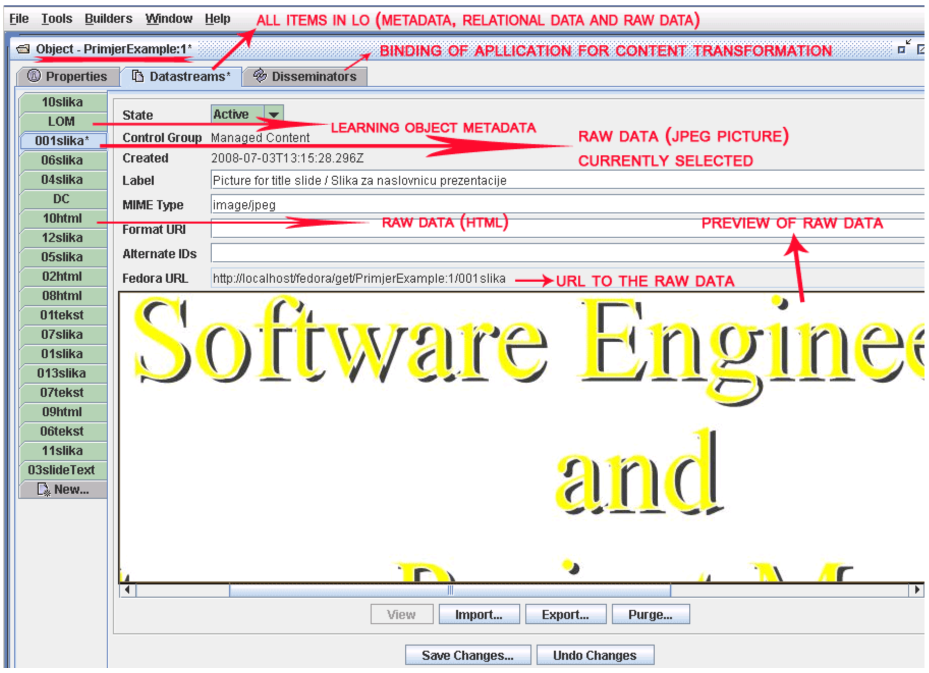
SQI federated repositories can also exchange data with FEDORA over FEDORA backend services and ECL supports the integration of OAI-PMH based frameworks so that interoperability with other repositories is achieved.

In the rest of the paper it will be shown how FEDORA LO can be referred to, modified and combined with another LO without being saved in the local repository of an LMS. Simple embedding application will be called from WebCT wrapper in the form of an external URL. Our application will manipulate LOs from FEDORA through FEDORA web service. The final product will be a URL with location of the lesson which in our case is a combination of LOs stored in FEDORA.

3.3. Manipulation of LO compositions in FEDORA

Because of the way in which FEDORA LO is defined, LMS objects can be stored inside LMS repository and can be manipulated through FEDORA. Fig. 4. shows the user interface of the administrator's backend and one LO stored with its Learning Object Metadata (LOM) [6] and its content items.

Fig. 4 shows all content items of the PrimjerExample:1 LO. The left hand side of the figure shows the list of content items and "001slika" (001picture) JPEG picture is presently the displayed item. Item details are shown for "001slika": MIME Type (image/jpeg) and the Fedora URL. Metadata (in this case LOM-based metadata) can also be stored for a content item and it is shown just above "001slika" content item. Metadata in this case are added manually but as described before, there are several ways for automating metadata descriptions. Each content item can be modified and each modification can be stored as a new content item accessible (by its own URL) and available for manipulation by third party web application over Internet. This also enables for easy storing of metadata from an LMS and harvesting of the same metadata by other systems

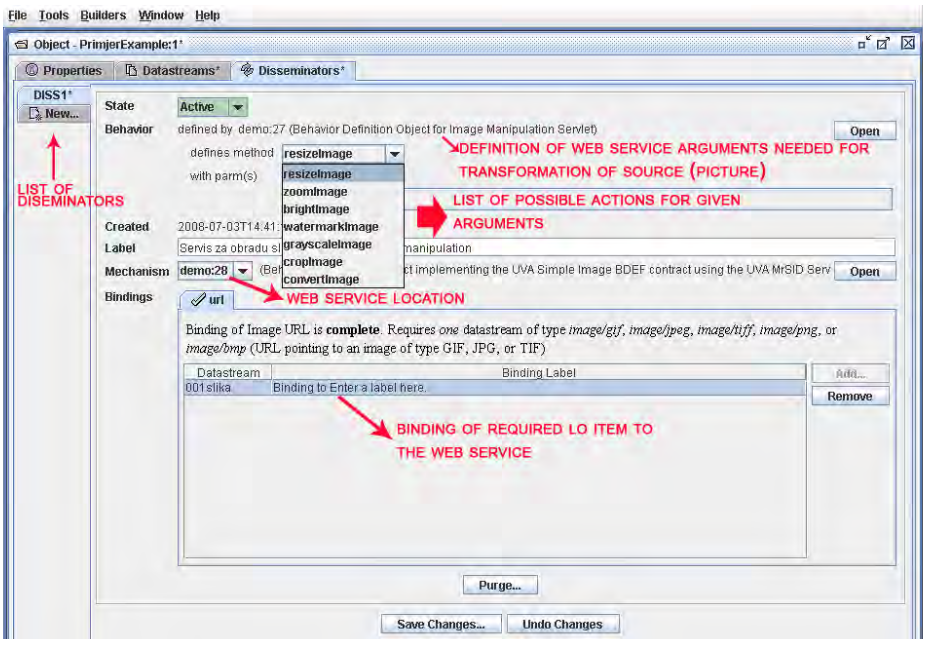


State: Active
Control Group: Managed Content
Created: 2008-07-03T13:15:28.296Z
Label: Picture for title slide / Slika za naslovnicu prezentacije
MIME Type: image/jpeg
Format URI:
Alternate IDs:
Fedora URL: http://localhost/fedora/get/PrimjerExample:1/001slika

Fig. 4. A screenshot of the administrator's backend with an example LO

Fig. 5. shows the definition of the dissemination part of a LO. This is not an end-user application but an administration backend for the definition of web service handlers for manipulation with a LO. Method definitions are automatically available according to the type of the content item, in this case a JPEG picture. According to the definitions of the chosen method (web service arguments), different mechanisms (web services) can be used for different manipulations. The respective URL is then available for calling the defined web service. For example, URL of web service for picture manipulation will be available when user is storing a picture.

So in our case Fig. 5 shows a web service connected with picture content item shown on Fig. 4. This web service (disseminator) is used for seven possible picture manipulations which are shown in the figure (resizeImage, zoomImage, brightImage, watermarkImage, grayscaleImage, cropImage and convertImage). It is important to emphasize that dissemination data (data needed for calling web services) is an elementary part of a LO just as any other content item or raw data. Dissemination applications (web applications), as described before, can be called from other remote server applications. This kind of architecture provides maximum interoperability for content and functionalities. Every part of a LO can be changed and stored as a new version. Users can keep or change old versions. They can also subscribe to a desired version of a LO, so any change of the LO in repository will also reflect on all the systems that integrate that LO.


File Tools Builders Window Help
Object - PrimjerExample:1*
Properties
Datastreams*
Disseminators*
DISS1*
New...
LIST OF DISEMINATORS
DEFINITION OF WEB SERVICE ARGUMENTS NEEDED FOR TRANSFORMATION OF SOURCE (PICTURE)
LIST OF POSSIBLE ACTIONS FOR GIVEN ARGUMENTS
WEB SERVICE LOCATION
BINDING OF REQUIRED LO ITEM TO THE WEB SERVICE
Open
url
Add...
Remove
Purge...
Save Changes...
Undo Changes

State: Active

Behavior: defined by demo:27 (Behavior Definition Object for Image Manipulation Servlet)

defines method: resizeImage

with parm(s)

resizeImage
zoomImage
brightImage
watermarkImage
grayscaleImage
cropImage
convertImage

Created: 2008-07-03T14:41:

Label: Servis za obradu sl manipulation

Mechanism: demo:28 (Beh ct implementing the UVA Simple Image BDEF contract using the UVA MrSID Serv

Bindings

Binding of Image URL is **complete**. Requires *one* datastream of type *image/gif*, *image/jpeg*, *image/tiff*, *image/png*, or *image/bmp* (URL pointing to an image of type GIF, JPG, or TIF)

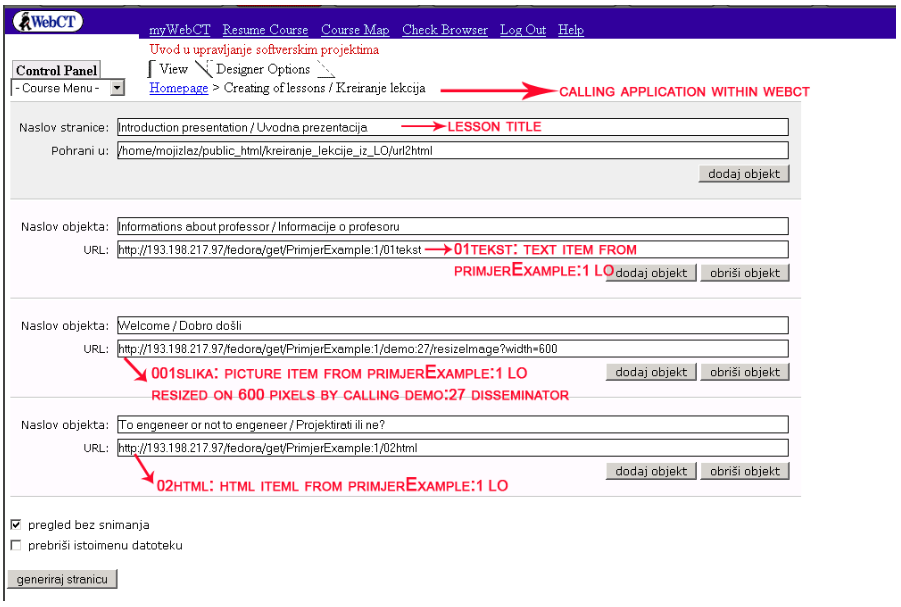
Datastream	Binding Label
001slika	Binding to Enter a label here.

Fig. 5. Definition of web service (dissemination data) for picture manipulation

4. Application for creating lessons from LO

Screenshot of our pilot application for creating lessons is shown in Fig. 6. The figure shows application integrated as a tool inside WebCT LMS. The application is installed on one web server, WebCT functions on another web server and FEDORA LOR functions on a third web server, so three independent web servers are present in this scenario.

Any designer in any WebCT course can start the application via URL from WebCT. User can define title of the lesson and URL of content item from FEDORA LOR. A simple lesson was build out of the stored content items of the PrimjerExample:1 LO. Our application lacks a search engine for searching and retrieving content item URL from FEDORA LOR or the network of LORs. That will be implemented in future versions.


WebCT
myWebCT Resume Course Course Map Check Browser Log Out Help
Control Panel
- Course Menu -
View
Designer Options
CALLING APPLICATION WITHIN WEBCT
LESSON TITLE
dodaj objekt
01TEKST: TEXT ITEM FROM PRIMJEREXAMPLE:1 LO
obriši objekt
001SLIKA: PICTURE ITEM FROM PRIMJEREXAMPLE:1 LO RESIZED ON 600 PIXELS BY CALLING DEMO:27 DISSEMINATOR
02HTML: HTML ITEML FROM PRIMJEREXAMPLE:1 LO
generiraj stranicu

Uvod u upravljanje softverskim projektima

Homepage > Creating of lessons / Kreiranje lekcija

Naslov stranice: Introduction presentation / Uvodna prezentacija

Pohrani u: /home/mojizlaz/public_html/kreiranje_lekcije_iz_LO/url2html

Naslov objekta: Informations about professor / Informacije o profesoru

URL: http://193.198.217.97/fedora/get/PrimjerExample:1/01tekst

Naslov objekta: Welcome / Dobro došli

URL: http://193.198.217.97/fedora/get/PrimjerExample:1/demo:27/resizeImage?width=600

Naslov objekta: To engeneer or not to engeneer / Projektirati ili ne?

URL: http://193.198.217.97/fedora/get/PrimjerExample:1/02html

☑ pregled bez snimanja

☐ prebriši istoimenu datoteku

Fig. 6. Screenshot of pilot application for creating lessons

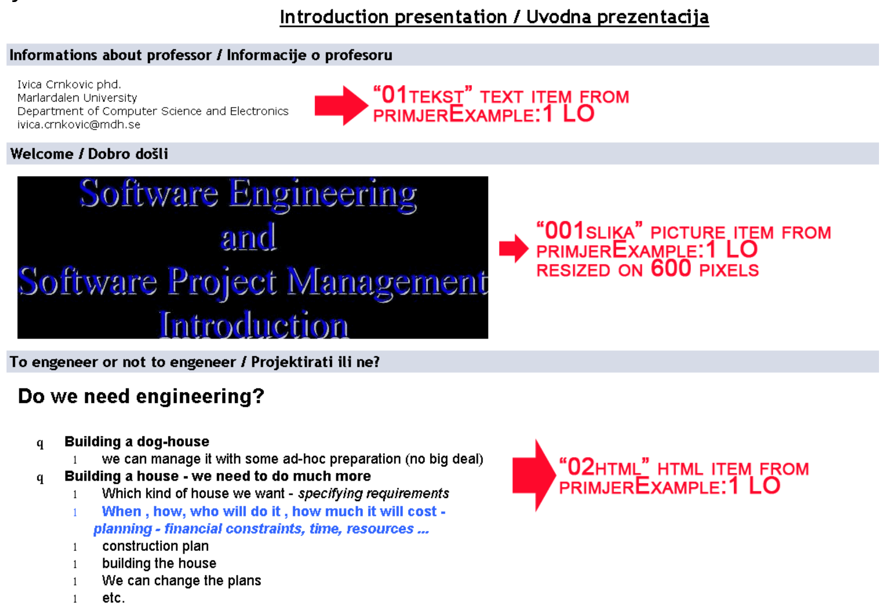
At this point, possibilities for the integration mentioned in the previous chapters will be demonstrated. The URL shown here can be easily obtained because FEDORA has a built-in search engine for searching of local storage.

The process of creating lessons is simple: a user searches FEDORA for LO. After finding the desired LO, the user can open that object inside FEDORA and just copy its corresponding FEDORA URL (see Fig. 4.). In our application, shown in Fig. 6, the user enters LO's title in the field "Naslov objekta" (Object Title) and pastes the corresponding FEDORA URL in the "URL" field. There is no limitation to the number of LOs that can be added in described way.

In our example, the generated lesson takes the form of a HTML page consisting of three content items from the PrimjerExample:1 LO: "01tekst" and “02html” content items (shown on the left hand side of Fig. 4) and “001slika” picture item (shown on the right hand side of Fig. 4).

After defining the URLs of content items, our application provides the designer with the URL of the whole created lesson. Final HTML simple lesson is shown in Fig. 7.

LO “001slika” shown in Fig.7 is a picture resized to 600 pixels, retrieved from a web service. Parameters and information about that web service are shown in Fig. 5. So when the user opens LO s/he can click on the disseminator tab (see Fig. 5.) and see what web services are available for that LO. After that, the user can define a desired manipulation over the object through web service interface, see Fig. 8. In our case, the specific disseminator that is being called is resizeImage with the parameter value set to 600. When the user sets desired parameters for the picture, and clicks “Run” next to it, an URL is automatically generated. After that, the user can just copy that URL, past it into our application and give it a name, just like for any other LO.



Introduction presentation / Uvodna prezentacija

Informations about professor / Informacije o profesoru

Ivica Crnkovic phd.
Marlardalen University
Department of Computer Science and Electronics
ivica.crnkovic@mdh.se

“01TEKST” TEXT ITEM FROM PRIMJEREXAMPLE:1 LO

Welcome / Dobro došli

Software Engineering
and
Software Project Management
Introduction

“001SLIKA” PICTURE ITEM FROM PRIMJEREXAMPLE:1 LO RESIZED ON 600 PIXELS

To engeneer or not to engeneer / Projektirati ili ne?

Do we need engineering?

- Building a dog-house
  - we can manage it with some ad-hoc preparation (no big deal)
- Building a house - we need to do much more
  - Which kind of house we want - *specifying requirements*
  - When , how, who will do it , how much it will cost - *planning - financial constraints, time, resources ...*
  - construction plan
  - building the house
  - We can change the plans
  - etc.

“02HTML” HTML ITEM FROM PRIMJEREXAMPLE:1 LO

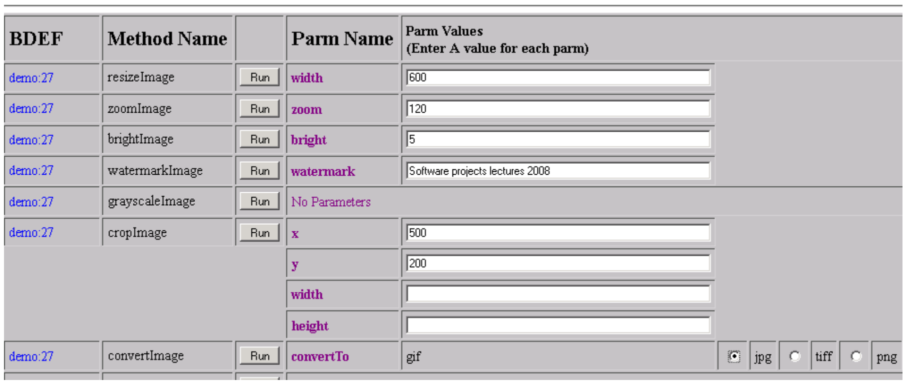
Fig.7. Lesson consisting of three content items reused from an existing LO

The web service used for generating the content for “001slika” is shown in Fig. 8, where all of its functionalities for picture manipulation are listed. User interface shown in Fig. 8. gives users the resulting URL with a modified picture after all the parameters have been defined. Specifically, in the given example the web service would fetch the original picture, change its maximum width to 600 pixels, zoom it to 120, change brightness to level 5, put watermark in the corner “Software projects lecture 2008”, crop image on 500 x 200 dimension and convert it to the GIF format.

Other similar features are available, for example if the user wishes to publish the PDF version of a document he or she can store the whole lesson as another LO and just put a link to the PDF disseminator anywhere in the LMS. The user can still modify a LO in the database and make new versions of the LO automatically available.

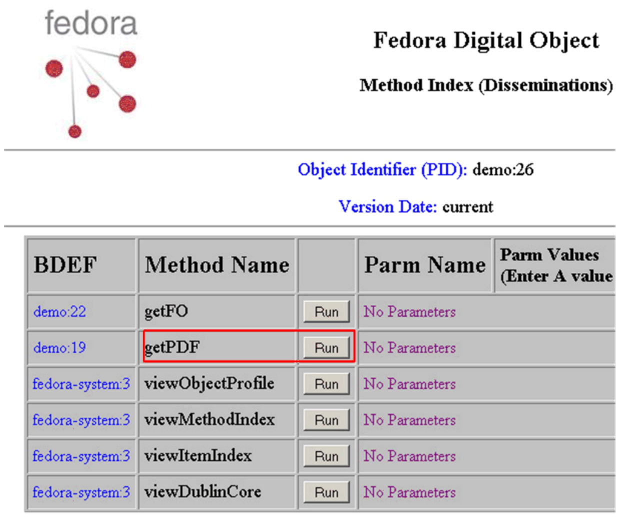
So when someone calls the PDF version of the LO, the disseminator application automatically collects the newest LO and makes an updated PDF document. Functionalities of the PDF disseminator are shown in Fig. 9.

Any other web service can be called and manipulated by user in the same way.



BDEF	Method Name		Parm Name	Parm Values (Enter A value for each parm)
demo:27	resizeImage	Run	width	600
demo:27	zoomImage	Run	zoom	120
demo:27	brightImage	Run	bright	5
demo:27	watermarkImage	Run	watermark	Software projects lectures 2008
demo:27	grayscaleImage	Run	No Parameters	
demo:27	cropImage	Run	x	500
			y	200
			width	
			height	
demo:27	convertImage	Run	convertTo	gif ● jpg ○ tiff ○ png

Fig. 8. Functionalities of the picture disseminator



fedora

Fedora Digital Object

Method Index (Disseminations)

Object Identifier (PID): demo:26

Version Date: current

BDEF	Method Name		Parm Name	Parm Values (Enter A value
demo:22	getFO	Run	No Parameters	
demo:19	getPDF	Run	No Parameters	
fedora-system:3	viewObjectProfile	Run	No Parameters	
fedora-system:3	viewMethodIndex	Run	No Parameters	
fedora-system:3	viewItemIndex	Run	No Parameters	
fedora-system:3	viewDublinCore	Run	No Parameters	

Fig. 9. Functionalities of the PDF disseminator

5. Evaluation and plan for improvements

The presented application is still under development at the Zagreb School of Economics and Management (ZSEM). Through this project, a part of an E-business elective course has already been developed. The application was installed on the School's web server and creating of lessons was tested on WebCT which is the School's mainly used LMS. An associate IT professor and several assistants at ZSEM have evaluated the application. Their feedback was collected through interviews after tool demonstration.

We have conducted qualitative research with a goal to test the usability of application in LMS in which users work, to realize advantages and shortcomings of the application and also to define the main fields for future development.

Evaluation participants have noted as main advantages of our application the ease of installation and almost unneeded maintenance because application doesn't need database support, ease of embedding in any LMS, possibility for integration on different portals and sites and not only to different LMSs.

They also noted that number of users is limited only by memory and processor power of web server and that application can be used from any Internet Browser without need for installation of additional plug-ins and also that data import and export, login and permissions can be implemented through LMS, LOR or even institutional Student Information System (SIS).

Disadvantages so far that participants have noticed is fact that users have to seek LO URLs in LOR and than copy-past each LO into application. Participants proposed an interface to support the following sequence of steps for creation of lessons:

- User seeks LO from LOR (or network of LORs) through application interface
- User uses check box option to check LOs that user would like to use for creating his own lesson.
- After picking desired set of LOs from repository, user will have ability to: edit, define desired sequence, assemble, and store new composition of LOs in database as new LO.
- User is offered new LO as complete lesson which can then be embedded.

Those comments were very useful and they will most probably be main guideline while developing future version of application.

We are planning to further develop application after the content from the faculty LMS - WebCT is disaggregated, described and transferred into FEDORA LOR. Development of the project for transfer of the course content is in preparation.

The proposed application is not yet suitable for end users. Further plans for development include:

- Interface for searching and retrieving content items

- A control panel for deployment of internal and external web services for multiple LOs in the repository
- User friendly interface for manipulating, modifying, composing and annotating of LOs through internal and external web services
- Integration of present solutions like SAMGL which will cover collecting of metadata from the LMS
- Integration of present solutions like ALOCOM or RAMLET which will cover decomposing and storing of LO into repository
- Definition of user roles and permissions
- Integration of web applications for user collaboration.

Possible problems may arise in case that a lesson consists of several LOs and the LMS designer (professor) wants to accept all possible changes of constituent LO from the repository. In case that LOs really change in time, a request for its constituent content items is sent every time a user opens the lesson (i.e. LO) in LMS. That can cause large workload on LO repository and increase network traffic between LMS and LO repository. Instead of that, the LMS can make a local copy of every lesson, cache it and periodically check for updates. In this way the overall network traffic and the LO repository workload can be more acceptable.

Another problem at the global level is that LMSs today don't apply the same rules for defining different user roles within an LMS. Security procedures should be standardized in the future. Permissions of LMS user over LO in repository should include strict rules considering authentication, accessing and modifying LOs from the repository. Otherwise it would become impossible to define standard communication for different types of users.

6. Conclusion

This paper presents an attempt to integrate federated LORs into LMSs as more and more popular e-learning environments of academic community. Many currently active projects that are covering related research fields are presented. Our conclusion is that the majority of these projects are not capable of delivering basic features to end users in any given LMS environment. We can illustrate this conclusion on a simple example. If a professor using any LMS environment wants to search for LOs through networked LORs and easily modify, integrate, annotate, combine and store them as new learning objects or new lessons, he or she would realize that there is no available system to allow this kind of interaction.

Another problem is that many federated LO repositories are hardly used, while at the same time LMSs are storing the majority of relevant LOs. Many projects in e-learning community are focused on achieving interoperability among different LORs, on ontology-based search over these repositories and on decomposition and reuse of LOs stored in such repositories. Therefore, this paper attempts to demonstrate how the architecture of data-service oriented digital repository can allow users to effectively interact through any LMS with a network of LORs.

We have explained how a LO stored in a digital repository of such architecture can be modified, combined and stored as a new lesson, independently of the user's LMS platform for accessing the repository.

References

1. A. O'Connor, S. Lawless, E. Walsh, V. P. Wade, Service Oriented E-Learning Platforms: From Monolithic Systems to Flexible Services, EEE Internet Computing, vol. 11, no. 3, pp. 28-35, 2007.
2. C. Lagoze, S. Payette, E. Shin, C. Wilper, Fedora: an architecture for complex objects and their relationships, International Journal on Digital Libraries, Volume 6, Number 2, 2006 .
3. M. Eap, M. Hatala, D. Gasevic, Technologies for enabling the sharing of Learning Objects, International Journal of Advanced Media and Communication, Volume 2, Number 1, 2008.
4. E. Duval, S. Ternier, M. Wolpers, J. Najjar, B. Vandeputte, K. Verbert, J. Klerkx, M. Meire, X. Ochoa, Open metadata for open educational resources in an open infrastructure, Proceedings of the OpenLearn 2007: pages:36-38, 2007
5. S. Ternier, and E. Duval, Interoperability of Repositories: The Simple Query Interface in ARIADNE, International Journal on E-Learning, 5 (1), pp. 161-166., 2006.
6. T. K. Shih, F. H. Lin, Y.-L. Du, L. R. Chao and W Kim., Extending CORDRA for Systematic Reuse, Lecture Notes in Computer Science, Volume 4823/2008, Springer Berlin / Heidelberg, 2008.

7 M. Hatala , G. Richards, J. Willms, The eduSource Communication Language: implementing open network for learning repositories and services, Symposium on Applied Computing archive, 2004.

8 K. Verbert, E. Duval, M. Meire, J. Jovanovic, D. Gasevic, Ontology-based learning content repurposing: the ALOCoM framework (abstract learning object content model), International Journal on E-Learning, January, 2006.

9 B. Haslhofer, B. Schandl, The OAI2LOD Server: Exposing OAI-PMH Metadata as Linked Data, Proceedings of WWW 2008 Workshop Linked Data on the Web, 2008.

10. X. Ochoa, K. Cardinaels, M. Meire, E. Duval, Frameworks for the Automatic Indexation of Learning Management Systems Content into Learning Object Repositories, Edmedia- World Conference on Educational Multimedia, Hypermedia & Telecommunications , 2005.
11. LUISA, http://luisa.atosorigin.es/www/, 10/02/2009
12. E. Newcomer, Understanding Web Services: XML, WSDL, SOAP, and UDDI, Addison-Wesley, 2002 .
13. W. Nejdl, M. Wolpers, European E-Learning: Important Research Issues and Application Scenarios, Proceedings of ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia & Telecommunications
14. H. Saliah-Hassane Kourri, A. De la Tejal, , Building a Repository for Online Laboratory Learning Scenarios, Frontiers in Education Conference, 36th Annual Volume , Issue , 27-31 Oct. 2006 Page(s):19 – 22

15. Q. Liu, Z. Yang, K. Yan, J. Jin, W. Deng, Research on DRM-enabled learning objects model, International Conference on Information Technology: Coding and Computing, 2005. ITCC.
16. H. Van de Sompel, M. Nelson, C. Lagoze, S. Warner, Resource harvesting within the OAI-PMH framework, D-Lib Magazine, 2004.
17. R. Denenberg, SRW/SRU Version 1.1, The Library of Congress, Washington, DC, 2004.
18. L. Dempsey, The subject gateway: experiences and issues based on the emergence of the Resource Discovery Network, Journal for Online Information Review, Volume: 24, Issue:1, Page:8 – 23.
19. B. Simon, D. Massart, F. Assche, S. Ternier , A simple query interface for interoperable learning repositories, Proceedings of the 1st Workshop on Interoperability of Web E-learning Systems, 2005.
20. D. Morr, Lionshare: A federated p2p app, Internet2 members meeting, Fall, 2004.
21. M. Hatala, J. Willms, G. Richards, T. Eap, Interoperability Performance of ECL Bridges: Implications for Connecting Learning Object Repositories, Joint Conference on Digital Libraries (JCDL), 2005.
22. W. Kraan, RAMLET study, Digital Repositories programme 2007-8, JISC Integrated Information Environment committee, 2008.
23. Sommaruga, L. (2004). An approach to content re-usability and adaptability in e-learning. In P. Kommers & G. Richards (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (pp. 2098 – 2104). Chesapeake, VA: AACE C. Duncan, Digital Repositories: e-Learning for Everyone, eLearn International, Edinburgh 9-12 February 2003.
24. Gasevic, D., Jovanovic, J. and Devedzic, V., G. Geser, Open Educational Practices and Resources: Te OlcOs Roadmap 2012, Universitat Oberta de Catalunya, vol. 4 n.1, Ontology-based annotation of learning object content, Interactive Learning Environments, 2007.
25. Swiss Virtual Campus project team, LOR Report 06/07: Test Results of Feasibility Study, Serving Swiss Universities - SWITCH, 2007.
26. Godwin-Jones, R., Emerging Technologies: Learning Objects Scorn or SCORM?, Language, Learning & Technology, Vol. 8, 2004.
27. ARIADNE Tools, Integrated Learning Management Systems, ARIADNE Foundation for the Knowledge Pool, 2006.

Krešimir Fertalj is an associate professor at the Department of Applied Computing at the Faculty of Electrical Engineering and Computing, University of Zagreb. Currently he lectures a couple of undergraduate and postgraduate courses in Computing. His professional and scientific interest is in computer-aided software engineering, complex information systems and in project management. He has written over a hundred scientific and professional publications and participated in conferences locally and abroad. He participated in a number of information system designs, implementations and evaluations. Fertalj is member of Croatian Academy of Engineering, ACM, IEEE and PMI.

Nataša Hoić-Božić received the B.Sc. degree in mathematics and information science from the University of Rijeka, Rijeka, Croatia, in 1990, the M.S. degree in computer and information science from the University of Ljubljana, Ljubljana, Slovenia, in 1997, and the Ph. D. degree in computing from the Faculty of Electrical Engineering and Computing (FER), University of Zagreb, Zagreb, Croatia, in 2002. She is currently an Associate Professor in the Department of Informatics, University of Rijeka. Her main research interests include adaptive hypermedia, multimedia systems, and educational technologies, focusing on blended learning approaches.

Hrvoje Jerković received B.Sc. degree in radio communications and professional electronics at the Faculty of Electrical Engineering and Computing, University of Zagreb in 2002., the M.S. on the same Faculty at the department of Applied Computing in 2007. He is ccurrently in Ph.D. program on the same Faculty. His main research fields include information systems, knowledge management, learning systems and digital repositories. He is currently a lecturer at the Department of Informatics at Zagreb School of Economics and Management.

Received: November 27, 2008; Accepted: February 01, 2010.

UDC 004.658.6, DOI:10.2298/CSIS090203002D

A Domain-Specific Language for Defining Static Structure of Database Applications

Igor Dejanović, Gordana Milosavljević, Branko Perišić, and Maja Tumbas

Faculty of Technical Sciences, Trg D. Obradovića 6,
21000 Novi Sad, Serbia
{igord,grist,perisic,majat}@uns.ac.rs

Abstract. In this paper we present DOMMLite - an extensible domain-specific language (DSL) for static structure definition of database-oriented applications. The model-driven engineering (MDE) approach, an emerging software development paradigm, has been used. The language structure is defined by the means of a metamodel supplemented by validation rules based on Check language and extensions based on Extend language, which are parts of the openArchitectureWare framework [1]. The metamodel has been defined along with the textual syntax, which enables creation, update and persistence of DOMMLite models using a common text editor. DSL execution semantics has been defined by the specification and implementation of the source code generator for a target platform with an already defined execution semantics. In order to enable model editing, a textual Eclipse editor has also been developed. DSL, defined in this way, has the capability of generating complete source code for GUI forms with CRUDS (Create-Read-Update-Delete-Search) and navigation operations [2,3,4,5].

Keywords: DSL; Domain-specific; MDE; MDSD; MDA; CRUD; Modeling; Meta-modeling; Generator.

1. Introduction

One of the issues that continues to pose difficulties for computer engineers and developers is increasing complexity of software and supporting hardware architecture. A variety of different methods has been employed in an attempt to overcome these issues, but what they all have in common is raising the level of abstraction. Although powerful, used abstraction are usually computer-, i.e. solution space-oriented, as opposed to being application domain-, i.e. problem space-oriented [6]. Developers still need to perform the mental mapping of concepts found in the solution domain to concepts found in the problem domain and to apply these mappings manually during the course of implementation [7].

The use of computer related concepts in solving real-world problems leads to all sorts of problems:

- Communication issues arise between technology and business domain experts. They usually have different interpretations of concept semantics, which leads to errors in early phases of software development.
- Mapping between the application domain and the technology domain is an error-prone manual process.
- A minor change in business domain requirements can lead to huge changes in the technology domain layer.
- Rapid pace of technology changes renders applications out-of-date and leads to constant need for migration to new platforms, or new versions of the same platform, which increases maintenance costs.

The importance of using domain-specific concepts in software development is explained in [8]:

"Perhaps the greatest difficulty associated with software development is the enormous semantic gap that exists between domain-specific concepts encountered in modern software applications, such as business process management or telephone call processing, and standard programming technologies used to implement them.

...

Clearly, the more directly we can represent concepts in the application domain, the easier it becomes to specify our systems.
Conversely, the greater the distance between the application domain and the model, the less value we get from modeling."

In this paper we present DOMMLite, a domain-specific language (DSL) for static structure definition of database-oriented applications. The purpose of this paper is to give an overview of the DOMMLite language and its supporting tools and to address some issues and choices that have been made in the design of the language. It is by no means a full specification of the language; therefore, we provide some insight into the differences between DOMMLite and other OO modeling languages and consider common concepts known from other languages. A full specification of the DOMMLite language is given in [9].

The language has been designed and implemented using model-driven engineering (MDE) techniques, which are a specialization of DSL engineering techniques [10]. DOMMLite builds on the concepts of other object-oriented modeling languages such as UML [11], MOF [12], ECore [13], and concepts expressed in Domain-Driven Design [14]. It is a declarative language and, although many of its constructs more or less resemble those found in other OO modeling languages, there are differences that will be identified in the rest of the paper; therefore, DOMMLite is not created by mere extension or restriction of existing languages/meta-models. Having that in mind we state that, according to the classification introduced by M. Fowler (see Sect. 2), DOMMLite is an external language.

The purpose of the language is to enable developers to specify, in a simple manner, static structure of database oriented applications with enough meta-data to generate fully working applications with implementation of CRUDS (Create-Read-Update-Delete-Search) operations without resorting to the more heavy-weight modeling languages such as UML. Both the abstract syntax, defined by a metamodel, and one of the possible concrete syntaxes have been developed. The implementation of the abstract and the concrete textual syntax of the language, the model editor and the source code generator has been carried out using the openArchitectureWare (oAW) generator framework [1], a metamodel agnostic framework which is well integrated with the Eclipse Modeling Framework [13]. For the purpose of this work a textual concrete syntax was created. Although creating a graphical syntax and a graphical editor is made a lot easier with projects such as GMF[1] and GEMS[2], the creation of a fully featured graphical editor in the context of changing requirements still requires a considerable amount of time. This is why work on further improvements of the textual syntax will continue until the language is stable enough. DOMMLite defines the execution semantics by the implementation of the source code generator for a target platform with an already defined execution semantics.

The rest of the paper is structured as follows. Section 2 gives overview of DSLs as the underpinning technique used in this paper. In section 3 the abstract syntax of the DOMMLite language is described. Section 4 gives an overview of the language's concrete syntax based on the xText [15] language. Section 5 explains the design and implementation process of the application code generator. Section 6 points out several issues regarding the completion of the eclipse editor generated by the oAW framework. Section 7 analyzes related work. Section 8 gives final conclusions.

2. Domain-Specific Languages

Domain-specific languages, in contrast to general-purpose languages (GPL), offer, through specific notations and abstractions, the power of expression focused on, and usually restricted to, a particular problem domain [16].
Some of the advantages of using DSLs over GPLs are:

- DSLs are usually more concise and expressive than GPLs, which enables programmers to represent their intentions more clearly.
- DSL syntax, both textual and graphical, can be tailored to the specific knowledge of the domain experts.

1 http://www.eclipse.org/gmf/
2 http://www.eclipse.org/gmt/gems/

- Concepts used in DSLs are found in the application domain, so their use does not require domain experts to have programming skills.
- Expressing the domain construct through concepts independent of the technology used results in a longer lifespan of the application. If applied correctly, application description needs to change only if business requirements change and is immune to changes in the technology layer, which can be handled by the application generator (DSL compiler).
- Using higher- level abstraction leads to the reduced number of lines of code (LOC) (in terms of textual syntaxes), which has a positive impact on the development and maintenance. Some researchers achieved a 50:1 ratio of LOC in favor of DSLs [17]. Software fault density (number of software faults per one thousand lines of code) does not significantly depend on the language being used [18]. Therefore, using DSL languages reduces the number of software bugs, which leads to increased software quality and lower maintenance costs.
- In situations where code analysis, verification, optimization, parallelization, and transformation techniques are considered to be very difficult or almost impossible to achieve with GPLs, it is possible to achieve them in the context of DSLs [19].

Better expressiveness of DSL languages does not come for free. For the sake of it they give away their generality [19], so DSLs are usually not very useful outside of the domain they were constructed for.

Some programming languages started out as DSLs, but have evolved towards GPLs by getting more features. A reverse process has not been observed in the history of programming languages [10].

There are also widespread languages that are essentially DSLs, although they may not be known as such. Examples of such languages include HTML - the language for describing hypertext documents, which forms a foundation of today’s global network, SQL - a structured language for querying, updating and deleting data in relational databases, LaTeX - a language for document typesetting. Even Spring's[3] XML-based configuration file language can be considered a DSL for expressing application configuration.

Different classifications of DSLs exist. DSLs can be classified in the same way as GPLs. For example, they can be classified as: object-oriented or functional, imperative or declarative, visual or textual.

Another classification, by the way DSLs are constructed, is given by Martin Fowler in [20].

He divides them into two groups:

[3] http://www.springframework.org/

- External DSLs - These languages are also called little languages and are very popular in the Unix community. Representatives include awk, sed, flex, yacc etc. Their main property is that they are built from scratch, with their syntax carefully tailored for the domain in question.
- Internal DSLs - In contrast to external DSLs, internal DSLs are built on top of an existing GPL, extending their syntax to add support for domain-specific constructs. They are gaining popularity with the emergence of GPLs which have mechanisms for easy extensibility. Some of the representatives are Ruby[4] [21], Scala[5], or Python[6] [22]. There are also internal DSLs based on main-stream programming languages like Java [23]. Using this approach one can get an entire tool-chain of the host language for free (editors, compilers, debuggers etc.). Internal DSLs are also called domain-specific embedded languages [24], or simply embedded languages [16].

Fowler's classification can also be applied in the context of modeling languages. An external DSL in the context of modeling technologies and visual syntaxes is usually referred to as domain-specific modeling language (DSML) [7]. Similarly, a UML profile-based modeling language is, according to Fowler's classification, an internal DSL.

A completely clear distinction between DSLs and GPLs does not exist, although attempts have been made to construct a method for quantifying domain specificity [25]. As stated in [19], domain specificity is a matter of degree: it largely depends upon the notion of a domain.

For the purpose of this paper we will use the definition of the DSL based on MDE ideas, given in [10], as follows.

A DSL is a set of coordinated models:

- Domain definition metamodel(DDMM) - is a conceptualization of the domain that introduces the basic abstractions of the domain and their mutual relations. Once such an abstract entity is explicitly represented as a model, it becomes the reference model for the models expressed in the DSL, that is, it is a metamodel. That metamodel is referred to as the domain definition metamodel (DDMM).

- Concrete syntax - represents a transformation of DDMM to the "display surface" metamodel. More than one such transformation can be defined, or to put it simply, for each DDMM multiple concrete syntaxes can be defined, both textual and visual.

- Semantics - A DSL can have execution semantics defined. Semantics is also defined by a transformation from DDMM into a

4 http://www.ruby-lang.org/
5 http://www.scala-lang.org/
6 http://www.python.org

DSL or GPL which has an already defined precise execution semantics.

Using this definition as the basis, the following section defines DDMM, the concrete syntax and semantics of DOMMLite.

3. Abstract Syntax

The abstract syntax of the DOMMLite language is defined by a metamodel. A great deal of DOMMLite is inspired by other OO based metamodels, such as MOF, UML and ECore, and in many ways it builds on top of the concepts from these reputable languages. The concrete implementation of DOMMLite is based on the Eclipse Modeling Framework (EMF[7]), an OO (meta)modeling infrastructure. The abstract syntax of the language will be presented using UML class diagram notation. The most important conceptual primitives of the DOMMLite language are based on well-known concepts described in [14].

The semantics of concepts in DOMMLite is given in the form of recommendations. The model compiler defines the semantics in detail, so the language's full semantics interpretation is left to the compiler developer. Recommendations for semantics interpretation described in the following sections have been followed in the implementation of the DOMMLite model compiler prototype (see Sect. 5). This section gives an overview of some of the most important conceptual primitives of the DOMMLite language.

3.1. Data Types

The *DataType* metaclass (see Fig. 1) defines, in DOMMLite terminology, simple types of the DOMMLite language. Simple types have no internal structure, which distinguishes them from complex types (e.g. entities, services etc) that do. *UserDataType* and *BuiltInDataType* are also referred to as primitive types. Primitive types can be built-in (defined by the language itself) or user-defined. Built-in types are: *void, bool, int, real, money, string, char, date, datetime*. User-defined types can be defined by the modeler and used troughout the model in the same way as the built-in ones. The semantics of the types is implemented on the target platform. The semantics of built-in types is implemented once for the given platform and can be reused in many projects without change. Following definitions in [26], it is a part of "the platform". The user-defined type semantics is specified on the target platform for the project where it is introduced in the model. If the source code generator is carefully tailored, it is usually not necessary to make

[7] http://www.eclipse.org/modeling/emf/

changes to the generator itself in order to support the newly defined type (see Sect. 5 for an example of the generator prototype).

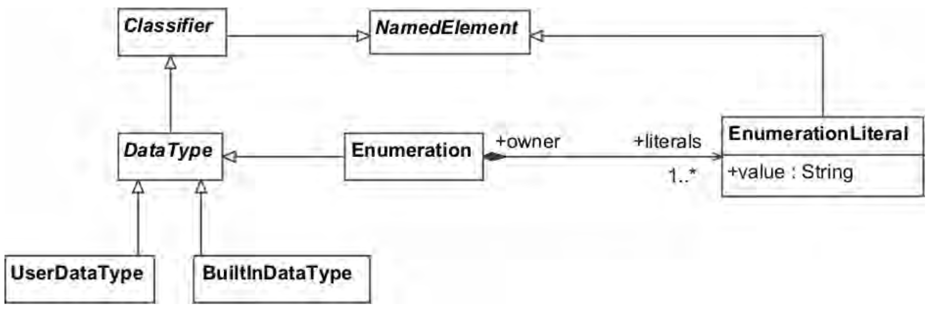


Fig. 1. The DataType metaclass

3.2. Model Organization

The DOMMLite language is organized in a structural manner using the notion of packages (see Fig. 2). The *Package* metaclass in DOMMLite has the semantics similar to that of the *Package* metaclass in UML, but it is implemented differently. The language elements that are organized in this way must inherit from the *PackageElement* metaclass. In order to support package nesting, the *Package* metaclass also inherits from the *PackageElement*. The model itself is described by the *DOMMLiteModel* metaclass. Instances of this metaclass contain zero or more *Package* metaclass instances but not *Classifier* instances (see Sect. 3.3); therefore DOMMLite forbids creation of classifiers outside of a package. In order to define a classifer one must enclose it inside a package.

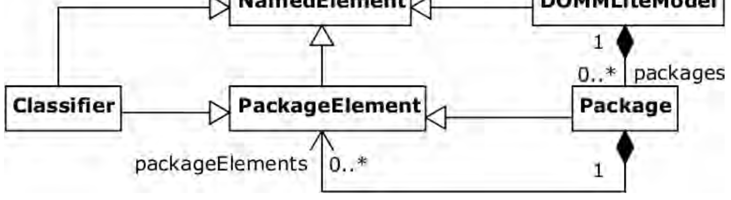


Fig. 2. Model organization

3.3. Classifier

The *Classifier* (see Fig. 3) abstract metaclass represents the superclass of metaclasses that describe key concepts of the language. Excluding *DataType*, these key concepts are, in DOMMLite terminology, also called complex types. There is a difference in the concept of a *Classifier* between the UML language and the DOMMLite language. For the sake of simplicity, the DOMMLite language unifies notions of type and classifier in the metaclass *Classifier* while those notions are separate in UML. There is no *Type* metaclass in DOMMLite – *Classifier* plays the role of the *Type* metaclass from the UML language. The classifier inherits the *PackageElement* and, consequently, can be nested inside packages. It is also a *NamedElement*, so all its descendants have a name, a short and a long description. The name should consist of letters, digits and underscores, and should begin with a letter, though these rules are not enforced by constraints in the current version of DOMMLite. The classifier name is usually mapped to the programming language identifier during model compilation; to make this mapping easier, these simple rules should be followed. The short description (*shortDesc*) is a description of an entity in a few words and is usually used as a classifier label in the GUI, or to aid model navigation and search. The long description (*longDesc*) can be arbitrarily long and is used to clarify the classifier's purpose. It can be used for tool tip generation, or for context-sensitive help support. The *NamedElement* metaclass, although it has different properties, represents the same concept from UML language.

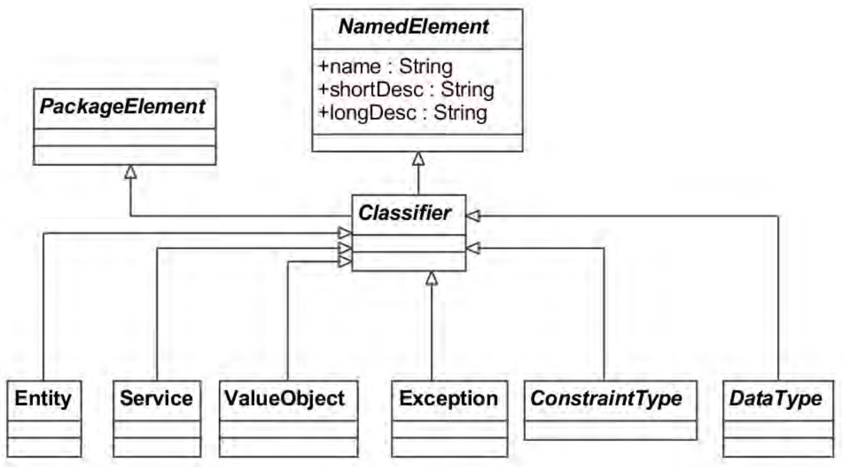


Fig. 3. The Classifier metaclass

3.4. Feature

Feature (see Fig. 4) is an abstract metaclass which represents properties (structural features) and operations (behavioral features) of main DOMMLite concepts with internal structure. *Feature* inherits the *TypedElement* metaclass, so all features have the following properties:

- ***required*** – If this flag is set, this feature's value must be defined. This serves as a support for NULL constraints in relational databases. For operations, this can be used as an indication that its return value is always non-null.
- ***many*** – If this flag is set, this feature is a collection (e.g. an array).
- ***ordered*** – This flag is taken into account only if the *many* flag is set to true. If it is set, this feature's values are ordered, meaning that elements of the collection have a notion of an index inside the collection.
- ***unique*** – If this flag is set, all elements of this feature must have a unique value. This flag is taken into account only if the many flag is set to true.
- ***multiplicity*** – The value of this property determines the multiplicity of the elements of multi-valued features. If it is set to zero, the multiplicity is not restricted. This flag is taken into account only if the *many* flag is set to *true*.
- ***type*** – The value of this property determines the type of this feature. The type of a feature can be any classifier, therefore the type can be simple or complex.

Features, being typed elements, can have a set of constraints (see Sect. 3.9), through which validation rules and tags can be attached to them for their further specification. Features are specialized into properties (the *Property* metaclass) and operations (the *Operation* metaclass).

3.4.1. Property

The *Property* (see Fig. 4) metaclass describes the structural features of the modeling elements. *Property*, being a typed element, can refer to other model elements which conform to the *Classifier* metaclass. Properties in DOMMLite can, on the semantic level, be compared to *EStructuralFeature* in ECore or *Property* in EMOF, but it is more similar to EMOF, because ECore divides this semantic concept into two metaclasses: *EAttribute* and *EReference*.

Property is a *TypedElement*; therefore it has type. If the type of a property is simple (primitive type or enumeration), we call it an attribute. If the type of a property is complex (e.g. *Entity*, *ValueObject* etc.), we call it a reference.

References in DOMMLite can be bidirectional. This ability is represented by the *oppositeEnd* reference of a *Property* metaclass, which "points to" the model element conforming to the *Property* metaclass. The *oppositeEnd* reference must point to the model object conforming to the *Property* metaclass which is contained inside the model object referenced by the type of a reference. This is enforced by a constraint.

References can be used to represent a containment relationship (containment property) between model elements. The semantics of the containment relationship is similar to that of a composite association in UML or a containment relationship in ECore. Containment references affect the object's life-cycle. An object can not exist without its containing object.

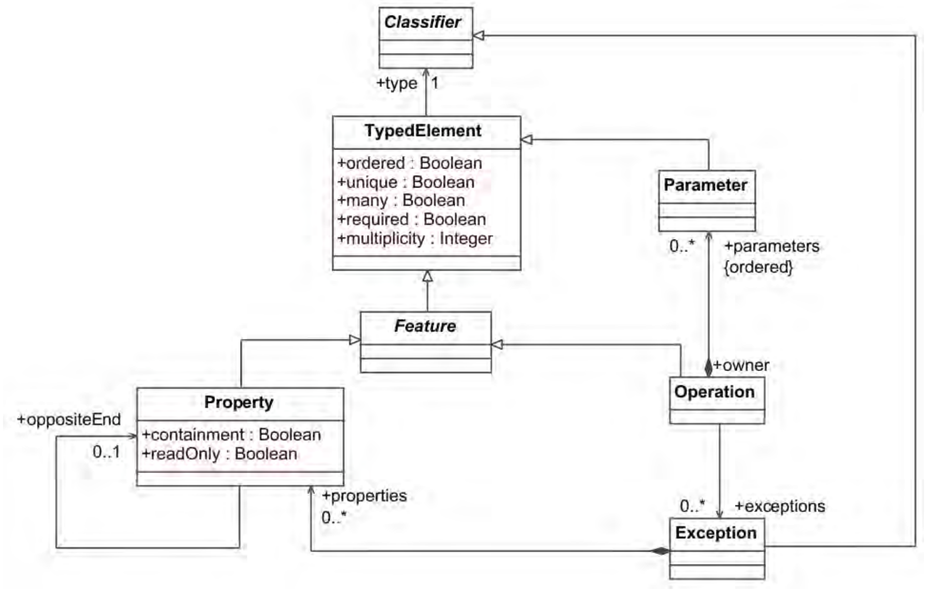


Fig. 4. TypedElement and Feature metaclass

3.4.2. Operation

The *Operation* (see Fig. 4) metaclass describes behavioral features of modeling elements. An operation can have parameters and a return type and can throw exceptions. The current version of DOMMLite deals primarily with structural properties of the observed systems, as a result of which modeling the behaviour of operations is currently out of scope of this language. As a means of supporting the specification of services (see Sect. 3.7) as well as the introduction of of behavioural model elements in latter versions of DOMMLite (see Sect. 8) we chose to introduce operations through operation signatures (name, return type, parameters, exceptions). The behavioural logic of operations is currently specified on the target platform programming language. The interpretation of operations and their semantics is left to the model compiler developer.

3.5. Entity

Entities (see Fig. 5) represent objects whose identity does not change throughout their lifetime. They have the ability to persist their state between application sessions. The identity of an entity is unique within the boundaries of a software system and is represented by one or more of its properties. Two entities are considered equal if their identities are equal.

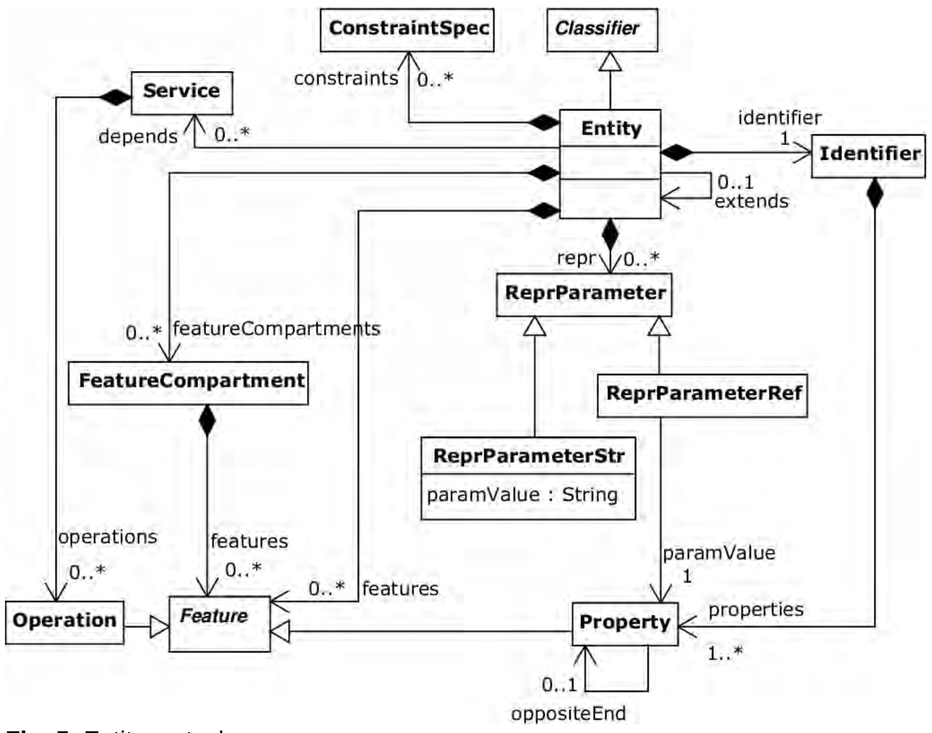


Fig. 5. Entity metaclass

3.5.1. Semantic Identifier

Identifier is a metaclass that represents the semantic identifier of an entity. Exactly one semantic identifier is defined for each entity. A semantic identifier consists of one or more properties which, together, uniquely identify an entity instance in the given software system.

In most cases, entities are persisted in relational databases, since they are still the most widespread storage mechanism. As a result, semantic identifier is closely related to the concept of a primary key. This concept deserves closer consideration.

Semantic vs. Synthetic identifier – There are two approaches in choosing properties which will represent the primary key of an entity in a relational database:

- ***Semantic identifier (natural key)*** – One or more existing properties of a business entity are chosen to represent the entity's primary key. We say that the identifier is semantic because it has meaning in the application domain. Choosing the right properties for an identifier is not an easy task, since the invariability of primary key value must be ensured throughout the lifetime of the entity instance.
- ***Synthetic identifier (surrogate key)*** – A synthetic identifier comprises properties without any business domain meaning. Its values are usually generated by the application or the database and its type is chosen based on the capabilities and performance of the database in dealing with such types. A synthetic identifier is a technical concept and does not need to be a part of the model.

Based on the experience in working with relational databases described in [27], the authors have come to the conclusion that using synthetic identifiers in relational databases is a better approach on the long run. On the other hand, using synthetic identifiers makes the implementation of the zoom technique [2] for manual data entry of references between entities (in some systems it is also known as the lookup technique) less efficient. The application user would have to specify the identifier of a referenced entity, which does not have any domain meaning. It would be irrational to expect that users would be able to remember these identifiers; this is why searching for referenced entities by the value of its other properties is the only viable option. The downside of this approach is that, in order to make a reference to another entity, the user would have to activate the search form for every reference that he or she makes. It is very common for the user to already know the semantic identifier of the referenced entity (e.g. social security number or bank account number); hence the need to call the search form each time a reference is entered, instead of directly specifying the semantic identifier, would severely slow data entry down.

Having that in mind, we propose a hybrid approach as a solution. On the modeling level a semantic identifier is always used. This will allow the use of approaches for automatic generation of the efficient zoom mechanism, which is used for reference entry and GUI forms navigation. In order to overcome shortcomings which the use of semantic keys in relational databases introduces, synthetic keys are generated for this purpose instead, while the application layer is in charge of mapping semantic identifiers to the synthetic database primary keys based on the information available in the DOMMLite model. Of course, DOMMLite does not impose use of synthetic identifier on the database level; it is only a recommendation. It is up to the developer of the source code generator to make a decision if the synthetic or the semantic identifier will be used in the database.

Global and local entity identifier – We classify unique entity identifiers into two categories: global and local. A local entity identifier is unique in the context of its containing entity; it does not need to be unique in the context of the entire modeled system. In DOMMLite local entity identifiers are modeled using *Identifier* metaclass. A global semantic identifier is unique in the context of the entire software system. It consists of the local semantic identifier of an entity combined with the global semantic identifier of the containing entity. If an entity does not have the containing entity, its local semantic identifier is equal to its global semantic identifier.

3.5.2. Feature Compartments

The basic idea of feature compartments is to logically group different features, both behavioral and structural, for easier model navigation inside coarse-grained entities, as well as to support automatic generation of GUI forms. In desktop applications compartments are usually used to create pages on the Tab visual component, which contains of visual components representing compartment properties. This enables us to generate more intuitive user interfaces without additional manual customization. For an example of feature compartment's usage see compartment *Contact Information* whose definition is shown in figure 10 and the generated web form is shown in figure 19.

3.5.3. Inheritance

Entities support single inheritance model. In the current version of DOMMLite inheritance is defined only at the level of the abstract and concrete syntax. Its semantics is left undefined and the model compiler developer is free to define its meaning. As a recommendation DOMMLite inheritance should follow the semantics of inheritance in other OO languages.

3.5.4. Service Dependency

Service dependency can explicitly be stated in the model. This information is used to support the *Dependency Injection* design pattern [28], thus making service reference available for entity operations by the time they get called without the need for the entity to obtain that reference by itself (e.g. for Spring framework this can be done by generating XML configuration files where reference injection is stated declaratively).

3.5.5. Entity Operations

An entity operation is described by the *Operation* metaclass (see Sect. 3.4.2). A business method that performs operations solely on one instance of an entity should be specified as an operation of that entity. It is recommended to model operations that operate on multiple instances of an entity as service operations.

3.5.6. Textual Representation

It is useful, if not necessary for entities to have a mechanism that will allow them to be represented in a textual form. For this purpose the collection *repr* of instances of *ReprParameter* metaclass is defined. The textual representation of an entity is obtained by concatenating strings (instances of *ReprParameterStr*) and textual representation of properties (instances of *ReprParameterRef*). This information is usually used for human-readable entity representation in the GUI. See figure 10 for an example (*firstName* and *lastName* are used for the textual represenation of the Student entity).

3.6. ValueObject

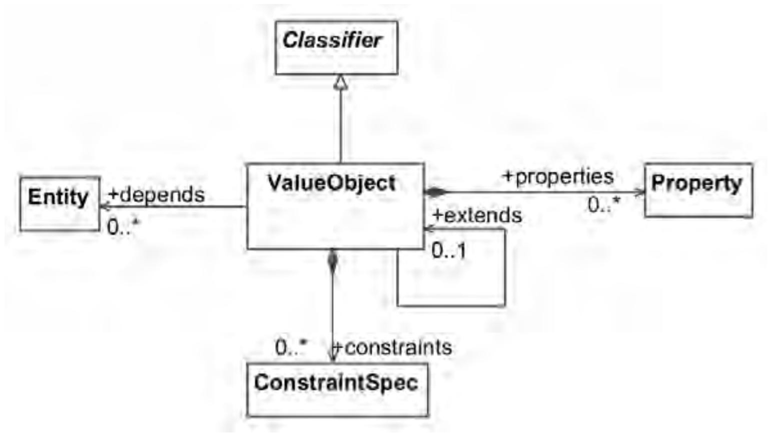


Fig. 6. ValueObject metaclass

Value (transfer) Objects (VO for short) are transient objects without identity. They are not meant to be persisted and are usually used to encapsulate data for interchange between different tiers of multi-tier applications. VOs can depend on entities (Fig. 6) if properties of a VO are based on properties of an entity. As with most metaclasses, the semantics of

VO can further be refined using a collection of constraints (see Sect. 3.9). Although VOs can have operations [14], in this version of DOMMLite language they are simple objects with structural features only. Inheritance of VOs is defined at the syntax level. Its precise semantics is left to the model compiler developer to define, but it is recommended to follow the semantics of inheritance defined in other OO languages.

3.7. Service

The role of the *Service* metaclass (see Fig. 7) is to describe objects whose purpose is to provide services to other domain objects. Services consist of logically interrelated operations that achieve a particular objective. These operations can query entities and can act on them by changing their state. In the current DOMMLite version services do not have properties, so their internal state is not modelled (they are stateless).

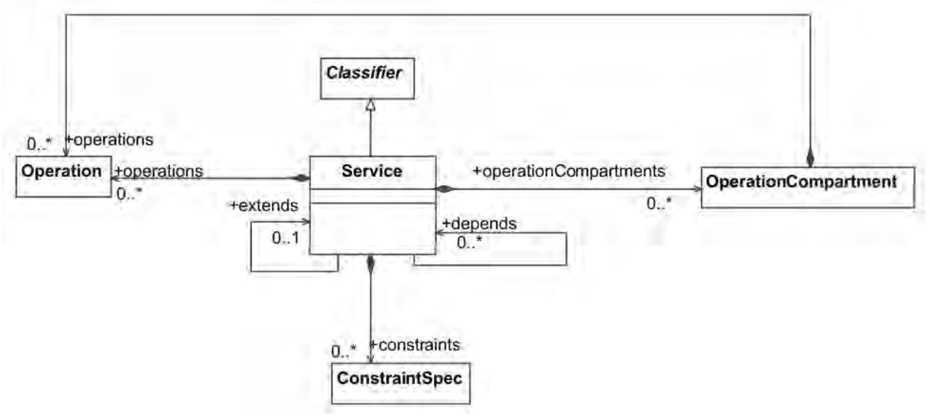


Fig. 7. Service metaclass

If necessary, service state can be modelled as an entity but the DOMMLite language currently has no support for explicitly stating which entity is used for state preservation of some particular service.

Service operations usually operate on multiple instances of entities or value objects. Operations that do not operate on a single entity instance, or do not operate on an entity instance at all should be modeled as service operations. Operations that query a single entity instance or change its state are usually better represented by entity operations (see Sect. 3.5). Service, being a *NamedElement*, has a name, a short and a long description. To facilitate model navigation and GUI generation, service operations can logically be grouped into operation compartments represented by the *OperationCompartment* metaclass.

Services can depend on each other. This information is used to implement the *Dependency Injection* design pattern. The single inheritance model is supported at the abstract and concrete syntax level. The semantics of inheritance is left to the model compiler developer to define, but it is recommended to follow the semantics of inheritance from other OO languages. In the current implementation of the generator, service inheritance is not used (see Sect. 5). The semantics of services and service operations can further be refined using a collection of constraint specifications (see Sect. 3.9).

3.8. Validators and Tags

DOMMLite models can be augmented by validators and tags that can be attached to model elements (see Fig. 8). Validators define rules that should be enforced upon a running system in order to maintain a consistent state. For an example of validator usage see figure 10. Figure 19 shows a generated web form with applied validators.

Tags are simple constraints, similar to UML tags and stereotypes, which are used to alter or further refine the semantics of modeling elements. For example, the built-in tag *plural* is used to define the plural name of a modeling element, while the built-in tag *searchBy* is used to define properties of an entity that will be searched during keyword-based searches. The next section describes validators and tags in the context of extensibility.

3.9. Extensibility

The technique used to achieve extensibility (see Fig. 8) in DOMMLite is similar to that used in UML profiles. A modeler can introduce new data types, validator types and tag types into a DOMMLite model. These elements can be used in the rest of the model in the same way as the built-in ones.

Main metaclasses, which support extensibility, are *UserDataType* (see Sect. 3.1), *ConstraintType* and *ConstraintSpec* metaclasses. *ConstraintType* represents the definition of the type of a constraint, while *ConstraintSpec* is a concrete usage or instance of that type. *ConstraintType* defines language metaclasses to which a constraint can be applied (the *appliesTo** property). This information is used to constrain instances of *ConstraintType* (the modeling object that conforms to *ConstraintSpec*) so they can only be applied to the modeling element specified by the *appliesTo** property. The parameters collection of the *ConstraintTypeParameter* type defines the list of formal parameters of a constraint. Their types can be string, int (integer), ref (property reference) and ellipsis (variable number of parameters). *appliesTo* constraint as well as parameter types are checked during model editing and source code generation by a Check language rules. *ConstraintType* is specialized by *ValidatorType* and *TagType* metaclasses. *ValidatorType* metaclass represents the type of validators and the *TagType* metaclass represents the type of a tag, which can be instantiated in the model using *ConstraintSpec* metaclass.

There are two types of validators: *BuiltInValidatorType*, supplied with the DOMMLite language, and *UserValidatorType*, defined by the modeler at the model level. Figure 10 shows examples of built-in validators (*isOnlyDigits* and *isOnlyLetters*) as well as an example of a user-defined validator (*mod*). The *mod* validator accepts one parameter of the integer type. Validators are implemented on the target platform and the run-time form validation based on modelled validators is shown in figure 19. As we can see in figure 19, the source code generator (see Sect. 5) generates validators that, by default, do not pass validation (*isOnlyLetters* validator is undefined and therefore will not validate). After the implementation of validators on the target platform (see Fig. 18), it will be ensured that only valid data is stored in the database.

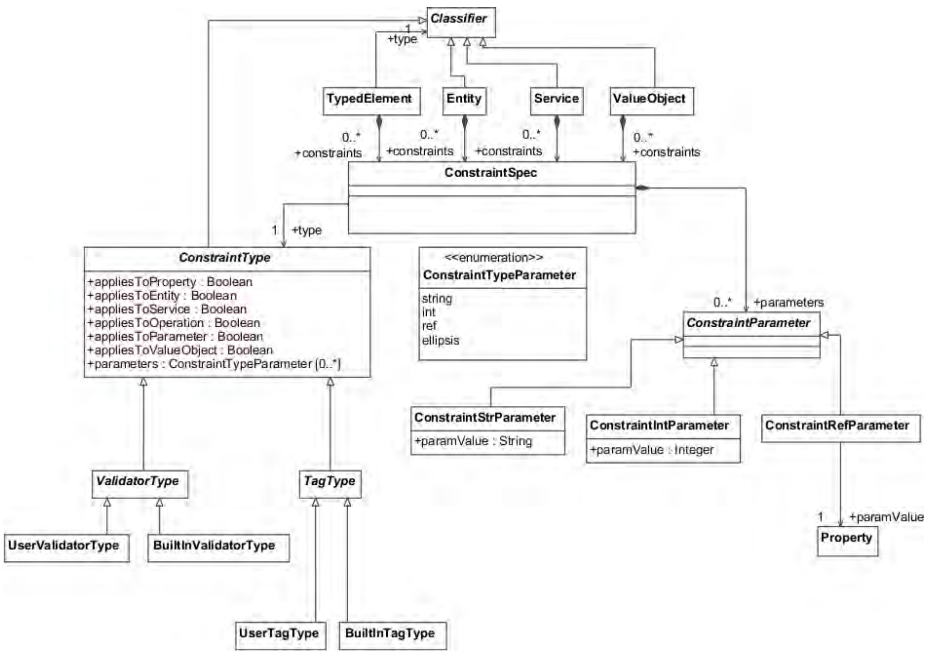


Fig. 8. Extensibility, validation and constraints

Tags can also be user-defined (*UserTagType*) or built-in (*BuiltInTagType*). Tags can be used to help identify a particular model element. For example, applying the finder tag to an entity operation could mark it as a so-called finder operation. Finder operation searches for and returns the entity or collection of entities that match given search criteria. Using this simple approach source code for this type of operation can automatically be generated.

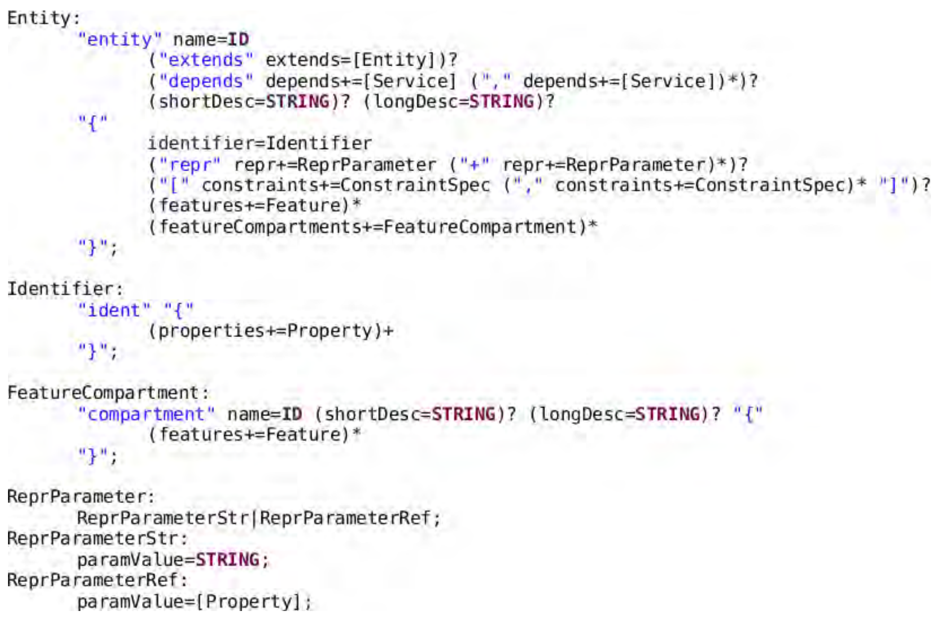
The instance of *ConstraintType* is defined by the *ConstraintSpec* metaclass. Objects that conform to *ConstraintSpec* metaclass can be assigned to all typed elements, entities, services and value objects, as long as they respect constraints enforced by *appliesTo** property of their

ConstraintType metaclass. Constraint parameters must conform to the type and ordinal position of the constraint type specification. If the formal parameter of a constraint type specification is ellipsis, all its instances can use any number of parameters of any valid parameter type.

4. Concrete Syntax

Based on the abstract syntax different concrete syntaxes are possible. In this section we describe a textual concrete syntax implemented using xText language and tool, which is part of the openArchitectureWare generator framework [15]. xText is an EBNF-like language that can be used to specify textual concrete syntaxes as well as abstract language syntax (the metamodel) using the same definition. This feature of xText is used in the implementation of DOMMLite so that abstract syntax described in section 3 is defined along with the definition of DOMMLite concrete textual syntax. The rest of this section presents specifications of concrete syntaxes of two main language concepts: entity and service, as well as the syntax of extensibility support.

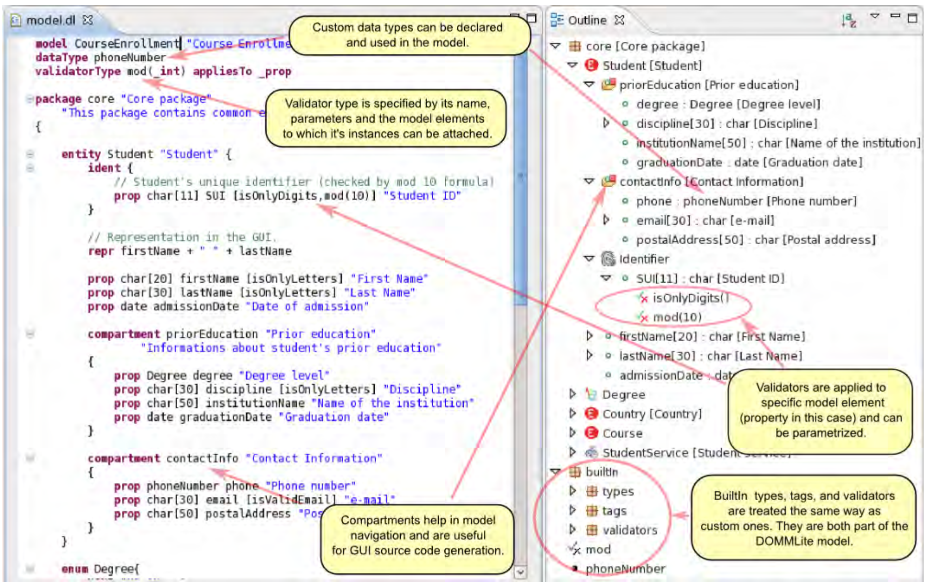
4.1. Entity Syntax



```
Entity:
      "entity" name=ID
            ("extends" extends=[Entity])?
            ("depends" depends+=[Service] ("," depends+=[Service])*)?
            (shortDesc=STRING)? (longDesc=STRING)?
      "{"
            identifier=Identifier
            ("repr" repr+=ReprParameter ("+" repr+=ReprParameter)*)?
            ("[" constraints+=ConstraintSpec ("," constraints+=ConstraintSpec)* "]")?
            (features+=Feature)*
            (featureCompartments+=FeatureCompartment)*
      "}";

Identifier:
      "ident" "{"
            (properties+=Property)+
      "}";

FeatureCompartment:
      "compartment" name=ID (shortDesc=STRING)? (longDesc=STRING)? "{"
            (features+=Feature)*
      "}";

ReprParameter:
      ReprParameterStr|ReprParameterRef;
ReprParameterStr:
      paramValue=STRING;
ReprParameterRef:
      paramValue=[Property];
```

Fig. 9. Entity syntax rule in xText language

Figure 9 shows the xText rule, which defines an entity. Definition of an entity begins with the keyword entity followed by the name of the entity. Entity inheritance relationship can be defined by the keyword *extends* followed by the name of the ancestor entity. Service dependency can be specified by the keyword *depends* followed by the list of comma-separated names of services this entity depends upon. An *Entity*, being a *NamedElement*, can have a short and a long description, which are defined as strings. Curly braces demarcate the entity body. An entity must define its semantic identifier. Semantic identifier definition starts with the keyword *ident* followed by a block demarcated by curly braces, and consists of a list of one or more properties. The definition of the string representation of an entity begins with the keyword *repr* followed by a list of strings or property names delimited by a plus sign. Constraints are specified inside square braces. Features are defined on the entity level or they can be grouped in feature compartments. Feature compartments begin with the keyword *compartment* followed by the compartment name and an optional short and long name.



```
model CourseEnrollment "Course Enrollme
dataType phoneNumber
validatorType mod(_int) appliesTo _prop

package core "Core package"
    "This package contains common e
{

    entity Student "Student" {
        ident {
            // Student's unique identifier (checked by mod 10 formula)
            prop char[11] SUI [isOnlyDigits,mod(10)] "Student ID"
        }

        // Representation in the GUI.
        repr firstName + " " + lastName

        prop char[20] firstName [isOnlyLetters] "First Name"
        prop char[30] lastName [isOnlyLetters] "Last Name"
        prop date admissionDate "Date of admission"

        compartment priorEducation "Prior education"
                "Informations about student's prior education"
        {
            prop Degree degree "Degree level"
            prop char[30] discipline [isOnlyLetters] "Discipline"
            prop char[50] institutionName "Name of the institution"
            prop date graduationDate "Graduation date"
        }

        compartment contactInfo "Contact Information"
        {
            prop phoneNumber phone "Phone number"
            prop char[30] email [isValidEmail] "e-mail"
            prop char[50] postalAddress "Pos
        }
    }

    enum Degree{
```

```
core [Core package]
  Student [Student]
    priorEducation [Prior education]
      degree : Degree [Degree level]
      discipline[30] : char [Discipline]
      institutionName[50] : char [Name of the institution]
      graduationDate : date [Graduation date]
    contactInfo [Contact Information]
      phone : phoneNumber [Phone number]
      email[30] : char [e-mail]
      postalAddress[50] : char [Postal address]
    Identifier
      SUI[11] : char [Student ID]
        isOnlyDigits()
        mod(10)
    firstName[20] : char [First Name]
    lastName[30] : char [Last Name]
    admissionDate : dat
  Degree
  Country [Country]
  Course
  StudentService [Studen
builtIn
  types
  tags
  validators
  mod
  phoneNumber
```

Custom data types can be declared and used in the model.

Validator type is specified by its name, parameters and the model elements to which it's instances can be attached.

Compartments help in model navigation and are useful for GUI source code generation.

Validators are applied to specific model element (property in this case) and can be parametrized.

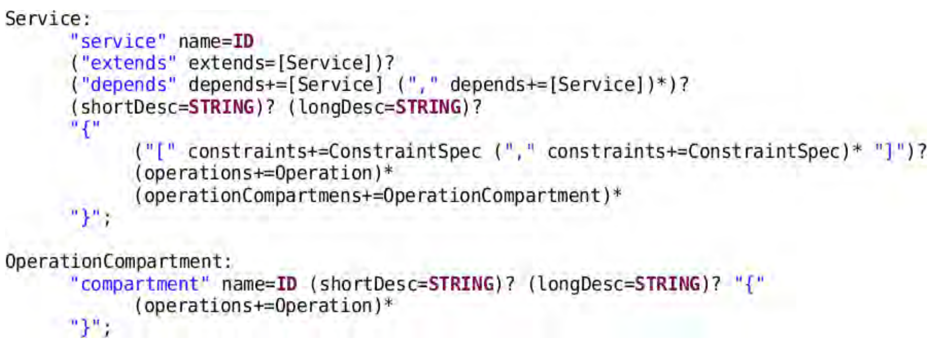
BuiltIn types, tags, and validators are treated the same way as custom ones. They are both part of the DOMMLite model.

Fig. 10. Entity in the Eclipse-based DOMMLite model editor

Figure 10 represents an example of an entity *Student* in an Eclipse-based editor. The semantic identifier of the *Student* entity consists of the property *SUI (Student Unique Identifier*), which is numeric (validator *isOnlyDigits*), and its maximum length is set to 11. SUI is checked by a mod 10 formula using custom-defined validator *mod*. On the user interface this property will have the label *Student ID*. String rendering of the *Student* entity (*repr* keyword) will be done using first name and last name separated by a space. First and last names are constrained to contain letters only by a built-in validator *isOnlyLetters*. There are two compartments: *priorEducation* and *contactInfo*. The *priorEducation* compartment has a long description defined in it, which will be used to further explain the content of the compartment. The *phoneNumber* data type, used in the *contactInfo* compartment and shown in the outline view, is user-defined. It is defined at the model level using the *dataType* keyword (see Sect. 3.9).

4.2. Service syntax

Service's xText sintax rule is shown in Figure 11. Service definition begins with the keyword *service* followed by the service name. Service inheritance is defined by the keyword *inherits* followed by the name of the ascendant service. Dependency is defined by the keyword *depends* followed by a comma-separated list of service names. Service body contains the specification of constraints, service operations and operation compartments.



```
Service:
      "service" name=ID
      ("extends" extends=[Service])?
      ("depends" depends+=[Service] ("," depends+=[Service])*)?
      (shortDesc=STRING)? (longDesc=STRING)?
      "{"
            ("[" constraints+=ConstraintSpec ("," constraints+=ConstraintSpec)* "]")?
            (operations+=Operation)*
            (operationCompartmens+=OperationCompartment)*
      "}";

OperationCompartment:
      "compartment" name=ID (shortDesc=STRING)? (longDesc=STRING)? "{"
            (operations+=Operation)*
      "}";
```

Fig. 11. Service syntax rule in xText language

4.3. Extensibility Features Syntax

Figure 12 shows the syntax rules for extensibility support. User defined data types are defined by the keyword *dataType* followed by the name of the new type.

TagType and *ValidatorType* have similar syntaxes. Their definition begins with the keyword *tagType* (or *validatorType*) and the name of the tag (validator). The name is followed by the definition of parameters as a comma-separated list of type parameters. After the keyword *appliesTo* metaclasses should be defined to which this tag (validator) can be applied. This information is used by the model editor as well as the model compiler to perform model validation.

The same type of constraint can be applied, if defined in that way, to different types of modeling constructs. Formal parameters of the constraint type definition represent the types of real parameters in constraint usage specification. The type of a formal parameter can be: _string - string surrounded by quotation marks, _int - integer, _ref - reference to a property represented by its name, ... - ellipsis means that parameter types and their number is undefined.

```
UserDataType:
      "dataType" name=ID (shortDesc=STRING)? (longDesc=STRING)?;

UserTagType:
      "tagType" name=ID
            ("("(parameters+=ConstraintTypeParameter)?
            (","parameters+=ConstraintTypeParameter)* ")")?
            ("appliesTo"
                  ((appliesToEntity?="_entity") |
                  (appliesToProperty?="_prop") |
                  (appliesToParameter?="_param") |
                  (appliesToOperation?="_op") |
                  (appliesToService?="_service") |
                  (appliesToValueObject?="_valueObject"))*)?
            (shortDesc=STRING)? (longDesc=STRING)?;

UserValidatorType:
      "validatorType" name=ID
            ("("(parameters+=ConstraintTypeParameter)?
            (","parameters+=ConstraintTypeParameter)* ")")?
            ("appliesTo"
                  ((appliesToEntity?="_entity") |
                  (appliesToProperty?="_prop") |
                  (appliesToParameter?="_param") |
                  (appliesToOperation?="_op") |
                  (appliesToService?="_service") |
                  (appliesToValueObject?="_valueObject"))*)?
            (shortDesc=STRING)? (longDesc=STRING)?;

Enum ConstraintTypeParameter:
      string="_string"|
      int="_int" |
      ref="_ref" |
      ellipsis="..."
;
```

Fig. 12. Data types and constraint types syntax rules

Figure 13 show the syntax rule for constraint usage. A constraint is defined by its name followed by real parameters surrounded by a pair of braces. In order to be used, a constraint must have its type defined in the model as a built-in or user-defined constraint type or there will be errors during model validation. The type and the number of real parameters must conform to the constraint type formal parameter specification.

```
ConstraintSpec:
       type=[ConstraintType] ("(" (parameters+=ConstraintParameter)?  (","
parameters+=ConstraintParameter)* ")")?;

ConstraintParameter:
      ConstraintIntParameter|ConstraintRefParameter|ConstraintStrParameter;

ConstraintStrParameter:
      paramValue=STRING;

ConstraintIntParameter:
      paramValue=INT;

ConstraintRefParameter:
      paramValue = [Property];
```

Fig. 13. Constraint usage specification syntax rules

5. Source Code Generator

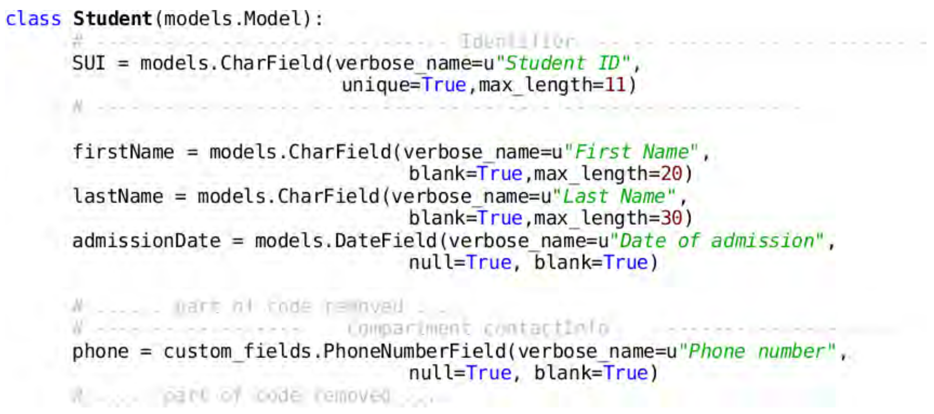
Details of DOMMLite execution semantics are given in the form of a source code generator (model compiler) for the chosen target platform. For the purpose of building the prototype proof-of-concept implementation, Django web framework [29], which is based on Python programming language as our target platform, was chosen.

```
«DEFINE adminClass FOR Entity»
class «name»AdminForm(forms.ModelForm):
      class Meta:
            model = «name»
      «FOREACH allPropertiesWithKeyFields() AS prop-»
            «IF !prop.allConstraints().isEmpty-»

      def clean_«prop.name»(self):
            «prop.name» = self.cleaned_data['«prop.name»']
            «FOREACH prop.eContents.typeSelect(ConstraintSpec)
                  .select(e|ValidatorType.
                        isAssignableFrom(e.type.metaType)) AS validator-»
            «validator.type.name»(«prop.name»«IF !
                        validator.parameters.isEmpty»,«ENDIF»
            «EXPAND validatorParam FOREACH validator.parameters»)
            «ENDFOREACH-»
            return «prop.name»
            «ENDIF-»
      «ENDFOREACH-»
      ... code removed
«ENDDEFINE»
```

Fig. 14. Xpand template fragment for Django admin class generation

The model compiler is defined as a set of templates based on Xpand language, a powerful DSL for defining templates for code generation, which is a part of the openArchitectureWare framework. Figure 14 shows a fragment of the Xpand template used to generate a Django admin form class for every entity in the DOMMLite model.

Figure 15 shows a fragment of a generated, in Django terminology, model. A Django model is generated for every entity in the DOMMLite model and it contains meta-data used for creating tables in the database and for configuring the Django Object-Relational Mapper (DORM). We can see that built-in types are mapped to Django model fields (e.g. for the *firstName*, which is of type *char*, a model field of type *models.CharField* will be generated). Custom types are mapped to custom fields that are specified manually in the *custom_fields* python module (e.g. for the phone field, which is of the user-defined type *phoneNumber*, a model field of type *custom_fields. PhoneNumberField* will be generated).



```
class Student(models.Model):
    # ------------------------------ Identifier ----------------------------------
    SUI = models.CharField(verbose_name=u"Student ID",
                           unique=True,max_length=11)
    # -----------------------------------------------------------------------

    firstName = models.CharField(verbose_name=u"First Name",
                                 blank=True,max_length=20)
    lastName = models.CharField(verbose_name=u"Last Name",
                                 blank=True,max_length=30)
    admissionDate = models.DateField(verbose_name=u"Date of admission",
                                 null=True, blank=True)

    # ...... part of code removed ...
    # -------------------    compartment contactInfo    ---------------
    phone = custom_fields.PhoneNumberField(verbose_name=u"Phone number",
                                 null=True, blank=True)
    # .... part of code removed ...
```

Fig. 15. Fragment of a generated Django model class

In figure 19 we can see that the phone field will be properly validated. *PhoneField* can be used in multiple places throughout the model and the generator will take care to create the right django model fields, which will instantiate the manually created *custom_fields.PhoneNumberField* class. Therefore, the generator doesn't need to be altered for user-defined types. Django will create a synthetic primary key (a column called id usually of the auto-incrementing type if the database supports it) that will be used in the framework for model/entity identification. This synthetic identifier can be accessed by the DORM API and used in service and entity operations.

In figure 16 a fragment of a generated django admin form class is presented. Django uses admin classes to drive the admin application, which is capable of generating CRUD forms on-the-fly from the information supplied in the Django model and admin classes (which are generated in this case). Admin form classes will call validators specified in the DOMMLite model to ensure that field values are validated prior to their storing in the database. Generated validators (Fig. 17) by default fail for every field value.

```
class StudentAdminForm(forms.ModelForm):
      class Meta:
            model = Student

      def clean_SUI(self):
            SUI = self.cleaned_data['SUI']
            isOnlyDigits(SUI)
            mod(SUI,10)
            return SUI
```

Fig. 16. Fragment of a generated Django admin class

```
#PROTECTED REGION ID(mod) START
def mod(value, param0_int):
      raise ValidationError(_(u'Validator "mod" is applied to
                             this field but is not defined yet.'))
#PROTECTED REGION END
```

Fig. 17. Generated default implementation of mod validator

After validator implementation (Fig. 18), the Django admin form will do proper validation of an entered field value (Fig. 19). In this case, validators are implemented as protected regions (a feature of oAW generator) to preserve manual modifications during subsequent generator invocations.

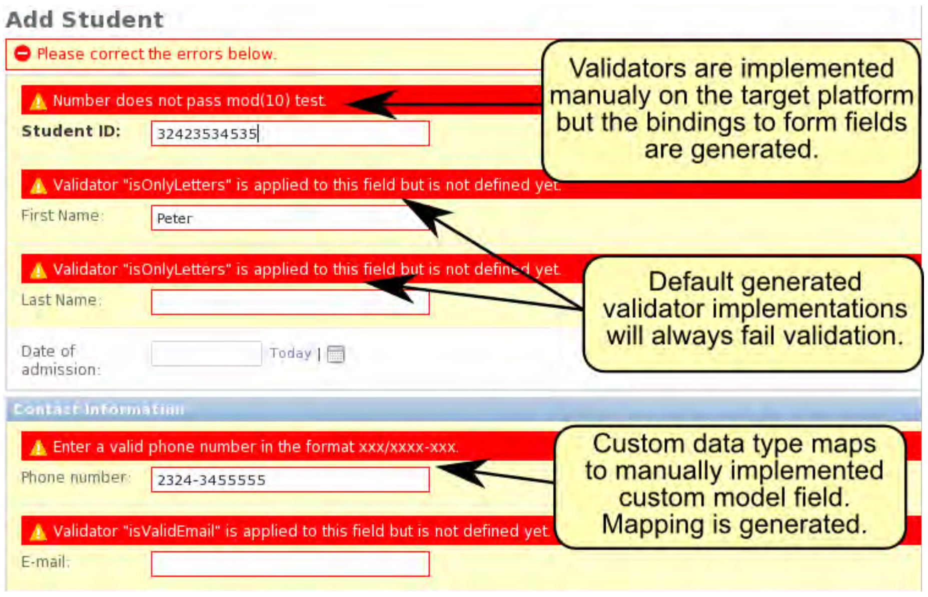
```
#PROTECTED REGION ID(mod) ENABLED START
def mod(value, modulus):
    _sum = 0
    alt = False
    for d in reversed(str(value)):
      d = int(d)
        if alt:
            d *= 2
            if d > modulus-1:
                d -= modulus-1
        _sum += d
        alt = not alt
    if not (_sum % modulus) == 0:
      raise ValidationError('Number does not pass mod(%d) test.' % modulus)
#PROTECTED REGION END
```

Fig. 18. Validator mod manually implemented in Python language

The execution of the Django admin application generates during run-time execution Web based forms (Fig. 19) for basic CRUDS operations, without any manual modifications of the Django application generated from the DOMMLite model. However, more complex business operations and workflows must be implemented on the target platform, as they still cannot be specified in the DOMMLite language.

The skeletons for entity and service operations can also be generated as protected regions also or other capabilities of target platform can be used for mixing generated and manually written source code (e.g. inheritance, or function override in case of Python) [26].

In the presented generator services are mapped to Python modules and service operations are mapped to module functions. Dependency information between services and entities is used to generate the proper import section, which will only import those Django models (DOMMLite entities) that service depends upon. Operations are implemented manually in the Python programming language using the Django framework.


Add Student

Please correct the errors below.

Number does not pass mod(10) test.

Student ID: 32423534535

Validator "isOnlyLetters" is applied to this field but is not defined yet.

First Name: Peter

Validator "isOnlyLetters" is applied to this field but is not defined yet.

Last Name:

Date of admission: Today |

Contact Information

Enter a valid phone number in the format xxx/xxxx-xxx.

Phone number: 2324-3455555

Validator "isValidEmail" is applied to this field but is not defined yet.

E-mail:

Validators are implemented manualy on the target platform but the bindings to form fields are generated.
Default generated validator implementations will always fail validation.
Custom data type maps to manually implemented custom model field. Mapping is generated.

Fig. 19. Django entry form for Student entity with custom validator and field type

6. Eclipse Editor for DOMMLite Models

Using the specification of a language in the form of xText rules as a starting point, openArchitectureWare is capable of generating an almost fully functional Eclipse-based model editor (Fig. 10). In order to achieve full functionality, additional configuration and completion of the generated editor should be performed.

First, we have to define modeling constraints, which will be enforced during model editing as well as during model compilation. For this purpose oAW offers a DSL called Check. Figure 20 shows the Check rule, which ensures that constraint specification is applicable to the given modeling element.

```
context ConstraintSpec ERROR "Constraint " + type.name + " is not applicable to
this model element!" :
      (eContainer.metaType==Entity && type.appliesToEntity) ||
      (eContainer.metaType==Service && type.appliesToService) ||
      (eContainer.metaType==Property && type.appliesToProperty) ||
      (eContainer.metaType==Operation && type.appliesToOperation) ||
      (eContainer.metaType==Parameter && type.appliesToParameter) ||
      (eContainer.metaType==ValueObject && type.appliesToValueObject)
;
```

Fig. 20. Check rule for constraint specification

In order to provide support for the outline view, there are operations that need to be supplemented using the oAW functional language Extend. These operations are *label* - for supplying the node label in run-time, and *image* for supplying node icon. The outline rule for entity is given in figure 21. The outline of a model in Eclipse is presented on the right side of figure 10.

```
String image(emf::EObject this) :
            metaType.name.split("::").get(1) + '.png';

label(dommlite::Entity this) : defaultExtendable(this);

String defaultExtendable(Object this):
      metaType.getProperty("name")!=null ?
            metaType.getProperty("name").get(this).toString() +
            (metaType.getProperty("extends").get(this)!=null ?
                  "<"+((dommlite::NamedElement)metaType.getProperty("extends")
                        .get(this)).name : "") +
                  (((List)metaType.getProperty("depends").get(this)).size>0 ?
                  "("+((List)metaType.getProperty("depends")
                        .get(this)).collect(e|((NamedElement)e).name)
                              .commaSeparated()+")" : "") +shortDesc(this) : null;
```

Fig. 21. Outline rules for the Entity metaclass

```
List[Proposal] completeProperty_oppositeEnd(emf::EObject ctx, String prefix) :
      ((Property)ctx).type.eContents.typeSelect(Property).select(e|
            e.type==((Property)ctx).containingEntity()).
            collect(x|newProposal(x.label(),x.id(),x.image()));
```

Fig. 22. Extend function for supporting bidirectional reference code completion

Code completion is supplemented by the implementation of the *complete** function. Code completion rule for bidirectional references is shown in figure 22. The rule from figure 22 will ensure that only properties from *Classifier* on the other side of relation that reference *Classifier* on this side of the relationship will be offered during code completion.

7. Related Work

Work related to the topics discussed in this paper includes research on design and application of domain-specific modeling languages and domain-specific languages in general to different domains.

DSLs have been most successful and are particularly popular in the domain of embedded systems. In [30] an approach to building embedded component infrastructures is proposed, which is based on the combination of component/container infrastructures (including the underlying communication middleware) with model-driven software development techniques. An example of a DSL for specifying an embedded application (a simple weather station) using MDSD is analyzed. The example metamodel is implemented as an extension of the UML metamodel. DSL concrete syntaxes are UML and XML based. In [31] a PICML DSML is proposed which simplifies and automates many activities associated with developing, and deploying component-based Distributed Real-time Embedded systems. In particular, PICML provides a graphical DSML-based approach which is used to define component interface definitions, specify component interactions, generate deployment descriptors, define elements of the target environment, associate components with these elements, and compose complex DRE systems from these basic systems in a hierarchical fashion. [32] presents a Kiosk Application Generator (KAG), a generator for Kiosk Applications featuring its own templating language for the description of the generated code, as well as a declarative DSL for form-flow based application description. Graphics Adaptor Language (GAL) [33] is a DSL for the generation of video display device drivers.

Integration of tools, data and services is another domain where DSLs have been extensively used. In [34] authors investigate the application of DSLs in the area of tool integration. They have defined a Tool Integration Framework based on the concept of Domain Schema, which is specified as a Model Specification File (MSF) written in a declarative DSL that captures the data model for the various entities and their relationships within a tool. Since it is a data-structure description language, it is in many respects similar to DOMMLite. However, its domain of application is quite different. DSL for MSF, with an aim to be the least-common denominator for different kinds of tools, has a much simpler abstract and concrete syntax. Another application of DSLs for integration purposes is presented in [35]. In this paper the Enterprise-Application Integration approach is described based on predefined components configured with DSL programs (Domain-Abstract Representation). The concrete syntax used for DAR is XML based.

The ideas, which have been used in DOMMLite to support the generation of standardized GUI forms, can be found in [2,3,4,5]. The tool for rapid prototyping of large-scale business information systems presented in [2] uses UML as a design language. The UML model is then transformed to metadata kept in the Application repository. Meta-data in the Application repository is customized by a Form Generator tool, which utilizes this information to generate application source code. Meta-data in the Application repository, although kept in a database and edited by a special-purpose tool, can be considered to be a DSL for the description of GUI forms. Using DOMMLite instead of UML makes this additional step of transforming UML to Application repository and doing further customization unnecessary, since all metainformations needed by a GUI form generator can be specified in the DOMMLite model itself.

Intermediate Form Representation (IFR) [5] is an XML-based DSL for the description of user interface forms, which supports multiple environments for user interface implementation. IFR files can be customized in terms of functionality and layout, while multiple iterations of code generation preserve all manual customizations done in between iterations. IFR files are generated from the Enterprise Java Beans (EJB) data model using Java introspection, making it an interesting alternative in the case of reengineering already existing EJB-based solutions.

In [36] the author investigates using UML as a basis for DSL construction through implementation of a DSL for modeling applications, targeting an already existing business application framework. The language described, although similar in concepts to DOMMLite, lacks support for in-model extensibility and coarse-grained entities (see Sect. 3.5.2), design time validation, run-time validation. As concluded by the author, UML, being a general-purpose modeling language, is not very well suited for the construction of DSLs.

AndroMDA [37] is an open-source MDA framework capable of generating source code for different platforms out of models specified in the UML. A model is defined using UML with stereotypes and exported to XMI format. From there a maven-driven build process parses it and the source code is generated by executing so-called cartridges (modules that perform code generation).

The language most alike DOMMLite, in terms of concepts and tools used, is Sculptor [38]. Sculptor is developed as a part of the open-source Fornax platform[8]. It is actively developed and documented, and is capable of generating source code for desktop RCP and Web applications with support for different technologies such as Hibernate, Spring, EJB, JSP, JFC etc. which is a very good option for teams that need to target different platforms and do not have time for development of source code generators. Being an open-source project, it also brings high quality of generated source code, as the templates are peer reviewed by all contributors and users. Sculptor follows Domain-Drive Design concepts very closely. As of this writing there seem not to be any published scientific papers describing Sculptor in more detail. In comparison to DOMMLite it features out-of-the-box source code generators for different popular technologies but at this time it does not have support for coarse grained entities (see Sect. 3.5.2) nor in-model extensibility (see Sect. 3.9).

8 http://www.fornax-platform.org/

8. Conclusions and Future Work

In this paper we have presented an extensible DSL implemented using MDE ideas and techniques, which is well suited for database-oriented business applications. From a DOMMLite model an entire application with navigation, and CRUDS operations can be generated. By using highly abstract languages that are based on concepts from the domain at hand, significant productivity increase can be achieved. The code generator propagates any change made in the model to all generated files, making the implementation consistent with the model and eliminating errors due to human factors, which inevitably occur if these changes are done manually. We have anticipated code quality improvement, since coding guidelines for generated code are enforced by templates and improvements in templates are immediately reflected on all generated code.

Hand-written code still needs to be maintained manually, which may lead to errors caused by misalignment to the generated code. However, using statical code analysis offered by contemporary integrated development environments, as well as test-driven development techniques, many bugs can be identified and eliminated quickly.

DOMMLite is designed from the ground up to be simple to use and simple to extend. By implementing its extension mechanism, it is possible to extend the language semantics on the model level without changing the metamodel and source code generator. This mechanism is fully supported by the eclipse editor with code completion and constraint validation for newly defined constraint types.

The development of DSLs and DSMLs is nowadays significantly simplified with the appearance of generator frameworks such as openArchitectureWare, which permit languages to be designed and supporting editors and compilers to be prototyped more quickly.

Further research and development is focused on extending the language and the functionality of the supporting tools. We plan to: (1) – extend the language to support behavioral elements (operation semantics), (2) – define the visual syntax of the language, as well as a supporting graphical editor, (3) – support additional target platforms, (4) – implement support for model version control, (5) – implement support for model and language evolution.

References

1. openArchitectureWare Generator Framework. [Online] Available: http://www.openarchitectureware.org/ (current December 2008)
2. Milosavljević, G., Perišić, B.: A method and a tool for rapid prototyping of large-scale business information systems, Computer Science and Information Systems ,vol. 1, pp. 57-82 (2004)

3. Milosavljević, B., Vidaković, M., Komazec, S., Milosavljević, G.: User interface code generation for data-intensive applications with ejb-based data models, in Software Engineering Research and Practice (SERP'03), Las Vegas, NV, (2003)
4. Milosavljević, B., Vidaković, M., Konjović, Z.: Automatic Code Generation for Database-Oriented Web Applications, pp. 89-97. Recent Advances in Java Technology: Theory, Application, Implementation, Trinity College Dublin (2003)
5. Milosavljević, B., Vidaković, M., Komazec, S., Milosavljević, G.: User interface code generation for ejb-based data models using intermediate formrepresentations, in ACM Principles and Practice of Programming in Java,(Kilkenny, Ireland) (2003)
6. Schmidt, D. C.: Guest editor's introduction: Model-driven engineering, Computer, vol. 39, no. 2, pp. 25-31 (2006)
7. Tolvanen, J.-P., Sprinkle, J., Gray J.: The 6th oopsla workshop on domain-specic modeling, in OOPSLA '06: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications , (New York, NY, USA), pp. 622-623, ACM (2006)
8. Booch, G., Brown, A., Iyengar, S., Rumbaugh, J., Selic, B.: An MDA manifesto, MDA Journal (2004)
9. Dejanović, I.: Meta-model, model editor and business application generator, (Master's thesis). Author's reprint, Library of Faculty of Technical Sciences, Novi Sad, Serbia (2008)
10. Bézivin, J., Jouault, F., Kurtev, I., Valduriez , P.: Model-based DSL frameworks, Companion to the 21st Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA, pp. 22-26 (2006)
11. OMG Unified Modeling Language (OMG UML), Infrastructure. Version 2.1.2, Final Adopted Specication, OMG Document formal/2007-11-04 (2007)
12. Meta Object Facility (MOF) Core Specication. Version 2.0 (2006)
13. Eclipse Modeling Framework - EMF. Online, Available: http://www.eclipse.org/ modeling/emf/ , (current December 2008)
14. Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software . Addison-Wesley Professional (2004)
15. Efftinge S., Völter, M.: oAW xText: A framework for textual DSLs, in Eclipse Summit 2006 Workshop: Modeling Symposium (2006)
16. Van Deursen, A., Visser, J.: Domain-specic languages: an annotated bibliography, ACM SIGPLAN Notices , vol. 35, pp. 26-36 (2000)
17. Wile, D. S.: Supporting the dsl spectrum, JOURNAL OF COMPUTING AND INFORMATION TECHNOLOGY , vol. 9, pp. 263-288 (2001)
18. Hatton, L.: The t experiments: Errors in scientic software, IEEE COMPUTATIONAL SCIENCE & ENGINEERING , vol. 4, no. 2, pp. 27-38 (1997)
19. Mernik, M., Sloane, A. M.: When and how to develop domain-specic languages, ACM Computing Surveys (CSUR) , vol. 37, pp. 316-344 (2005)
20. Fowler, M.: Domain specic languages. Online, Available: http://martinfowler.com/ dslwip/ (current December 2008)
21. Cunningham H. C.: A little language for surveys: Constructing an internal dsl in ruby, ACM-SE 46, Auburn, Alabama, USA (2008)
22. Paul, R.: Designing and implementing a domain-specic language, LinuxJ. , vol. 2005, no. 135, p. 7 (2005)
23. Kabanov, J., Raudjärv, R.: Embedded typesafe domain specic languages for java, pp. 189-§197, ACM New York, NY, USA (2008)

24. Hudak , P.: Building domain-specic embedded languages, ACM Computing Surveys (CSUR) , vol. 28, p. 196 (1996)
25. Haugen, Ø., Mohagheghi, P.: A Multi-dimensional Framework for Characterizing Domain Specific Languages, Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling (DSM'07), Computer Science and Information System Reports (2007)
26. Völter, M., Stahl, T.: Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons (2006)
27. Bauer, C., King, G.: Hibernate in Action, Manning Publications (2004)
28. Fowler, M.: Inversion of control containers and the dependency injection pattern. Online, Available: http://www.martinfowler.com/articles/injection.html (current February, 2008) (2004)
29. Django web framework. Online, Available: http://www.djangoproject.com/, (current December, 2008).
30. Vöelter, M., Salzmann, C., Kircher, M.: Model driven software development in the context of embedded component infrastructures, LECTURE NOTES IN COMPUTER SCIENCE , vol. 3778, p. 143 (2005)
31. Balasubramanian, K., Balasubramanian, J., Parsons, J., Gokhale, A., Schmidt, D. C.: A platform-independent component modeling language for distributed real-time and embedded systems, Journal of Computer and System Sciences , vol. 73, pp. 171§-185 (2007)
32. Živanov, Ž., Rakić, P., Hajduković, M.: Using code generation approach in developing kiosk applications, Computer Science and Information Systems , vol. 5, pp. 41-§59 (2008)
33. Thibault, S. A., Marlet, R., Consel, C.: Domain-Specic Languages: From design to implementation application to video device drivers generation, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, pp. 363§-377 (1999)
34. J. Gray and G. Karsai, An examination of dsls for concisely representing model traversals and transformations, p. 10 (2003)
35. Nussbaumer, M., Freudenstein, P., Gaedke, M.: The impact of domain-specific languages for assembling web applications, Engineering Letters, vol. 13 (2006)
36. Anonsen, S.: Experiences in modeling for a domain specic language, UML Modeling Languages and Applications , vol. 3297/2005, pp. 187§-197 (2005)
37. AndroMDA - MDA framework. Online, Available: http://www.andromda.org, (current December 2008)
38. Enterprise java community: Improving developer productivity with sculptor. Online, Available:http://www.theserverside.com/tt/articles/article.tss?l=ProductivityWithSculptor, (current June 2007) (2007)

Igor Dejanović received his M.Sc. (5 years, former Diploma) degree from the Faculty of Technical Sciences in Novi Sad. He completed his Mr (2 year) degree at the University of Novi Sad, Faculty of Technical Sciences. Currently, he works as a teaching assistant at the Faculty of Technical Sciences at the University of Novi Sad, where he assists in teaching several Computer Science and Software Engineering courses. His research interests are related to Domain-Specific Languages, Model-Driven Engineering and Software Configuration Management.

Gordana Milosavljević is a teaching assistant and Ph.D. student at University of Novi Sad, Faculty of Engineering, Computer Sciences Department. She has received her B.Sc. and M.Sc. from University of Novi Sad, Faculty of Engineering, Computer Sciences Department. Her research interests focus on software engineering methodologies, rapid development tools and enterprise information systems design.

Branko Perišić is an associated professor at University of Novi Sad, Faculty of Technical Sciences. He has received his engineer diploma from University of Sarajevo, Faculty for electrical engineering, M.Sc. and PhD diplomas from University of Novi Sad, Faculty of Technical Sciences. He is currently a Computer center manager and leads Software development team at Faculty of Technical Sciences. As a teaching professor he has developed and teached a variety of Computer Engineering, Software Engineering and Information System Design courses at different Universities. His major research interests are related to Model Driven Software Development, Business Information Systems Design, Software Configuration Management and Secure Software Design.

Maja Tumbas is a teaching assistant at the Faculty of Technical Sciences at the University of Novi Sad, where she received her M.Sc. degree. Her fields of interest include different areas of software engineering, including software modeling and computer security.

Received: February 03, 2009; Accepted: December 18, 2009.

UDC 519.8, DOI:10.2298/CSIS090422005S

Prompt System Redesign: Shifting to Open Source Technology to Satisfy User Requirements

Igor Svetel[1], Aleksandar Đurović[2], and Vencislav Grabulov[3]

[1] Innovation Center, Faculty of Mechanical Engineering, Kraljice Marije 16, Belgrade, Serbia
isvetel@mas.bg.ac.rs
[2] Society for Structural Integrity and Life DIVK, Belgrade, Serbia
aca@divk.org.rs
[3] Institute for Materials Testing - Institute IMS, Bulevar vojvode Mišića 43, Belgrade, Serbia
venciaiv@eunet.rs

Abstract. The paper describes a redesign project undertaken in a short period to adapt a software system to user needs. Additional goals of the project included a shift to Open Source software and the selection of technology to enable sustainable system development. The paper chronologically describes all phases of the project and provides reasons for all decisions taken during the development process. The paper concludes with a discussion of the merits of the redesign methodology.

Keywords: System redesign; Open Source; Welder's Passport; JEE; EJB; JSF.

1. Introduction

System redesign is the process of changing the structure of a system while preserving its external behavior. In the field of software development this process is referred as 'model refactoring' [1] as distinct from the term 'refactoring' [2], which focuses on the modification of software code. The term 'redesign' is used in this paper to emphasize the architectural point of view and to highlight dilemmas and decisions that go beyond software construction.

The main intention of the redesign task was the development of an agile or "change resistant" system [3]. The experience with the first system demonstrated that users are not able to clearly identify their requirements before they start to use the software. Further, the rate of change in software technology is so rapid that no one can expect that a system will hold to the same technology for its lifetime.

Since the time frame for the project was short, it was necessary to find a method that would allow the reuse of existing components. This goal has a major impact on achieving deadlines since it enables the adoption of a sustainable design methodology. The redesign philosophy was organized around the notion that both the design process and the resulting software should be amenable to repeatable change cycles and yet enable the reuse of existing components. One way to achieve these requirements is strict adherence to standards and rejection of all non-standard extensions to the technologies.

The sections in the paper are organized in a sequence that reflects the chronological progression of the redesign process as accurately as possible. Yet, the linear nature of the exposition prevents illustration of many decision loops that were part of the process.

2. Former System

Welder’s Passport (WP) 1.0 was a commercial implementation of the concept demonstration software developed as part of the E!2774 project. The EUREKA E!2774 project involved numerous institutions from European countries. The application domain was a computer information system that would enable the monitoring of a welder’s career from the beginning of his/her training, through gaining certification and during his/her professional carrier until the end of welding activities. The main result of the project was a data model that encompasses all necessary information. The model gained endorsement from all parties in the international welding industry [4].

The WP 1.0 system inherited Oracle as the DBMS from the demonstration software. The choice of all other technologies was driven by the decision to use JDeveloper as the integrated development environment (IDE). Struts was the main development paradigm supported in JDeveloper at the time of the implementation. All other technologies that were applied in the project were those provided as default IDE settings: Oracle ADF, OC4J, and JSP. CSS and JavaScript technologies were included in the final development phase, as the ad hoc solution to the client’s need for a more flexible user interface [5].

The E!2774 project supported only the certification scheme approved by the European Federation for welding, joining and cutting (EWF) [6]. Since in Serbia and the surrounding region various certification schemes exist in parallel, it was necessary to include additional data that would enable management of other certification schemes in addition to the approved certification scheme.

The system consists of three separate web interfaces connected to a central database (Fig. 1). The Company interface enables management of the information regarding welders, welding engineers and their certificates. The ATB interface is dedicated to accredited training bodies and enables them to manage information regarding training courses. The ANB interface is designed for accredited national bodies and enables management of the certification process that a particular ANB supervises.

The main criticism was that the WP 1.0 system was driven by the futuristic vision of the joined multinational market in which welders freely move from one project to another. The reality of the market is diverse. Companies that employ welders usually treat their personnel and their knowledge as the company's competitive advantage. They are frightened by the notion that data about welders and their certificates are kept on some computer server located outside the premises of the company, no matter how much the security of the underlying software system is promised. A second criticism voiced by all organizations was that the whole certification process was an unnecessary system feature. Everyone wanted a system that would provide an advantage for their company, and the concept that a system's functionality should depend on all stakeholders was inconceivable. A third criticism was the high cost of the Oracle (database management system (DBMS)) license that was considered prohibitive by companies that insisted on implementing the system in their premises.

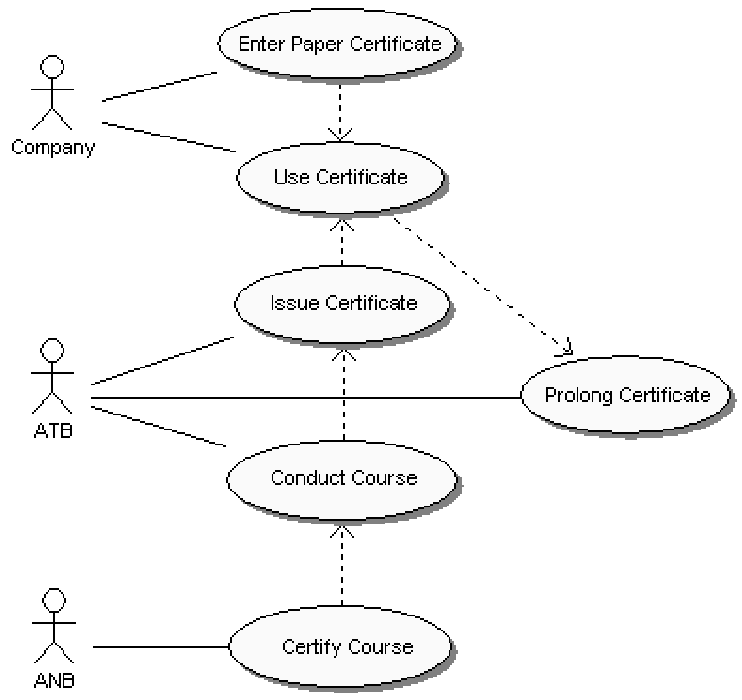


Fig. 1. Structure of the certification scheme

3. System Redesign

The main intent of the system redesign was the inclusion of the system architect in the development team. The development team responsible for the

WP 1.0 system included programmers and welding specialists. Unfortunately, they did not undertake any market research, but based their decisions on the expectations that a maximum level of improvement and change would be welcomed by all users and that technology is a remedy for all problems. The failure of the final product in the market puzzled them because of the positive reactions to their E!2774 research results.

The first phase in the system redesign focused on changing the expectations and habits of all team members. Welding specialists were fond of the certification scheme implementation. It was necessary to conduct a few specially prepared WP 1.0 demonstrations to potential users to persuade welding specialists that the certification scheme implementation was an unnecessary feature. When asked about feature that was of greatest interest to them, the potential users pointed to the process of certification management (i.e., creating, editing, and using). Even ATBs and ANBs perceived certification management as the best feature of the system. All other aspects were seen either as unnecessary or as a potential threat to their business. The programmers had a different set of prejudices. They were proud of their achievement in implementing the complex structure of the certification scheme, and thought that the users should be willing to adapt their usual way of working to the software. Accordingly, they perceived the rejection of the certification scheme as an insult, and it took a great deal of effort to persuade them that the design of a system that is more responsive to user needs will create enough programming challenges.

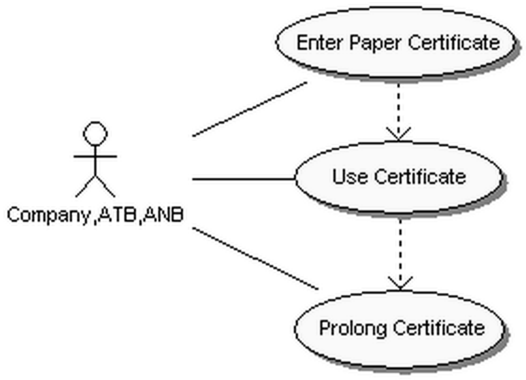


Fig. 2. Structure of the WP 2.0 system

Once the development team was persuaded, the design of WP 2.0 system was outlined. The decision was made that only the certification management process would be implemented as the application running on the stand-alone computer (Fig. 2). In addition it was decided that the full scope of the WP 1.0 database would be implemented, thus preserving compatibility with all of the certification schemes. Also, instead of a single computer application, the full client server technology would be implemented, although hidden from the user. These decisions were taken to enable the evolution of the system as the user needs increase. It is easy to foresee that once a system has been accepted a need for more workstations will quickly surface and, in the not so distant future, there will be an increasing desire by users to access the database from different locations. By implementing a client-server architecture in the first place, the system will easily satisfy all future expansion needs. Also, by preserving a full database structure, the system will support certification schemes once such a need arises.

In this way the development methodology shifted from a strictly sequential model to an iterative model [7]. WP 2.0 was conceived from the start as an ever-changing system. Some changes were easy to predict, like the expansion of the scope of the system and the inevitable change of the underlying technologies. Others, which will occur during the system's lifetime cannot be predicted in advance. Under these circumstances the only solution is to design an adaptive system and adopt an adaptive software development methodology [8]. Accordingly, the term 'change' became a high priority criterion in the whole decision-making process of WP 2.0. All considerations made during system development were finally evaluated in regard to their ability to adapt to the change, and only those solutions that were scored by all development team members as most adaptable were accepted.

4. Selection of the Technology

In answer to the high cost of software licenses, a shift to Open Source software was proposed. At first sight this appeared to be an easy task since all necessary replacement technologies existed as Open Source software. However, this initial optimism quickly vanished as we were faced with the selection of the most appropriate software from the vast amount of available technologies. It soon became obvious that the selection of the proper technologies would be crucial for the success of the redesign project. Since the project had only three months to deliver the new system, it was decided that two months should be allocated for the selection of the technology, since all of the team members were new to the domain of Open Source software.

The selection of the DBMS was based on the positive experience that the whole Open Source community appeared to have had with the MySQL DBMS. Since the maximum size of the MySQL database was not a limiting factor and the support for Unicode existed, it was decided to use this DBMS technology without considering other options. Since MySQL is mature and well-supported technology and is used by many development teams, it was estimated that it would facilitate the software system to be adapted to all future changes.

The decisions on the programming language and the platform for server programming was also straightforward. The same technologies as in WP 1.0

were selected, namely Java and Java Enterprise Edition (JEE). Both Java and JEE are well supported and are clearly among the most adaptable technologies on the market.

The next technology that was taken into consideration was IDE. Although programmers opted for JDeveloper, an environment with which they where already familiar, it was precisely this familiarity that led to the rejection of this technology. In the previous project, the team had been accustomed to adopt any Oracle technology that the JDeveloper system provided as a default. The rejection of JDeveloper was clear sign to them that the new system should not have any resemblance to the old one. In addition, the programmers admitted that every time a new version of JDeveloper arrived they had many problems to migrate the existing version of the system to the new software version. After this decision was taken, two IDEs were considered: Eclipse and NetBeans. Both were seen to provide the same functionality as JDeveloper, and they both seemed to have the same level of adaptability based on the number of supporting companies. The final decision, in favor of Eclipse, was taken much later after the persistence technology had been chosen, since Eclipse had slightly better support for EJB 3.0.

The programmers' tendency to include, somewhat unsystematically, all kinds of technologies in their IDE required some preventative measures. The system architect had a separate development system on which only selected technologies were installed, and used that system to regularly test if he could replicate the programmers results. This precaution prevented programmers from finding easier paths and applying methods that differed from accepted standards. During the development of the system an inconsistency was detected only once, when programmers made their own decision on the inclusion of the technology for PDF creation.

The selection of version control technology was also undertaken with a view to controlling the activities of the programmers. In the WP 1.0 system the programmers had attempted to use CVS technology but had found some inexplicable errors that probably reflected their unwillingness to be controlled. For that reason, the Subversion version control technology was selected without consultation of the programmers. The rationale behind this selection was that it is a new technology developed to overcome the shortcomings inherent in CVS. That fact forestalled any arguments about errors in CVS. At the time of selection Subversion was not among the default version control technologies used in any IDE, but the strength of supporting companies promised its adaptability. This proved to be good forecast since today Subversion is included as a default version control system in all major IDEs. Once Subversion was introduced as the version control technology, a VisualSVN Server 1.1 [9] was installed as a central repository and Subversive [10] was selected as the Eclipse client. Subversion proved itself as a robust technology and the programmers accepted it and were pleased with its automatic merging features. Only once did a problem with synchronization occur, when a programmer took a copy of the whole system, moved it to his home computer, and returned with a modified version. This situation served as a good example that strict adherence to specified technologies is the single most important means to achieve good results.

Because we decided to use a traditional relational DBMS for data storage in combination with Java-based object-oriented technology to access data, the issue of selecting an object-relational mapping technology became an important topic. Two technologies where considered, Hibernate 3.0 and Enterprise JavaBeans 3.0 (EJB 3.0). EJB was the first attempt to provide a standard solution to problems of persistence, transactional integrity, and security. Regardless of the fact that IBM and Sun Microsystems backed this technology, it was perceived to be too complex. Hibernate is one among several technologies developed as an answer to the shortcomings of earlier EJB versions. The technology gained widespread acceptance, and for some time was considered as the best persistence technology. As part of the EJB 3.0 development (JSR 220), the Java Persistence API (JPA) standard was developed as the merger of expertise from TopLink, Hibernate, JDO, and EJB developers. Since JPA accumulates the best properties of all previous persistence technologies and is supported by largest software companies, we decided to select EJB 3.0 as the persistence technology for our project. TopLink Essential [11] was selected as the persistence provider (JPA implementation) because it was the EJB 3.0 reference implementation.

Selection of the web application framework took most of the time allocated for the technology selection process. The number of available technologies is large and any one technology can have several implementations offering different functionality, thus making the selection a challenging task. To reduce complexity we sorted out only those frameworks that are based on the Java programming language. From those, based on popularity in the developer community, we considered the following frameworks: JavaServer Faces, JBoss Seam, Apache Struts, Spring Framework, and Tapestry. After a first round of consideration, Spring and Tapestry were rejected because their compatibility with already selected technologies was not straightforward. Finally, JavaServer Faces was selected as the technology that offers just enough necessary functionality and does not goes into unnecessary extensions. In addition, it offers the cleanest integration with the set of already selected technologies. Once a technology was selected, we had to decide on a particular implementation. This task was particularly difficult because each implementer typically adds their particular component libraries in addition to the basic ones. Fortunately, a live demonstration for each technology can be found on the Internet, and the required functionality can be compared with the actual one. To accomplish this task we used paper and pencil to draw the required user interface. Then, using live demonstrations, we investigated whether a particular implementation could render all necessary components. We finished with two technologies ICEfaces [12] and RichFaces [13] as providing all of the necessary components. The final decision was based on the development of prototype WP systems using both technologies. Both of the prototype systems provided all of the necessary functionality, with similar ease of use. Finally, a slight delay in supporting the latest version of the development technology offered by ICEfaces decided in favor of RichFaces.

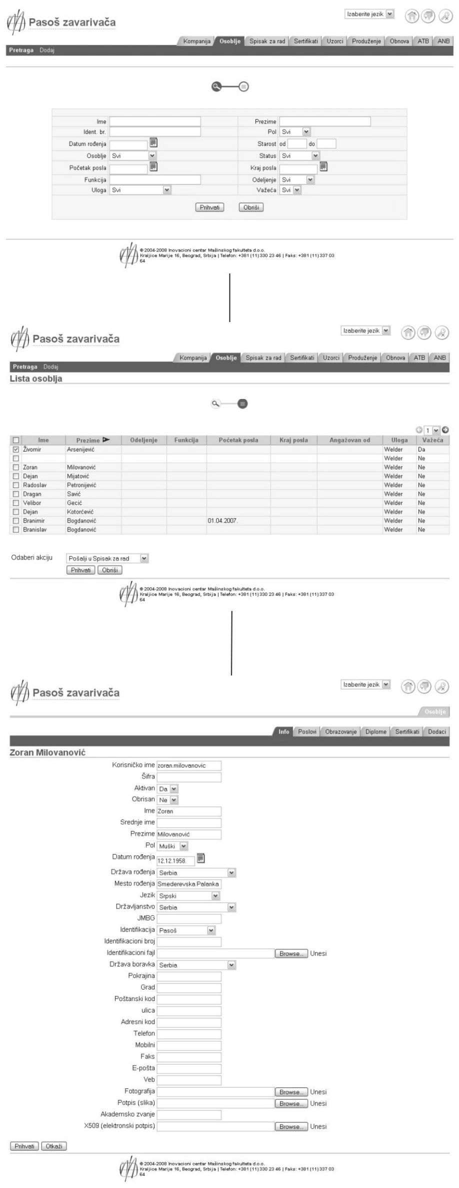
The final decision on the application server took into account all of the selected technologies. The GlassFish server [14] (GlassFish Project - V2 UR2 Final Build) was seen to provide the appropriate combination of support for the selected technologies and appeared to guarantee that its technology would accommodate future changes.

The complete technology selection process took almost two thirds of the time allocated for the project and proved to be a most challenging task. The Open Source community offers many alternative technologies that cover the same functionality and services. To accomplish the selection task, we relied on three types of criteria: a) whether the technology provides all of the required functionality, b) the degree of market acceptance, and c) market strength of the company supporting the particular technology. Those criteria combined with different levels of technology research (information found on the Internet, technical documents, demonstration versions) proved to be sufficient in all cases.

5. GUI Redesign

The WP 1.0 system used a default graphical user interface (GUI) provided by JDeveloper. It was based on tables, bitmap tabs and buttons. The developers did not consider ease of use and system look and feel as important features. Instead, they placed all of their efforts on functionality, with the belief that this is what the market is waiting for. After the first users struggled to enter their data, it became obvious that a more flexible GUI was needed. The user interface in WP 1.0 supported synchronous loading of content. This was often a frustrating experience for the user, because incorrectly entered data were detected only after all of the data had been entered and an attempt was made to submit the data to the database. In addition, a system or communication error during a data entry process required all of the data to be entered again from the beginning.

When the final decision to develop the WP 2.0 system was made, the need for a new GUI was considered as important a feature as the transition to Open Source technologies. By that time it had become obvious that to gain the approval of the users a Web-based application would need to look and feel as any other OS-based computer application. The main GUI change was rejection of the single level tab-based interface (Fig. 3). Instead, a visually recognizable, hierarchical interface based on windows and dialog boxes was adopted (Fig. 4). This design provides clear navigation. At any point in time, the users know on which type of data they are working. This is important because the user needs to consult many external documents to gather data required for entry into the database. During that process, the users are likely to turn away from the computer screen, and sometimes even need to leave their workstation. It is important for maintaining productivity that the users have a clear understanding of the data elements on which they are working upon returning their attention to the computer screen. The selected JSF technology and RichFaces implementation enabled us to meet that requirement.



Ime		Prezime	
Ident. br.		Pol	Svi
Datum rođenja		Starost	od do
Osoblje	Svi	Status	Svi
Početak posla		Kraj posla	
Funkcija		Odeljenje	Svi
Uloga	Svi	Važeća	Svi

© 2004-2008 Inovacioni centar Mašinskog fakulteta d.o.o.
Kraljice Marije 16, Beograd, Srbija | Telefon: +381 (11) 330 23 46 | Faks: +381 (11) 337 03 64

Lista osoblja

	Ime	Prezime	Odeljenje	Funkcija	Početak posla	Kraj posla	Angažovan od	Uloga	Važeća
✓	Živomir	Arsenijević						Welder	Da
								Welder	Ne
	Zoran	Milovanović						Welder	Ne
	Dejan	Mijatović						Welder	Ne
	Radoslav	Petronijević						Welder	Ne
	Dragan	Savić						Welder	Ne
	Velibor	Gecić						Welder	Ne
	Dejan	Kotorčević						Welder	Ne
	Branimir	Bogdanović			01.04.2007.			Welder	Ne
	Branislav	Bogdanović						Welder	Ne

Odaberi akciju: Pošalji u Spisak za rad

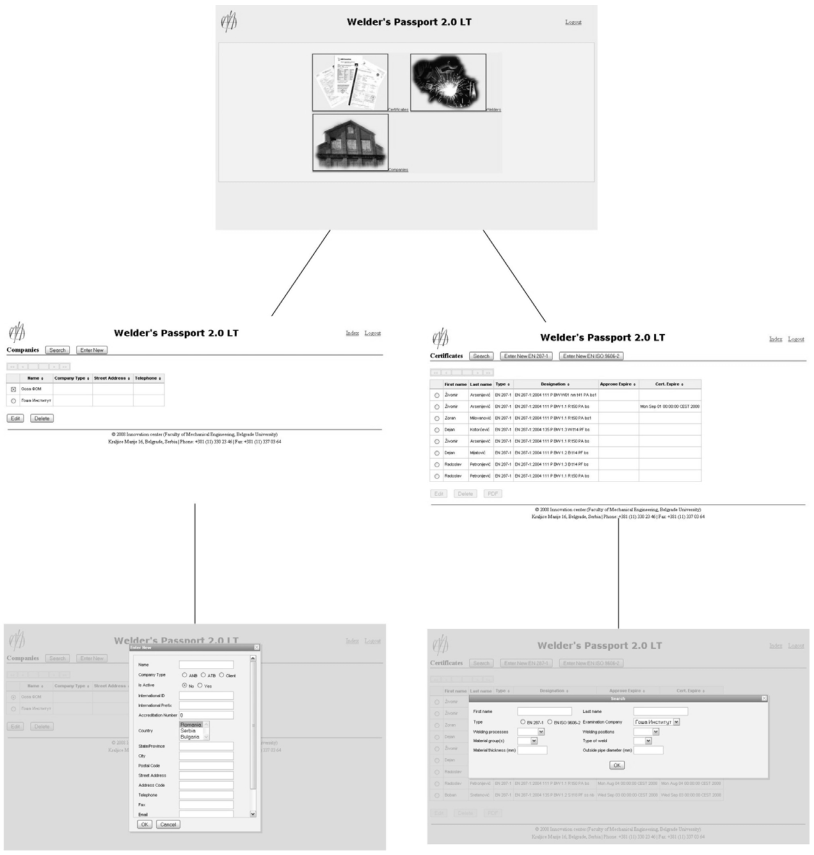
© 2004-2008 Inovacioni centar Mašinskog fakulteta d.o.o.
Kraljice Marije 16, Beograd, Srbija | Telefon: +381 (11) 330 23 46 | Faks: +381 (11) 337 03 64

Zoran Milovanović

Korisničko ime	zoran.milovanovic
Šifra	
Aktivan	Da
Obrisan	Ne
Ime	Zoran
Srednje ime	
Prezime	Milovanović
Pol	Muški
Datum rođenja	12.12.1958.
Država rođenja	Serbia
Mesto rođenja	Smederevska Palanka
Jezik	Srpski
Državljanstvo	Serbia
JMBG	
Identifikacija	Pasoš
Identifikacioni broj	
Identifikacioni fajl	
Država boravka	Serbia
Pokrajina	
Grad	
Poštanski kod	
ulica	
Adresni kod	
Telefon	
Mobilni	
Faks	
E-pošta	
Veb	
Fotografija	
Potpis (slika)	
Akademsko zvanje	
X509 (elektronski potpis)	

© 2004-2008 Inovacioni centar Mašinskog fakulteta d.o.o.
Kraljice Marije 16, Beograd, Srbija | Telefon: +381 (11) 330 23 46 | Faks: +381 (11) 337 03 64

Fig. 3. Visually linear nature of the WP 1.0 interface



	Name	Company Type	Street Address	Telephone
●	Goša ФОМ			
○	Гоша Институт			

© 2008 Innovation center (Faculty of Mechanical Engineering, Belgrade University)
Kraljice Marije 16, Belgrade, Serbia | Phone: +381 (11) 330 23 46 | Fax: +381 (11) 337 03 64

	First name	Last name	Type	Designation	Approve Expire	Cert. Expire
○	Živomir	Arsenijević	EN 287-1	EN 287-1:2004 111 P BW W01 nm t41 PA bs1		
○	Živomir	Arsenijević	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs		Mon Sep 01 00:00:00 CEST 2008
○	Zoran	Milovanović	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs1		
○	Dejan	Kotorčević	EN 287-1	EN 287-1:2004 135 P BW 1.3 W t14 PF bs		
○	Živomir	Arsenijević	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs		
○	Dejan	Mijatović	EN 287-1	EN 287-1:2004 111 P BW 1.2 B t14 PF bs		
○	Radoslav	Petronijević	EN 287-1	EN 287-1:2004 111 P BW 1.3 B t14 PF bs		
○	Radoslav	Petronijević	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs		

Enter New

Name	
Company Type	○ ANB ○ ATB ○ Client
Is Active	[✓ No] ○ Yes
International ID	
International Prefix	
Accreditation Number	0
Country	[✓ Romania] Serbia Bulgaria
State/Province	
City	
Postal Code	
Street Address	
Address Code	
Telephone	
Fax	
Email	

Search

First name		Last name	
Type	○ EN 287-1 ○ EN ISO 9606-2	Examination Company	Гоша Институт
Welding processes		Welding positions	
Material group(s)		Type of weld	
Material thickness (mm)		Outside pipe diameter (mm)	

○	Radoslav	Petronijević	EN 287-1	EN 287-1:2004 111 P BW 1.1 R t50 PA bs	Mon Aug 04 00:00:00 CEST 2008	Mon Aug 04 00:00:00 CEST 2008
○	Boban	Sretenović	EN 287-1	EN 287-1:2004 135 P BW 1.2 S t18 PF ss nb	Wed Sep 03 00:00:00 CEST 2008	Wed Sep 03 00:00:00 CEST 2008

Fig. 4. Visually hierarchical nature of the WP 2.0 interface

The next major change was adoption of more flexible interface methods. In the WP 1.0 system, an ad hoc inclusion of CSS and JavaScript was applied. In WP 2.0, a more systematic approach in the form of Ajax was applied, particularly the way Ajax is implemented in RichFaces in the form of the Ajax4jsf framework. The main objective was to speed up the data entry process by automating it. It is irritating for any user to have to enter the same data repeatedly. For example, in WP 1.0 the name of the city for every person living in the same town had to be entered repeatedly, because the interface had only HTML input fields or dropdown lists. An interface based on Ajax provides input fields with various data suggestion mechanisms based on asynchronous data retrieval methods that make data entry a less frustrating experience.

A very important feature of RichFaces is ‘skinnability’ – the ability to quickly change graphical appearance of the GUI. The look of the application is not a superficial system property. Users often have problems to shift their manual work process to a computer-based process if the computer-based process does not resemble the previous work environment. At this time it appears that the WP 2.0 users have accepted the new GUI, but we have already received a few suggestions that the printed-paper version of the certificate should precisely match the existing certificate form if we wish a company to start considering our system.

6. Database Reimplementation

The database reimplementation process was straightforward. The database structure in the WP 1.0 system was defined using the SQL language. So, for the new version the same code was used with minor modifications relating to data types (Table 1). During data import into the MySQL DBMS, a maximum row size error emerged as the problem. After reviewing the database structure it was revealed that the programmers were too comfortable in Oracle DBMS environment and had defined unnecessarily large fields. After reducing their size, a new database was implemented without any loss in functionality.

Table 1. Data type differences in the WP 1.0 and WP 2.0 systems

WP 1.0	WP 2.0
Oracle	MySQL
NUMBER	INT, FLOAT
NVARCHAR2	VARCHAR
ORDIMAGE, ORDDOC	LONGBLOB

Another point in using MySQL technology is Unicode support. The default system installation does not support Unicode. It is necessary to define Unicode support during the installation process and afterward to manually change a few `.ini` files. After that the Unicode support worked flawlessly.

7. Code Reimplementation

The actual code implementation exploited much of the code from the previous software version. Fortunately, the composition of the programming team was the same in both projects. During the technology selection process, the team recognized many similarities between the old and new technologies, particularly on the code level, and they used their findings to reuse as much code as possible. While in the previous project the programmers applied code that suited their needs and used many non-standard extensions, the restriction that only code conforming with standard definitions could be applied was imposed for the development of WP 2.0. The principal reason for this decision was to ensure the creation of the code that would be adaptable to future changes.

The programmers established an analogy between the systems based on the Model–View–Controller (MVC) pattern (Fig. 5). The Model part consisted of the Oracle ADF Business Component in the WP 1.0 system and of the EJB 3.0 JPA in the WP 2.0 system (Table 2). The first phase of the code reimplementation consisted of building the JPA entities. This process is automated in Eclipse IDE and requires the programmer to point to appropriate tables in the database for the creation of all relevant classes. Some modifications were necessary in the case of the `int` data type that needed modification to `integer` to support a `null` value. The `timestamp` data type needed manual conversion. Session bean functionality was written from scratch. Since the SQL code was located in the core of the software structure, the programmers established an analogy with the WP 1.0 code and managed to reuse it. The lack of the View object in the EJB 3.0 implementation required additional effort in programming entity objects that combined the data from different tables. Much of the Java code relating to business rules, as well as logical and numeric control was reused in the new system.

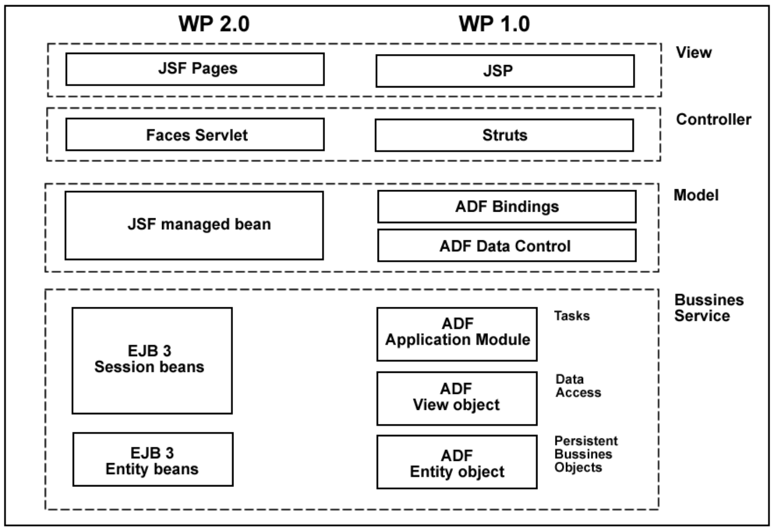


Fig. 5. Model–View–Controller (MVC) pattern of the WP 1.0 and WP 2.0 systems

On the view-controller side of the system, an analogy between Struts and JSF was established. This phase of the code reimplementation required a great deal of new programming.

8. Design Method

The redesign process shares many similarities with the design process. The main difference is that the design process starts from the need for a new system, while the redesign process is the consequence of dissatisfaction with the existing system. However, in respect to methodology, the same method can be applied in both cases.

The initial software design method depicted the process as the linear sequence of activities [15]. This is known as the “waterfall model”. Main phases (i.e., analysis, design, coding, testing, and operation) followed each other and continuation with the next phase was allowed only after the previous phase has been complete. Although this method guaranties an excellent final product, it was recognized that often it is impossible to define all of the requirements in advance because the users cannot fully describe their needs before they have had some preliminary experience with the software system.

The next generation of design methods has adopted iterative and incremental development as their foundation [16]. The main process phases have remained same. However, instead of progressing from phase to phase, during any particular iteration a usable part of the software is provided to users so that the designers can include any feedback in the next cycle. The learning occurs both during the development and during the use of the system. In this way a system evolves toward full implementation.

A further advance in software design methods is the Model-Driven Architecture (MDA) [17] concept. The basic notion of MDA is to separate the design process from the definition of the software architecture and implementation technology. User requirements are defined using a computation independent model (CIM). CIM describes the problem and the proposed solution without reference to the software technology that will be used. CIM serves as the foundation for the platform independent model (PIM) concept. PIM describes the software model but does not specify a concrete software platform. In the next phase the platform specific model (PSM) is defined, serving as a basis for the implementation model that guides the actual coding.

The most recent design methodology, referred to as agile software development [3], derives from the iterative model. It represents a family of methods that define a conceptual framework for software development. The methods are organized around a central set of basic principles such as communication, simplicity, feedback, courage, and respect. Communication and respect imply that communication, participation, respect, and trust must exist among stakeholders in the software development process. These principles play a crucial role during the user requirements definition phase. For the sake of simplicity the communication among participants must be effortless, comprehensible, and based on models that all team members understand. The feedback is based on the use of models and visual representations that enable prompt comments on particular ideas and their rapid adaptation to real needs. Courage involves the ability of the development team members to make prompt decisions. They need to have sufficient courage to completely change the development direction or abandon particular ideas if they lead to unacceptable results.

This paper describes the decisions and actions that were taken during the system redesign as a linear sequence. However, the actual process was far from linear. It was marked by many decision loops. Frequently, final assessments forced us to reconsider our initial assumptions. In one case, a daring decision was required to terminate an endless loop. As a method, the redesign process was more similar to a technique referred to as “cooperative gaming of invention and communication” [18] then any of the design methods described above.

We had three objectives as a goal: a) to deliver software, b) to create software amenable to change, and c) to create an adaptable redesign methodology. We spent two thirds of the allocated time to achieve the second and third goal, and as a result managed to achieve the first goal in one third of the allocated time. A crucial success factor was the improved understanding among project members that came as the result of the technology selection process. In addition, daring decisions were made each time a system architect recognized that the decision loop was entering a third cycle. In that case, a technology that appeared most promising was selected, but the decision was not forced onto the other team members. Instead, the entire decision cycle was described to all of the team members, and the reasons for the selection of the particular technology were presented. At that point, suggestions from other team members were accepted only if they proposed arguments that would result in the immediate acceptance of another technology (i.e., arguments yielding to a new decision cycle were rejected).

Communication played a major role in the redesign process. In the development of the WP 1.0 system, both programmers and welding specialists forced to express their viewpoints. As a consequence, the system grew too large and too complex. The system architect assumed a role in this project that architects traditionally have in building design – the role of the mediator between interacting parties. Having had training and experience as both an engineer and software developer, he understood that engineers and programmers often use the same words to represent different concepts. In that way, he largely increased the speed with which information moved between the welding specialists and the programmers. He also used his understanding of both disciplines to effectively emphasize by analogy to building design and construction that the WP 2.0 system should serve people instead of forcing users to become servants of the technology, even if that requires a large effort on the part of the engineers and programmers. In the communication process, no deadlocks were allowed: if one approach to a team member did not succeed, then another approach was devised until a shared understanding among the team members was achieved.

9. Conclusion

In retrospect we can say that we succeeded in producing usable software and preparing for the inevitable future software changes in the available period. The software is now in the deployment phase in two companies in Serbia, and under consideration for use in a few companies in Romania. The main problem that is still delaying widespread use is the need to maintain both the manual and the computer-based certification management processes during the deployment phase.

We have no evidence that our preparation for future software changes has been effective. Only case, so far, requiring change was the development of the system's demonstration version. Since it is quite difficult to explain to probable users how to install the SQL server, web server, and all of the Java libraries and code, we decided that the creation of a working USB demonstration version was an appropriate solution. Since the development was conducted in the Microsoft Windows environment, the live USB implementation required the WP 2.0 system to work under the Linux OS. The whole implementation took one day to complete, providing us with mild optimism that our system will be able to adapt to future changes.

10. Acknowledgments

The authors wish to express their appreciation to all members of the E!2774 project, who provided many results that were used during the design and development of the WP 2.0 system. Professor Stojan Sedmak was a constant driving force behind the development of both the WP 1.0 and WP 2.0 systems. Horia Dascau, Zlatan Milutinović, and Aleksandar Živković provided many insightful comments on the actual experience of using the system.

References

1. Zhang, J., Lin, Y., Gray, J.: Generic and Domain-Specific Model Refactoring using a Model Transformation Engine. In Beydeda, S., Book, M., Gruhn, V., (eds.): Model-driven Software Development - Research and Practice in Software Engineering, Springer, Berlin Heidelberg. 199-217 (2005)
2. Opdyke, W. F.: Refactoring: A Program Restructuring Aid in Designing Object-Oriented Application Frameworks. PhD Thesis, University of Illinois at Urbana-Champaign, USA. (1992)
3. Cockburn, A.,: Agile Software Development. Addison Wesley, Boston. (2000)
4. Radović, N., Radaković, Z., Đurović, A., Sedmak, S., Jandrlić, A., Golubović, D., Zrilić, M., Prokić-Cvetković, R., Popović, O., Milović, Lj., Rakin, M., Engh, E.: Welders passport-program structure and application. In Proceedings of the 1st South-East European Welding Congress - Welding and joining technologies for a sustainable development and environment, Timisoara, Romania, 260-263, (2006)

5. Sedmak S., Svetel I., Đurović A., Kovačević T.: Welders passport information system. Zavarivanje i zavarene konstrukcije, vol. 53, no. 3, 117-123, (2008) (in Serbian)
6. Quintino L., Ferraz R., Fernandes I.: The EWF Integrated Manufacturer Certification System. In Proceedings of the 1st South-East European Welding Congress - Welding and joining technologies for a sustainable development and environment, Timisoara, Romania, 207-213, (2006)
7. McConnell, S.: Code Complete, Second Edition. Microsoft Press, Redmond. (2004)
8. Highsmith, J.: Adaptive Software Development. Dorset House, New York. (2000)
9. http://www.visualsvn.com/server/
10. http://www.eclipse.org/subversive/
11. https://glassfish.dev.java.net/javaee5/persistence/
12. http://www.icefaces.org/main/home/
13. http://www.jboss.org/jbossrichfaces/
14. https://glassfish.dev.java.net/
15. Royce, W.W.: Managing the development of large software systems: concepts and techniques. Proceedings of the 9th international conference on Software Engineering, Monterey, 328- 338, (1987)
16. Larman, C., Basili V.R.: Iterative and Incremental Development: A Brief History. IEEE Computer, vol. 36, no. 6, 47-56, (2003)
17. Mellor, S.J., et al.: MDA Distilled: Principles of Model-Driven Architecture. Addison Wesley, Boston, (2004)
18. Cockburn, A.: The End of Software Engineering and the Start of Economic-Cooperative Gaming. Computer Science and Information Systems, vol. 1, no. 1, 1 -32, (2004)

Igor Svetel graduated at the Faculty of Architecture, University of Belgrade in 1986, and completed his Ph.D. at the same school in 2003. He is research associate at Innovation center, Faculty of mechanical engineering in Belgrade. His research interests are related to the design methodology applied both to the fields of software development and AEC, and application of ICT in architecture.

Aleksandar Đurović is the programmer specialized in the development of MVC web applications using J2EE, Oracle, and Open Source technologies. His interests include Oracle and MySQL databases and object-relational mapping in Oracle ADF, EJB3, and Hibernate using JSF, Spring, Struts, and Struts2 on Oracle AS, Glassfish, JBoss, and Tomcat application servers. He is also developing mods for internet forums developed in PHP.

Vencislav Grabulov graduated at the Faculty of Technology and Metallurgy, University of Belgrade in 1975, and completed his Ph.D. at the same school in 1995. He served as the President of the Serbian Welding Society and Chief executive of Authorized National Body for education of welding personal. He is now general manager of the Institute for Testing of Materials in Belgrade. His research interests are related to the: research and development of high strength steels and Al alloys, materials testing, fracture mechanics testing of metallic materials and composites, welding and fabrication by welding, behavior of materials during welding, welding metallurgy and weldability, and nondestructive testing methods.

Received: April 22, 2009; Accepted: January 19, 2010.

UDC 004.413.4, DOI:10.2298/CSIS090923008L

Software Testing Optimization by Advanced Quantitative Defect Management

Ljubomir Lazić

State University of Novi Pazar, Vuka Karadzica bb,
36 300 Novi Pazar, Serbia
llazic@np.ac.rs

Abstract. Software testing provides a means to reduce errors, cut maintenance and overall software costs. Numerous software development and testing methodologies, tools, and techniques have emerged over the last few decades promising to enhance software quality. While it can be argued that there has been some improvement it is apparent that many of the techniques and tools are isolated to a specific lifecycle phase or functional area. This paper presents a set of best practice models and techniques integrated in optimized and quantitatively managed software testing process (OptimalSQM), expanding testing throughout the SDLC. Further, we explained how can Quantitative Defect Management Model be enhanced to be practically useful for determining which activities need to be addressed to improve the degree of early and cost-effective software fault detection with assured confidence is proposed. To enable software designers to achieve a higher quality for their design, a better insight into quality predictions for their design choices, test plans improvement using Simulated Defect Removal Cost Savings model is offered in this paper.

Keywords: Software Testing; Defect Management; Optimization.

1. Introduction

This paper presents some research results of ongoing project [5-7][1], designed to study software defect data as a means toward identifying where resources should be allocated most effectively to provide the highest quality of software product while reducing the overall cost of software testing. The software development industry spends more than half of its budget on maintenance related activities. Software testing provides a means to reduce errors, cut maintenance and overall software costs. The importance of software testing has been emphasized more and more, as the quality of software affects its

[1] This work has been done within the project 'Integrated and Optimized Software Testing and Maintenance Process', supported in part by the Ministry of Science and Technological Development of the Republic of Serbia under Project No. TR-13018.

benefit to companies significantly [1-4]. The identification and removal of software defects constitutes the basis of the software testing process a fact which inevitably places increased emphasis on defect related software measurements. Early in the history of software development, testing was confined to testing the finished code, but, testing is more of a quality control mechanism. Avoidable rework consumes a large part of development projects, i.e. 20-80 percent depending on the maturity of the organization and the complexity of the products [9]. High amounts of avoidable rework commonly occur when having many faults left to correct in late stages of a project. In fact, research studies indicate that the cost of rework could be decreased by up to 50 percent by finding more faults earlier [2, 5, 9]. Numerous software development and testing methodologies, tools, and techniques have emerged over the last few decades promising to enhance software quality. While it can be argued that there has been some improvement it is apparent that many of the techniques and tools are isolated to a specific lifecycle phase or functional area.

This paper focuses on software testing and the measurements which allow for the quantitative evaluation of this critical software development process. The software testing process requires practical measurements for the quantification of all software testing phases. Software product quality and software testing process (STP) improvement commence with addressing the testing process in a quantitative manner [7]. The continuous monitoring of the testing process allows for establishing an adequate level of confidence for the release of software products and for the quantification of software risks, elements which traditionally have plagued the software industry. The mechanism for this study is development of a series of simulation models to improve STP [7,8].

In this paper, Quality and Efficiency in Software Testing by our OptimalSQM framework is described and its components defined and exemplified. It also discusses practical applications of OptimalSQM and research model for investigating its antecedents and impacts is presented. OptimalSQM provide alignment between testing and development which has been raised as an issue for successful systems development. Missing however have been actionable how to methodologies for assessing and enhancing such alignment [12,13,16]. This paper attempts to fill this gap by describing a systematic methodology, a development-testing alignment (DTA) methodology which posits that such alignment leads to beneficial effects such as lower costs and shorter time of development, greater system quality, fewer errors and a better relationship between the corporate IT units.

Systematic mechanisms for tying testing strategy and capabilities to development strategies and capabilities are also discussed. This paper presents a set of best practice models and techniques integrated in optimized and quantitatively managed software testing process, expanding testing throughout the SDLC. It includes best practice from Design of Experiments, Modeling & Simulation, integrated practical software measurement, Six Sigma strategy, Earned (Economic) Value Management (EVM) and Risk Management (RM) methodology through simulation-based software testing scenarios at various abstraction levels of the SUT to manage stable (predictable and controllable) software testing process at lowest risk, at an affordable price and time.

To enable software designers to achieve a higher quality for their design, a better insight into quality predictions for their design choices, test plans improvement using Simulated Defect Removal Cost Savings model is offered in this paper. Much rather we aim to define a simulation method with which it is possible to assist the test manager in evaluating test plan alternatives and adjusting test process improvement decisions in a systematic manner.

2. Need for Research

Cost to an organization (both in dollars and in image) is significant when software defects are identified after installation at a client site. This project intends to identify areas where improvements in software testing resource allocations would provide added value to the organization.

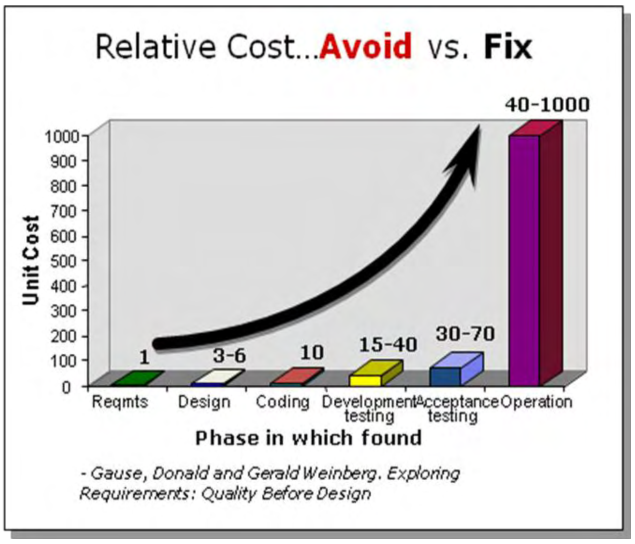


Fig. 1. Average Cost Of Defect Removal [10]

This paper proposes a development-testing alignment (DTA) methodology which posits that such alignment leads to beneficial effects such as lower costs and shorter time of development, greater system quality, fewer errors and a better relationship between the corporate IT units and customers in business functions who have commissioned new systems (see Fig. 2). Alignment models and measurements have been studied in other related contexts [16] but never within corporate IT units and specifically between the development and testing functions. The paper therefore decomposes DT alignment into a series of aspects for the purpose of assessing and analyzing each of the construct. These aspects are drawn from the overarching framework developed initially from prior literature [8,16]. The DTA methodology will allow IT managers to improve the effectiveness of testing and development by both synergistically integrating testing in the development process and by aligning the testing and development units in terms of strategy and execution capability.

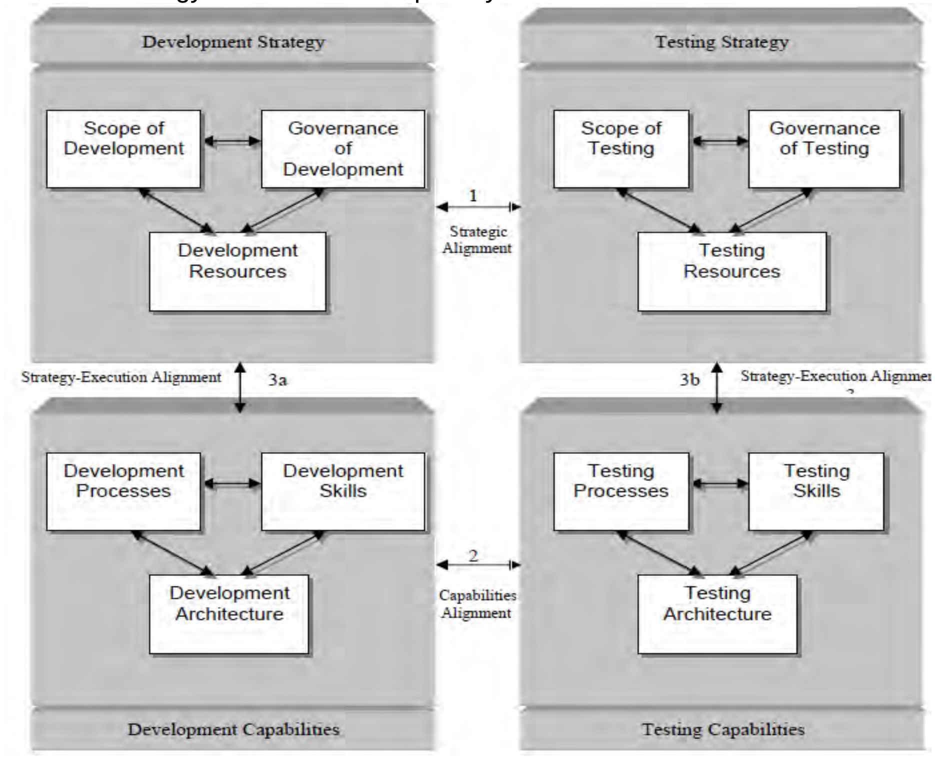


Fig. 2. Alignment model for testing and development (adapted from [13])

Organizations that engage in software development and testing benefit significantly if their management team has tools to assist them in determining the most effective use of financial resources that might result in the fewest software errors in delivered systems [2-10,20-24]. To be most effective, this tool needs to be developed after a thorough review of the specific organization’s testing data [17,24]. Once developed, the tool will identify the specific phases and processes during the development life cycle where additional resources would provide the best return on investment and highest software quality. The use of this tool will provide a major reduction in the number and severity of software defects that exist after software testing. It will also reduce the overall cost of software testing by focusing on the appropriate process for a specific organizational environment [7,9,17-19,24].

To summarize, the purpose of this research is to increase software quality and reduce overall costs of software testing by focusing resources where they provide the most value. According to Gartner [14], on average, 7% of software functionality that was paid for is actually used, while 85% of IT projects failing to meet objectives (32% being cancelled outright). Dhaliwal and Onita [13] posit that many of these development failures are a result of poorly executed development process. These employ either inadequate development models or flawed implementation due, in part, to the lack of proper testing and effective collaborative mechanisms between testers and developers. These issues have not yet found a proper solution, due, in part, to a lack of a methodology that would allow the analysis and correction of software development processes. A review of the testing and development literature reveals that relations between the development and testing functions are lacking for projects of medium and large magnitude, where testing is separate from the development activities [12,15].

2.1. Research Questions

Based on the outcome of the evaluation of related work conducted in the previous subsection, the our project has identified some challenges to address. The challenges can be broken down into five sub-questions to address in this paper. The initial main research question that was posed for the complete research in this project was:

RQ1 or Main Research Question: How can software testing be performed efficiently and effectively i.e. Optimal, that is, do we have a framework model targeted specific software testing domains or problem classes described below in RQ2 to RQ5?

To be able to address the main research question several other research questions needed to be answered first (RQ2–RQ5). The first question that needed an answer, after the main research question was formulated, was:

RQ2: Which metric or set of metrics can assess effectiveness of test detecting techniques and what is the potential in combining different software testing techniques with respect to effectiveness (and to some extent efficiency)?

The answer to this research question is to be found in Section 3 and 4 together with an analysis of how software testing is used in different types of projects. To begin with, the research aimed at exploring the factor of defect detection and removing effectiveness (DRE) during SDLC while later focusing on early aspects of software cost of quality. In order to examine if the current practice in software development projects was satisfactory for developing software with sufficient quality and budget constraint, RQ3 evolved into:

RQ3: Which metric or set of metrics can identify and prioritize software quality attributes, can assess cost of software quality management process in a specific project i.e. how to optimize software quality to pay off investment in STP improvement (ROI)?

To enable software designers to achieve a higher quality for their design, a better insight into quality predictions for their design choices that evolved to the RQ4:

RQ4: Which metric or set of metrics can identify and prioritize improvements to achieve early and cost-effective software fault detection, can assess the improvements potential of improving the degree of early and cost-effective software fault detection?

A systematic model which enables to minimize the cost of switching between test plan alternatives, when the current choice cannot fulfill the quality constraints, is proposed in this paper as one answer to the RQ4. This is main concern and contribution of this paper which is described in Section 5. New kinds of STP improvement is introduced and hence indirectly led to Research Question 5:

RQ5: How should a software development organization apply the metric(s) suggested above for assessing ongoing and finished projects with an Dynamic Control Model view i.e. What are optimality and stability criteria of very complex STP dynamics problem control?

Section 5 explain how can Quantitative Defect Management (QDM) Model be enhanced (as one solution to RQ4) to be practically useful for determining which activities need to be addressed to improve the degree of early and cost-effective software fault detection with assured confidence, optimality and stability criteria of very complex STP dynamics problem control which is partially solved in our paper [19].

To be able to implement all proposed solutions, one must choose a research methodology. Iterative approaches for improvement exist in the quality management area. The PDCA (plan-do-check-act) or "Shewhart Cycle", the WV (or zigzag) framework and the DMAIC (define-measure-analyze-improve-control) cycles are analogous methods to capture a generic framework for the improvement of a process or system [1,3,8]. A similar model, the "simulate-test-evaluate process" iterative experimentation cycle was developed by the office of the US Secretary of Defense, called the Simulation, Test and Evaluation Process (DoD STEP framework) to integrate M&S into the test and evaluation process of the system/software under test (SUT) [17]. Long design iteration loops with late feedback drive cost and schedule overruns in SDP-STP requires further research of this stability criteria of very complex STP dynamics problem control.

3. Integrated, quantitatively managed and optimized software testing process - OptimalSQM solution

When design and testing activities are not coupled, the information testing provides on product design is delivered at a wrong point in the process. This late information is either not useful any more or shows design problems too late, causing undesired late rework. Thus, iteration cycles should be kept short and rapid.

To address the research questions stated above, multiple studies have been conducted [5-8] about alignment between the development and testing functions which can be defined as the strategic and operational fit between the development and testing functions on components of strategy and capabilities [13-16]. Since systems development as well as systems testing is integral parts of the corporate technology acquisition strategy, they too have to be aligned to ensure business success. In many organizations, there is a gap, or misalignment, at the strategic and/or execution level, between the development and testing groups as well as between individual testers and developers. To correct these misalignments, this paper proposes a methodology, grouped under the DTA model [13] that draws upon the strategic alignment model initially proposed in [16]. This DTA model focuses on the fit between the development and testing functions. A high level of integration of business and IT plans facilitates communication and collaboration [16]. Integration represents the level of linkage between development and testing, while correspondence represents how closely their capabilities mirror and complement each other. Varying levels of alignment can either promote or hinder integration and correspondence. This is a common characteristic of all alignment models in the literature as verified by Dhaliwal, J. and Onita C. in their work [13]. Figure 2 details the key structural and flow components of the DT alignment model for development and testing within the corporate IT unit. This model decomposes the alignment of the development and testing functions along three key flow dimensions: 1) strategic alignment, 2) capabilities alignment, and 3) strategy-execution alignment.

Based on our study in [8], a methodology for achieving DT Alignment through Collaborative Techniques & Technology, enables OptimalSQM to be realised. The methodology is derived from a survey of the literature from Strategic Alignment [13-16] Testing [1-4], [8-10] to Project Management and Information Systems development methods [10-16]. To improve the reliability and validity of this methodology, alignment case studies and field studies were conducted and real life examples are given to improve the applicability of the methodology. A list of techniques is also mapped onto each step of the methodology.

3.1. Integrated and Optimized Software Testing Process (IOSTP) framework - OptimalSQM solution

To answer the main research question (**RQ1**) we applied DTA model, described above, in OptimalSQM framework which combine best practice from Design of Experiments, Modeling & Simulation, integrated practical software measurement, Six Sigma strategy, Earned (Economic) Value Management (EVM) and Risk Management (RM) methodology through simulation-based software testing scenarios at various abstraction levels of the software under test (SUT) to manage stable (predictable and controllable) software testing process at lowest risk, at an affordable price and time [8,9], [17,18] as depicted in Fig. 3. Unlike conventional approaches to software testing (e.g. structural and functional testing) which are applied to the software under test without an explicit optimization goal, the IOSTP with embedded Risk Based Optimized STP (RBOSTP) approach designs an optimal testing strategy to achieve an explicit optimization goal, given a priori [8,17]. This leads to an adaptive software testing strategy. A non-adaptive software testing strategy specifies what test suite or what next test case should be generated, e.g. random testing methods, whereas an adaptive software testing strategy specifies what testing policy should be employed next and thus, in turn, what test suite or test case should be generated next in accordance with the new testing policy to maximize test activity efficacy and efficiency subject to time-schedule and budget constraints.

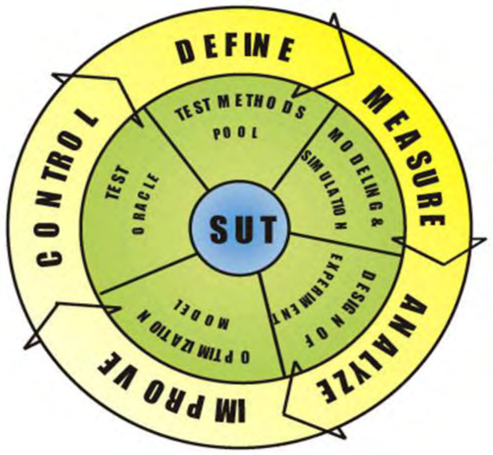


Fig. 3. Integrated and optimized software testing process (IOSTP) framework, core of OptimalSQM framework [17]

The use of state-of-the-art methods and tools for planning, information, management, design, cost trade-off analysis, and modeling and simulation, Six Sigma strategy significantly improves STP effectiveness as in Fig. 3 which graphically illustrates a generic IOSTP framework that makes core of the OptimalSQM framework [17].

The main components of IOSTP with embedded RBOSTP approach to STP:

- Integrate testing into the entire development process
- Implement test planning early in the life cycle via Simulation based assessment of test scenarios
- Automate testing, where practical to increase testing efficiency
- Measure and manage testing process to maximize risk reduction
- Exploit Design of Experiments techniques (optimized design plans, Orthogonal Arrays etc.)
- Apply Modeling and Simulation combined with Prototyping
- Continually improve testing process by pro-active, preventive (failure mode analysis) Six Sigma DMAIC model
- Continually monitor Cost-Performance Trade-Offs (Risk-based Optimization model, Economic Value and ROI driven STP).

In order to significantly improve software testing efficiency and effectiveness for the detection and removal of requirements and design defects in our framework of IOSTP, during 3 years of the IOSTP framework deployment to STP of embedded-software critical system such as Automated Target Tracking Radar System (ATTRS) [17], we calculated overall value returned on each dollar invested i.e. ROI of 100:1 .

4. Optimum DDTs combination selection and optimization study in OptimalSQM

The research question - **RQ2** divided the research, as presented in this paper, into two areas covering effectiveness in software testing techniques (defect detection techniques – DDT) and efficiency in software testing with development-testing alignment (DTA) methodology [5-9]. Such alignment leads to beneficial effects such as lower costs and shorter time of development, greater system quality, fewer errors and a better relationship between the corporate IT unit and customers in business functions who have commissioned new systems. To begin with **RQ2**, the research aimed at exploring the factor of defect detection and removing effectiveness (DRE) during SDLC is answered in our work [6]. Here is brief description of main ideas of **RQ2** answer.

4.1. Statement Of the Problem - Defect removal effectiveness

A key metric for measuring and benchmarking the software testing efficacy is by measuring the percentage of possible defects removed from the product at any point in time. Both a project and process metric – can measure effectiveness of quality activities or the quality of a all over project by:

$$DRE = E/(E+D) \qquad (\mathbf{1})$$

Where *E* is the number of errors found before delivery to the end user, and *D* is the number of errors found after delivery. The goal is to have *DRE* close to 100%. The same approach is applied to every test phase denoted wit *i* as shown on Fig. 4:

$$DRE_i = E_i \,/\, (\, E_i + E_{i+1}) \qquad (\mathbf{2})$$

Where E_i is the number of errors found in a software engineering activity *i*, and E_{i+1} is the number of errors that were traceable to errors that were not discovered in software engineering activity *i*. The goal is to have this DRE_i approach to 100% as well i.e., errors are filtered out before they reach the next activity. Projects that use the same team and the same development processes can reasonably expect that the *DRE* from one project to the next are similar. For example, if on the previous project, you removed 80% of the possible requirements defects using inspections, then you can expect to remove ~80% on the next project. Or if you know that your historical data shows that you typically remove 90% before shipment, and for this project, you've used the same process, met the same kind of release criteria, and have found 400 defects so far, then there probably are ~50 defects that you will find after you release. How to combine Defect Detection Technique to achieve high *DRE*, let say >85%, as a threshold for IOSTP required effectiveness [2-5], is explained in this section, which describe optimum combination of software defect detection techniques choices.

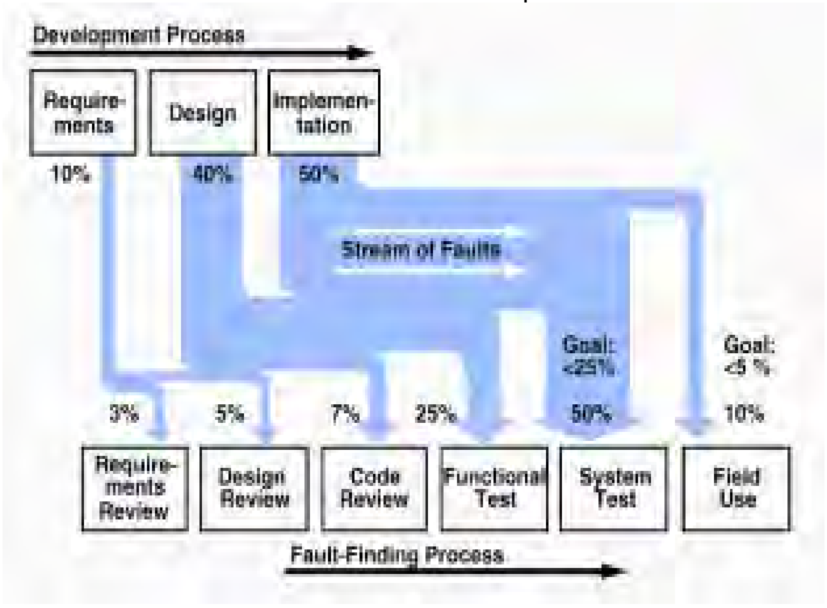


Fig. 4. Fault Injection and Fixing Model

Note that the defects discussed in this section include all severity levels, ranging from severity 1: *activity stoppers*, down to severity 4. Obviously, it is important to measure defect severity levels as well as recording numbers of defects.

4.2. The optimum combination of software defect detection techniques choices determination

Planning, managing, executing, and documenting testing as a key process activity during all stages of development is an incredibly difficult process. There is strong demand for software testing effectiveness and efficiency increases. Software/System testing effectiveness is mainly measured by percentage of defect detection and defect leakage (containment), i.e. late defect discovery. Software testing efficiency is mainly measured by dollars spent per defect found and hours spent per defect found. The first step of test strategy definition is to decide what to test in the SDP process as described in [3,20,23]. The requirements verification matrix method supports this goal by cross-referencing each product requirement with suitable verification methods (inspection, analysis, demonstration, or test), verification classes (design proof, first article, or production), and special verification procedures (e.g., Failure Mode and Effect Analysis, Design of Experiments, Finite Element Method, etc.). Product requirements with high priority are the critical requirements, where test planning has to concentrate its resources.

The process is a building block approach designed to build upon the strengths and minimize the weakness of each testing technique and available resources. Main task was to develop a versatile optimization model [8] for assessing the cost, duration and effectiveness of alternative test scenario through feasible series of experiments: software test method, field test, through simulation, or through a combination, which represent sequence of test events.

Such scenarios are invaluable for determining where testing resources should be spent at the beginning of software development project. With an optimized testing solution, you can create what-if scenarios to help users understand the impact of changing risks, cycle attributes and requirements as priorities change. This insight proves invaluable when a testing organization is trying to determine the best way to balance quality with cost and schedule. By understanding the impact of different factors on testing, IT managers can identify the right balance.

We applied the End-to-End (E2E) Test strategy in our IOSTP framework [6,17]. End-to-End Architecture Testing is essentially a "gray box" approach to testing - a combination of the strengths of white box and black box testing. In determining the best source of data to support analyses, IOSTP with embedded RBOSTP considers credibility and cost of each test scenario i.e. concept. Resources for simulations and software test events are weighed against desired confidence levels and the limitations of both the resources and the analysis methods.

Our study [6] focuses on rapid multidisciplinary analysis and evaluation-on-a-DRE maximum-basis for DDT combination choices selection for each test phase activities in an traditional SDP i.e. P1- software requirement, P2- High level design, P3- Low Level Design, P4- code under test, P5- integration test, P6- system under test and finally P7- Acceptance test. Recall section 4, in our work [6], for Different Defect Detection Strategy and Techniques options, together with critical STP variables performance characteristics (e.g. DRE, cost, duration), in which we are studied to optimize design, development, test and evaluation cost using orthogonal arrays for computer experiments. The optimum combination of software defect detection techniques choices were determined applying orthogonal arrays constructed for post mortem designed experiment with collected defect data of a real project. First, we applied adapted Borda voting method, on similar way, to rank all used Defect Detection Techniques through software development life cycle from most-to-least performance and quality characteristics of DDT in revealing software faults (bugs, errors). In this way we reduced huge possible number of DDTs, in particular, the DDT with the highest Borda Count is the best DDT according to testers Performance and Quality multi-criteria assessment [6], the DDT with the second highest count is the next DDT with highest score, and so forth to only three most ranked DDT. According to testers assessment of 5 most frequently used DDT in IOSTP [6,]: DDT1= Inspection – DBR, DDT2= PBR, DDT3= CEG+BOR+MI, DDT4= M&S, DDT5= Hybrid (Category Partition, Boundary value analysis, Path testing etc.) three of DDTs have the highest rank 0 i.e. DDT1=DDT2=DDT4=0, then DDT3= CEG+BOR+MI is next ranked and the last was DDT5. Because of that we will group those three DDT with highest rank 0, call them Static Test Techniques – TT1 and treat all three DDTs as one factor in optimization experiment applying Orthogonal Arrays as Optimization Strategy. Next high Borda ranked DDT4= CEG+BOR+MI we designate with TT2 and the last ranked DDT5 as TT3.

In this study, design of maximum DRE percentage of STP optimization problem solving with best DDT choice combination in each phase P1 to P7 as controlled variables values is determined by designed experiment plan using orthogonal arrays designed for this computer experiment. Seven major test phases - P1 to P7, for accounting maximum DRE percentage all over STP fault injection and removal model (see Fig. 4, 7 and 8 below) for DDT candidate selection in each test phase were determined. These were the Static Test Techniques – TT1 (consisting of three DDTs as one factor in optimization experiment applying Orthogonal Arrays as Optimization Strategy), the TT2 i.e. DDT4= CEG+BOR+MI and TT3 – Hybrid Detection Technique= DDT5 (consisting of Category Partition, Boundary value analysis, Path testing etc.). The objective of this investigation was then to determine the best combination of Test Techniques (*TTi* , *i*=1,2 and 3) options for the seven major test phase activities sections optimized for maximum DRE percentage under cost and time constraints [6,pages 1333-1335].

As the next step, least squares regression analysis is used to fit the second order approximation model given by equation (3) to the DRE data in terms of the seven design variables P_i , i=1 to 7. This parametric model accounts for the response surface curvature (square terms) and two factor interactions (cross terms):

$$DRE (\%) = 111.71 - 2.58 *P1 + 1.22*P2 -1.95*P3 - 7.61*P4 - 0.69*P5 + 0.94*P6 -13.04*P7 - 0.36*P2^2 + 1.46*P4^2 + 0.79*P5^2 - 0.36P6^2 + 3.15*P7^2 \quad (\mathbf{3})$$

Note that, in this response surface approximation model, the parameter values for P_i design variables are restricted to 1 (TT1), or 2 (TT2), or 3 (TT3). In Table 1, a Maximum DRE (%) value and corresponding Test Techniques choices (TT1,TT2 and TT2) per test phase solution is given.

Table 1. Maximum DRE (%) value and corresponding Test Techniques choices per test phase solution [6, page 1335]

P1	P2	P3	P4	P5	P6	P7	DRE [%]
TT1	TT2	TT1	TT1	TT3	TT2	TT2	94.3

At these levels, the IOSTP DRE was predicted to be 94.03 % using a second order prediction model (3). As a next step, a verification analysis was performed. The DRE (%) of an IOSTP calculated from these test techniques choices, according to the post-mortem real project data using optimized DDT choices from Table 1, we computed DRE (%) to be 93.43 % . Difference is 0.6%=94.03%-93.43% that is acceptable to validate our prediction model for DRE (%) in equation (3) for optimal DDT combination choice given in Table 1.

Optimal combination of DDT choices per phase P given in Table 1 made increase of about 6 %, compared to un-optimized DDTs combination per each test phase we used in our real project in which we achieved DRE of 87.43 % in our case study.

5. Advanced Quantitative Defect Management (AQDM) Model

The investment in software quality, particularly in software testing, like any investment has an immediate cost, with an expected net payback. There is where Quality Cost Analysis could be used as effective tool to make them understand the ROI. In our paper [19], we defined techniques to analyze and interpret return on the testing investment (ROTI) values: Financial ROI and Schedule Benefits as one possible answer to **RQ3** based on our studies [5,18,19] i.e. which metric or set of metrics can identify and prioritize software quality attributes, can assess cost of software quality management process in a specific project i.e. how to optimize software quality?

In our work [19] we proposed a model that traces design decisions and the possible alternatives. With this model it is possible to minimize the cost of switching between design alternatives, when the current choice cannot fulfill the quality constraints.

5.1. Faults-Slip-Through (FST) Model

The partial answer to **RQ5** can be found in our work [19], too. In this section we explain how can Quantitative Defect Management (QDM) Model be enhanced (as answer to **RQ4**) to be practically useful for determining which activities need to be addressed to improve the degree of early and cost-effective software fault detection with assured confidence.

The main objective of the case study presented in this section was to investigate how fault statistics could be used for removing unnecessary rework in the software development process. This was achieved through a measure called Faults-Slip-Through (FST) [9, Section 2], i.e. the measure tells which faults that would have been more cost-effective to find in earlier phases.

5.2. The defect containment measure

An error in an activity of development phase P_i (i=1 to N) is made that causes a failure (see Fig. 10-13). The failure leads to a reported anomaly. When the reported anomaly is analyzed, the fault(s) causing the failure is found and corrected. *Rework* is about revising an existing piece of software or related artifact. Therefore, a typical rework activity is to correct reported anomalies. Rework can be divided into two primary types of corrective work [9]:

- *Avoidable rework* is work that would not have been needed if the previous work would have been correct, complete, and consistent. Such rework consists of the effort spent on detecting and fixing software difficulties that could have been discovered earlier or avoided altogether [2,5].
- *Unavoidable rewor*k is work that could not have been avoided because the developers were not aware of or could not foresee the change when developing the software, e.g. changed user requirements or environmental constraints [9].

This section describes the selected method for how to achieve the objectives stated in the previous section. The method can be divided into the following three steps:

1. Determine which faults that should have been avoided or at least found earlier,
2. Determine the average cost of finding faults in different phases,

3. Determine the improvement potential from the results in (1) and (2).

The three sub-sections below describe how to perform each of the three steps.

5.3. The raw defect containment data

This section is dedicated to a model for assessing a plan for SQA defect-removal effectiveness and cost. The model, a multiple filtering model as shown on Fig. 4, is based on data acquired from a survey of defect origins, percentages of defect removal achieved by various quality assurance activities, and the defect-removal costs incurred at the various development phases. The model enables quantitative comparison of quality assurance policies as realized in quality assurance plans. The application of the proposed model is based on three types of data, described under the following headings from [1, pages 135-142].

5.4. Defects removal improvement potential

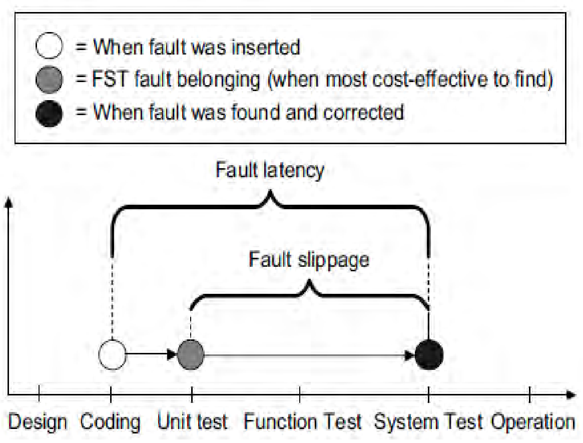

○ = When fault was inserted
● = FST fault belonging (when most cost-effective to find)
● = When fault was found and corrected
Fault latency
Fault slippage
Design
Coding
Unit test
Function Test
System Test
Operation

Fig. 5. Example of Fault Latency and FST

As previously mentioned, FST measurement was used for determining this, i.e. it evaluates whether each fault slipped through the phase where it should have been found or not. The main difference between FST measurement and other related measurements is when a fault is introduced in a certain phase but it is not efficient to find in the same phase. For example, a certain test technique might be required to simulate the behaviour of the function. Then it is not a fault slippage. Figure 5, further, illustrates this difference. A consequence of how FST is measured, a definition must be created to support the measurement, i.e. a definition that specifies which faults that should be found in which phase. To be able to specify this, the organization must first determine what should be tested in which phase. Therefore, this can be seen as test strategy work. Thus, experienced developers, testers and managers should be involved in the creation of the definition. The results of the case study in Section 5.5 further exemplify how to create such a definition.

When having all the faults categorized, the next step is to estimate the cost of finding faults in different phases. From the measure, the improvement potential of different parts of the development process is estimated by calculating the cost of the faults that slipped through the phase where they should have been found (see Fig. 12 and 13 below). The usefulness of the method was demonstrated by applying it on two completed development projects [1] and [2]. The results determined that the implementation phase had the largest improvement potential since it caused the largest FST cost to later phases, i.e. from 56 to 87 percent of the total improvement potential in the two studied project scenarios. It is assumed that the filtering effectiveness of accumulated defects of each quality assurance activity is not less than 40% (i.e., an activity removes at least 40% of the incoming defects). Typical average defect filtering effectiveness rates for the various quality assurance activities, by development phase, based on Galin, D. [1] and 11. Boehm, B. et al [11], are listed in Table 2.

Data collected about development project costs show that the cost of removal of detected defects varies by development phase, while costs rise substantially as the development process proceeds. For example, removal of a design defect detected in the design phase may require an investment of 2.5 working days; removal of the same defect may require 40 working days during the acceptance tests. Estimates of effectiveness of software quality assurance tools and relative costs of defect removal are provided by McConnell [10]. Although defect removal data are quite rare, professionals agree that the proportional costs of defect removal have remained constant since the surveys conducted in the 1970s and 1980s. Instead of average per phase defect removal cost we propose average relative defect-removal costs injected in phase $P_{i\ (i=1\ \text{to}\ 7)}$ and detected and removed latter in downstream phases P_j, $j>i$ up to the operation phase ($j=7$) as shown in Table 3.

The **improvement potential** (**IP**) is determined by calculating the difference between the cost of faults in relation to what the fault cost would have been if none of them would have had slipped through the phase where they were supposed to be found. Figure 6. provides the elements of matrix, with corresponding formulas, for making such a calculation and the improvement potential can be calculated in a two-dimensional matrix. The formulae for calculating the improvement potential for each cell **IPir** is:

$$IPir = No\_faults\_in\_P(i,r) * AverCDR(r,i) - No\_faults\_in\_P(i,r) * AverCDR(i,i)$$

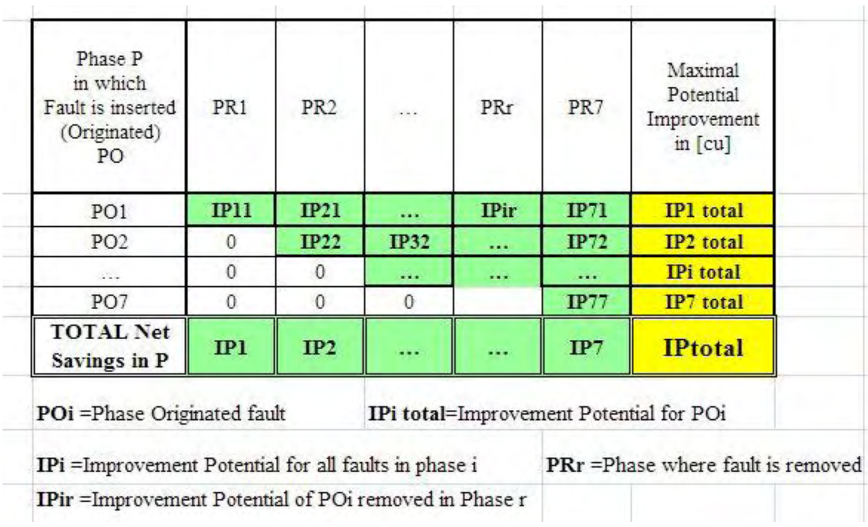
for $i=1$ to 7, and $r \geq i$ correspond to a cell in phase removed/originated matrix. *AverCDR(r,i)* is average cost of faults originated in *i* and removed in *r* as shown in Table 3. **IPi total** and **IPi** are calculated by summarizing the corresponding row/column. In order to demonstrate how to use and interpret the matrix, Figure 12 provides an example calculation by applying the previous formula on the values in the table in the Fig. 10. The most interesting cells are those in the rightmost column that summarizes the total cost of faults in relation to fault belonging and the bottom row that summarizes the total unnecessary cost of faults in relation to phase found. For example, the largest improvement potential is in the **LL Design** test phase, i.e. the phase triggered **30661 [cu]** of unnecessary costs in later phases due to a large FST from faults injected in Requirement phase. Therefore, the primary usage of the values is to serve as input to an expected ROI calculation when prioritizing possible improvement actions according to formula:

ROI= (CostToFixOld - CostToFixNew) /CostToFixOld=39.2%.

5.5. Qunatitative Defect Removal Model

The model is based on the following assumptions:

■ The development process is linear and sequential, following the waterfall model of CMM Level 5. Software size is aproximately 100FP (1 injected defect/FP) i.e. for Java implementation about 50KLOC of source code [4].



Phase P in which Fault is inserted (Originated) PO	PR1	PR2	…	PRr	PR7	Maximal Potential Improvement in [cu]
PO1	**IP11**	**IP21**	**…**	**IPir**	**IP71**	**IP1 total**
PO2	0	**IP22**	**IP32**	**…**	**IP72**	**IP2 total**
…	0	0	**…**	**…**	**…**	**IPi total**
PO7	0	0	0		**IP77**	**IP7 total**
TOTAL Net Savings in P	**IP1**	**IP2**	**…**	**…**	**IP7**	**IPtotal**

POi =Phase Originated fault

IPi total=Improvement Potential for POi

IPi =Improvement Potential for all faults in phase i

PRr =Phase where fault is removed

IPir =Improvement Potential of POi removed in Phase r

Fig. 6. Matrix Formula for Calculation of Improvement Potential

■ A number of “new” defects are introduced in each development phase. For their distributions, see Fig. 8 and 9.

■ Review and test software quality assurance activities serve as filters, removing a percentage of the entering defects and letting the rest pass tothe next development phase. For example, if the number of incoming defects is

30, and the filtering efficiency is 60%, then 18 defects will beremoved, while 12 defects will remain and pass to be detected by the next quality assurance activity. Typical filtering effectiveness rates for the Standard quality assurance activities are shown in Table 2.

Table 2. Average filtering (defect removal - DR) effectiveness by Standard quality assurance activities plan, adapted from [1, pages 136-138]

No.	Quality assurance activity	Defect removal effectiveness	Average Cost of removing a detected defect (cost units)
1	Requirement specification review	50%	1
2	Design review	50%	2.5
3	Unit test – code	50%	6.5
4	Integration test	50%	16
5	Documentation review	50%	16
6	System test	50%	40
7	Operation phase	100%	110

■ At each phase, the incoming defects are the sum of defects not removed by the former quality assurance activity together with the “new” defects introduced (created) in the current development phase.

■ The cost of defect removal is calculated for each quality assurance activity by multiplying the number of defects removed by the relative cost of removing a defect (see Table 3, 3rd column).

■ The remaining defects, unfortunately passed to the customer, will be detected by him or her. In these circumstances, full removal entails the heaviest of defect-removal costs. In this model, each of the quality assurance activities is represented by a filter unit, as shown for Design in Fig. 7. The model presents the following quantities:

■ POD = Phase Originated Defects (from Fig. 8)

■ PD = Passed Defects (from former phase or former quality assurance activity)

■ %FE = % of Filtering Effectiveness (also termed % screening effectiveness) (from Table 2)

■ RD = Removed Defects

■ CDR = Average Cost of Defect Removal (from Table 2)

■ TRC = Total Removal Cost: TRC = RD ×CDR.

The illustration in Fig. 8 of the model applies to a standard quality assurance plan (“standard defects filtering system”) that is composed of six quality assurance activities (six filters), as shown in Table 2.

Table 3. Representative average relative defect-removal costs and fixing multiplier because FST

No.	Quality assurance activity	Average Cost of DR [cost units]	Fixing multi-plier (CM) P1→ P7	Fixing multi-plier (CM) P2→ P7	Fixing multi-plier (CM) P3→ P7	Fixing multi-plier (CM) P4→ P7	Fixing multi-plier (CM) P5→ P7	Fixing multi-plier (CM) P6→ P7
1	Requir. specification review	1	1					
2	Design review	2.5	5	1				
3	Unit test – code	6.5	10	2	1			
4	Integration test	16	50	10	5	1		
5	Documenta-tion review	16	130	26	13	3	1	
6	System test	40	368	64	37	7	3	1
7	Operation phase	110	400	75	40	20	15	10

A comprehensive quality assurance (QA) plan (“comprehensive defects filtering system”) achieves the following:

(1) Adds two quality assurance activities, so that the two are performed in the design phase as well as in the coding phase;

(2) Improves the “filtering” effectiveness of other quality assurance activities. The comprehensive quality assurance plan can be characterized as shown in Table 4.

The main conclusions drawn from the comparison are:

(1) The standard plan successfully removes only 57.6% (28.8 defects out of 50) of the defects originated in the requirements and design phase, compared to 92.0% (46 defects out of 50) for the comprehensive plan, before coding begins.

(2) The comprehensive plan, as a whole, is much more economical than the standard plan as it saves 41% of total resources invested in defect removal, compared to the standard plan.

(3) Compared to the standard plan, the comprehensive plan makes a greater contribution to customer satisfaction by drastically reducing the rate of defects detected during regular operations (from 6.9 % to 3 %).

The comparison also supports the belief that additional investments in quality assurance activities yield substantial savings in defect removal costs. Alternative models dealing with the cumulative effects of several qualityassurance activities are discussed by [2,5,9] as described below. A process-oriented illustration of the comprehensive quality assurance plan and model of the process of removing 100 defects is provided in Fig. 9. A comparison of the outcomes of the standard software quality plan versus the comprehensive plan is revealing as shown in Table 5.

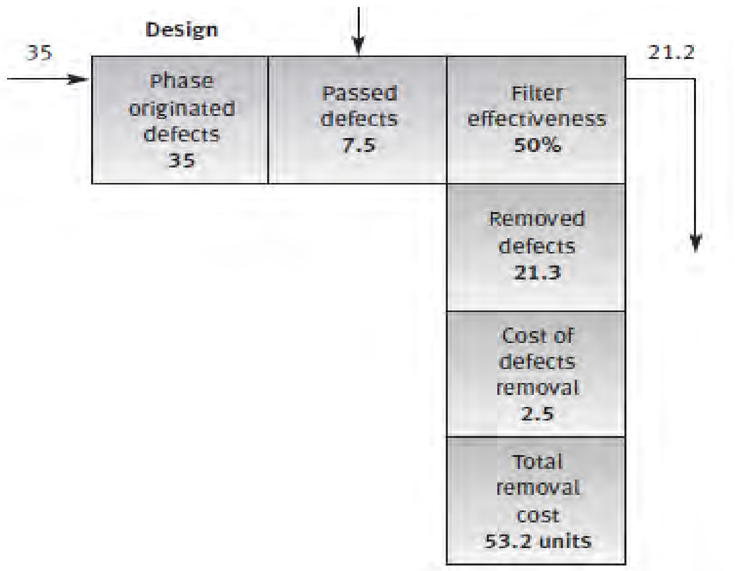

Design
35
Phase originated defects 35
Passed defects 7.5
Filter effectiveness 50%
21.2
Removed defects 21.3
Cost of defects removal 2.5
Total removal cost 53.2 units

Fig. 7. A filter unit for defect-removal effectiveness: example (100 defects) from [1]

Table 4. Average filtering (defect removal) effectiveness by Comprehensive quality assurance activities plan [1, page 140]

No.	Quality assurance activity	Defect removal effectiveness	Cost in [cu] of removing a detected defect
1	Requirement specification review	60%	1
2	Design inspection	70%	2.5
3	Design review	60%	2.5
4	Code inspection	70%	6.5
5	Unit test – code	40%	6.5
6	Integration test	60%	16
7	Documentation review	60%	16
8	System test	60%	40
9	Operation phase	100%	110

Table 5. Comparison of the standard and comprehensive QA plans [1, page 142]

No.	Quality assurance activity	Standad plan		Comprehensive plan	
		Percentage of removed defects	Cost of removing a detected defect (cost units)	Percentage of removed defects	Cost of removing a detected defect (cost units)
1	Requirement specification review	7.5%	7.5	9%	9
2	Design inspection	-	-	28.7%	71.8
3	Design review	21.3%	53.2	7.4%	18.5
4	Code inspection	-	-	24.4%	158.6
5	Unit test – code	25..6%	166.4	4.2%	27.3
6	Integration test	17.8%	284.8	9.8%	156.8
7	Documentation review	13.9%	222.4	9.9%	158.4
8	System test	7.0%	280	4%	160
	Total for internal QA activities	**93.1%**	**1014.3**	**97.4%**	**760.4**
	Defects detected during operation	6.9%	759	2.6%	286
	Total	**100.0%**	**1773.3**	**100.0%**	**1046.4**

5.6. Simulation results of AQDM improvement

Unlike conventional approaches to software testing which are applied to the software under test without an explicit optimization goal, as described above, the OptimalSQM approach designs an optimal testing strategy to achieve an explicit optimization goal, given a priori [5,6]. Improvement of original project data from [2] given in Fig. 10 (Note: original Defect Removal Efficiency [%], shown on Fig. 11 is less then Standard quality assurance activities plan, Scenario 1 in Table 2).

Also, comprehensive quality assurance plan (Scenario 2) which is realised through feasible series of experiments: software test method, field test, through simulation, or through a combination, represent new test sequence determined by Simulated Defect Removal Cost Savings model.

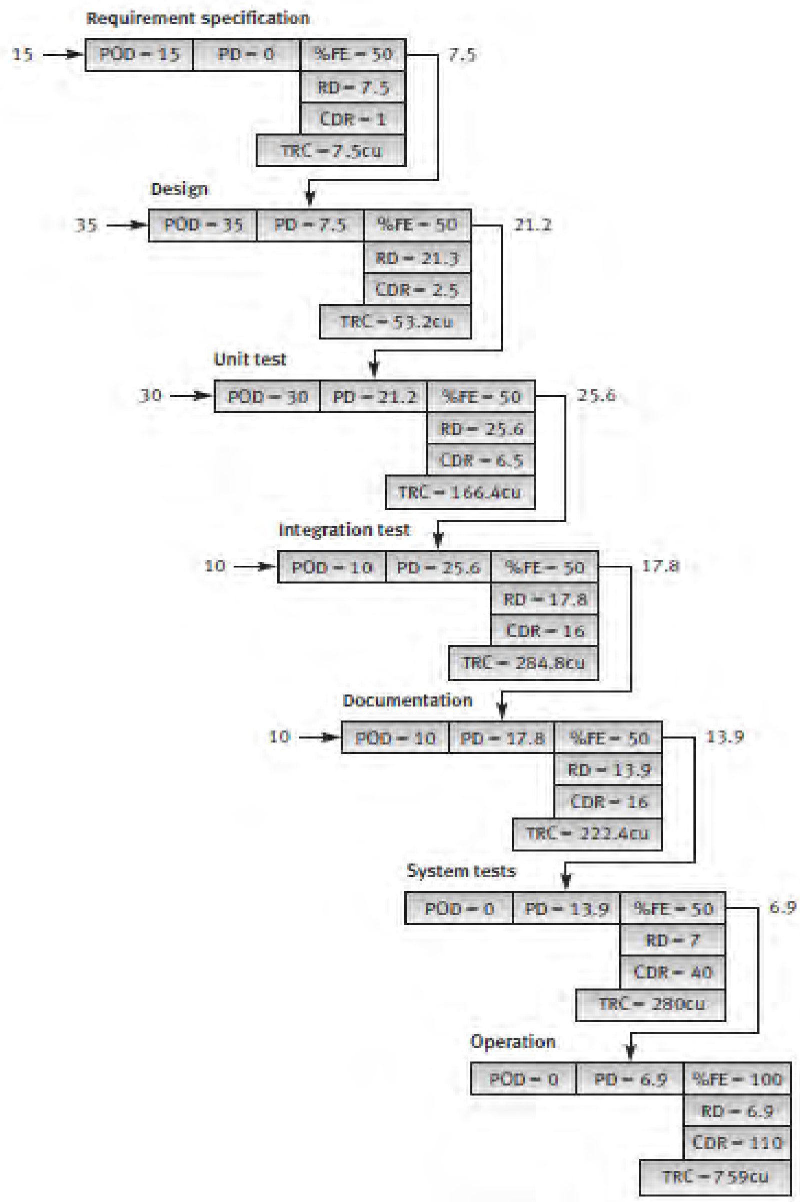
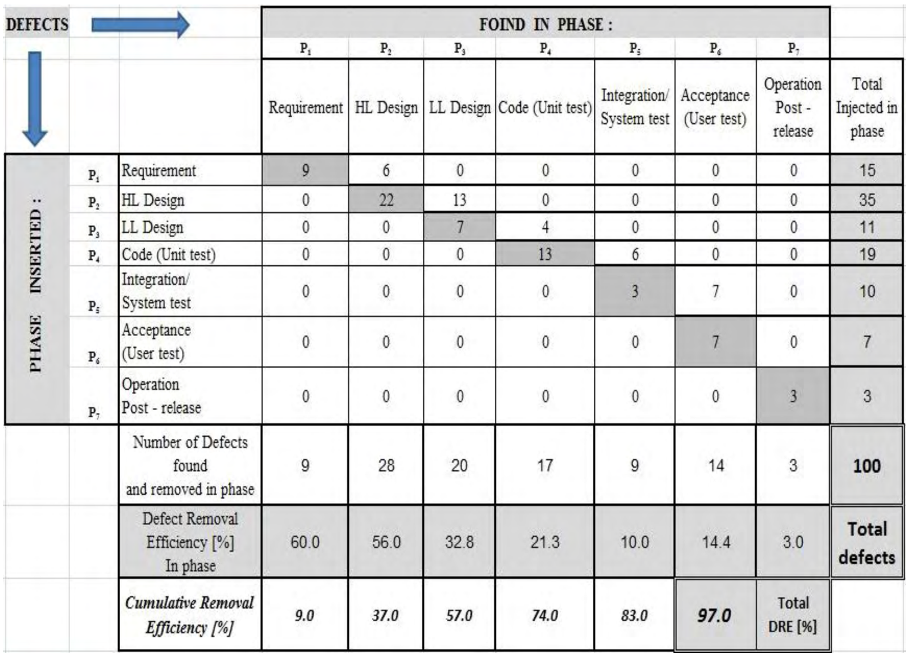


Fig. 8. DRE and costs of Standard QA plan and model of the process of removing 100 defects [1, page 139]



DEFECTS			FOIND IN PHASE :							
			P_1	P_2	P_3	P_4	P_5	P_6	P_7	
			Requirement	HL Design	LL Design	Code (Unit test)	Integration/ System test	Acceptance (User test)	Operation Post - release	Total Injected in phase
PHASE INSERTED :	P_1	Requirement	9	6	0	0	0	0	0	15
	P_2	HL Design	0	22	13	0	0	0	0	35
	P_3	LL Design	0	0	7	4	0	0	0	11
	P_4	Code (Unit test)	0	0	0	13	6	0	0	19
	P_5	Integration/ System test	0	0	0	0	3	7	0	10
	P_6	Acceptance (User test)	0	0	0	0	0	7	0	7
	P_7	Operation Post - release	0	0	0	0	0	0	3	3
		Number of Defects found and removed in phase	9	28	20	17	9	14	3	**100**
		Defect Removal Efficiency [%] In phase	60.0	56.0	32.8	21.3	10.0	14.4	3.0	**Total defects**
		Cumulative Removal Efficiency [%]	***9.0***	***37.0***	***57.0***	***74.0***	***83.0***	***97.0***	Total DRE [%]	

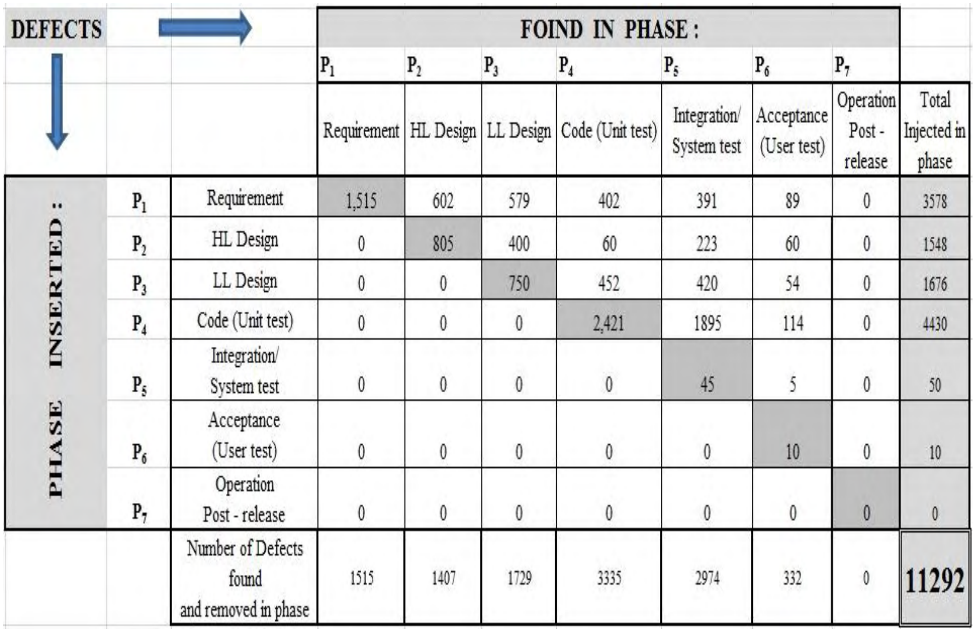
Fig. 9. DRE of Comprehensive QA plan and model of the process of removing 100 defects [1]

The Simulated Defect Removal Cost Savings model, uses net savings approach that is calculated using this formula:

$$NS = IP_{r \to r+1} = FST_{\mathrm{Pr} \to \mathrm{Pr}+1} * (CM_{r \to r+1} - CM_{r \to r}), \quad r=1..6.$$

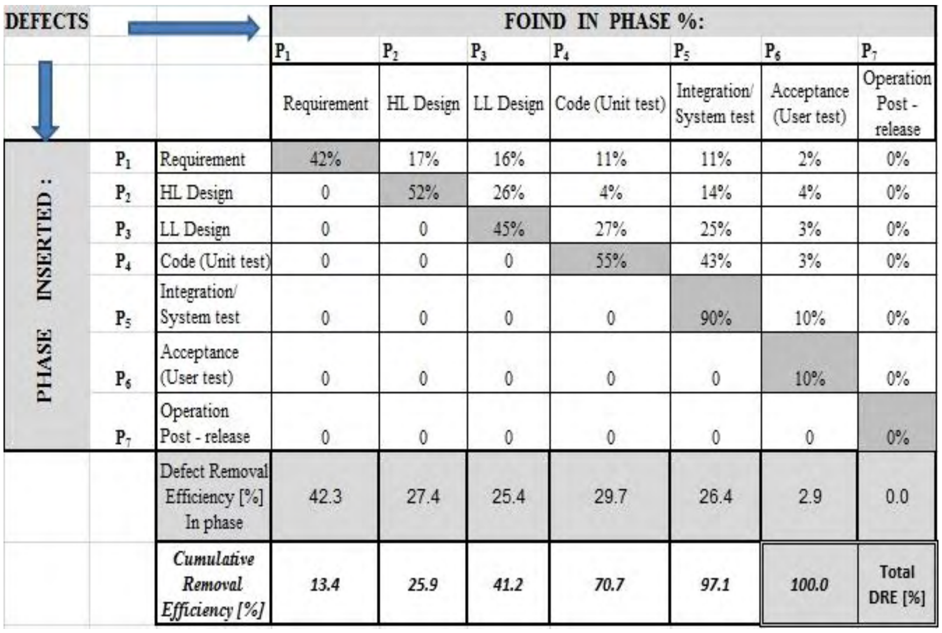
For the given large (~11300 FP, Java implementation about 600KLOC of source code) project example from [2], and for original data of process defect removal effectiveness given in Fig. 11, and simulated calculations of two Scenarios 1 and 2 are shown in Fig. 12, Fig.13 and Fig. 14.

Calculated Matrix of Improvement Potential calculation for Scenario 2 is given in Fig 12. From Fig. 13 and 14, we can easy find **maximal improvement potential point to be in phase P3**.



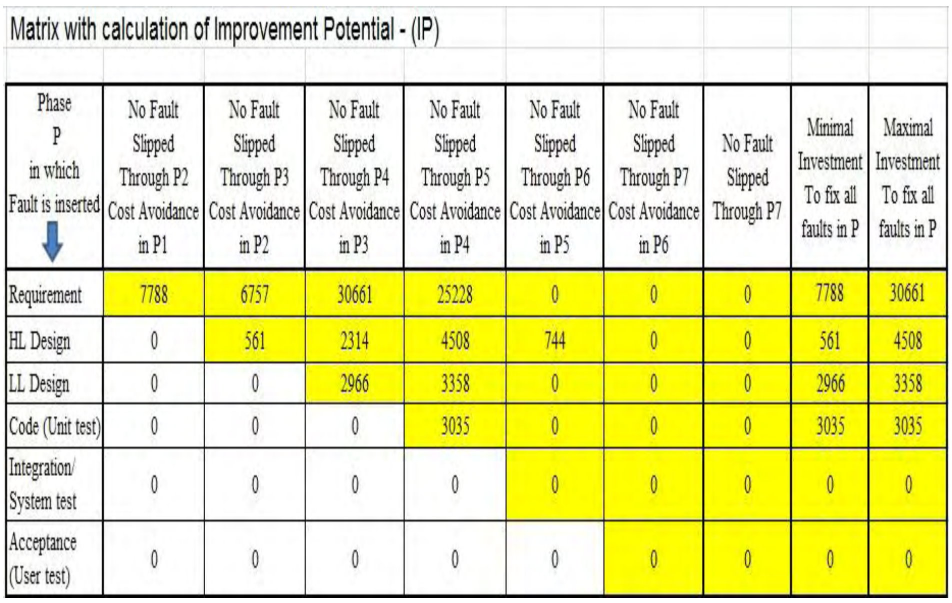
DEFECTS → / ↓ PHASE INSERTED :		FOIND IN PHASE : P₁ Requirement	P₂ HL Design	P₃ LL Design	P₄ Code (Unit test)	P₅ Integration/ System test	P₆ Acceptance (User test)	P₇ Operation Post - release	Total Injected in phase
P_1	Requirement	1,515	602	579	402	391	89	0	3578
P_2	HL Design	0	805	400	60	223	60	0	1548
P_3	LL Design	0	0	750	452	420	54	0	1676
P_4	Code (Unit test)	0	0	0	2,421	1895	114	0	4430
P_5	Integration/ System test	0	0	0	0	45	5	0	50
P_6	Acceptance (User test)	0	0	0	0	0	10	0	10
P_7	Operation Post - release	0	0	0	0	0	0	0	0
	Number of Defects found and removed in phase	1515	1407	1729	3335	2974	332	0	**11292**

Fig. 10. Original Software Process Defect Containment Matrix [2]



DEFECTS → / ↓ PHASE INSERTED :		FOIND IN PHASE %: P₁ Requirement	P₂ HL Design	P₃ LL Design	P₄ Code (Unit test)	P₅ Integration/ System test	P₆ Acceptance (User test)	P₇ Operation Post - release
P_1	Requirement	42%	17%	16%	11%	11%	2%	0%
P_2	HL Design	0	52%	26%	4%	14%	4%	0%
P_3	LL Design	0	0	45%	27%	25%	3%	0%
P_4	Code (Unit test)	0	0	0	55%	43%	3%	0%
P_5	Integration/ System test	0	0	0	0	90%	10%	0%
P_6	Acceptance (User test)	0	0	0	0	0	10%	0%
P_7	Operation Post - release	0	0	0	0	0	0	0%
	Defect Removal Efficiency [%] In phase	42.3	27.4	25.4	29.7	26.4	2.9	0.0
	Cumulative Removal Efficiency [%]	***13.4***	***25.9***	***41.2***	***70.7***	***97.1***	***100.0***	**Total DRE [%]**

Fig. 11. Software Defect Containment Percentage Matrix – PCE



Matrix with calculation of Improvement Potential - (IP)

Phase P in which Fault is inserted	No Fault Slipped Through P2 Cost Avoidance in P1	No Fault Slipped Through P3 Cost Avoidance in P2	No Fault Slipped Through P4 Cost Avoidance in P3	No Fault Slipped Through P5 Cost Avoidance in P4	No Fault Slipped Through P6 Cost Avoidance in P5	No Fault Slipped Through P7 Cost Avoidance in P6	No Fault Slipped Through P7	Minimal Investment To fix all faults in P	Maximal Investment To fix all faults in P
Requirement	7788	6757	30661	25228	0	0	0	7788	30661
HL Design	0	561	2314	4508	744	0	0	561	4508
LL Design	0	0	2966	3358	0	0	0	2966	3358
Code (Unit test)	0	0	0	3035	0	0	0	3035	3035
Integration/ System test	0	0	0	0	0	0	0	0	0
Acceptance (User test)	0	0	0	0	0	0	0	0	0

Fig. 12. Matrix with Calculation of Improvement Potential for Scenario 1

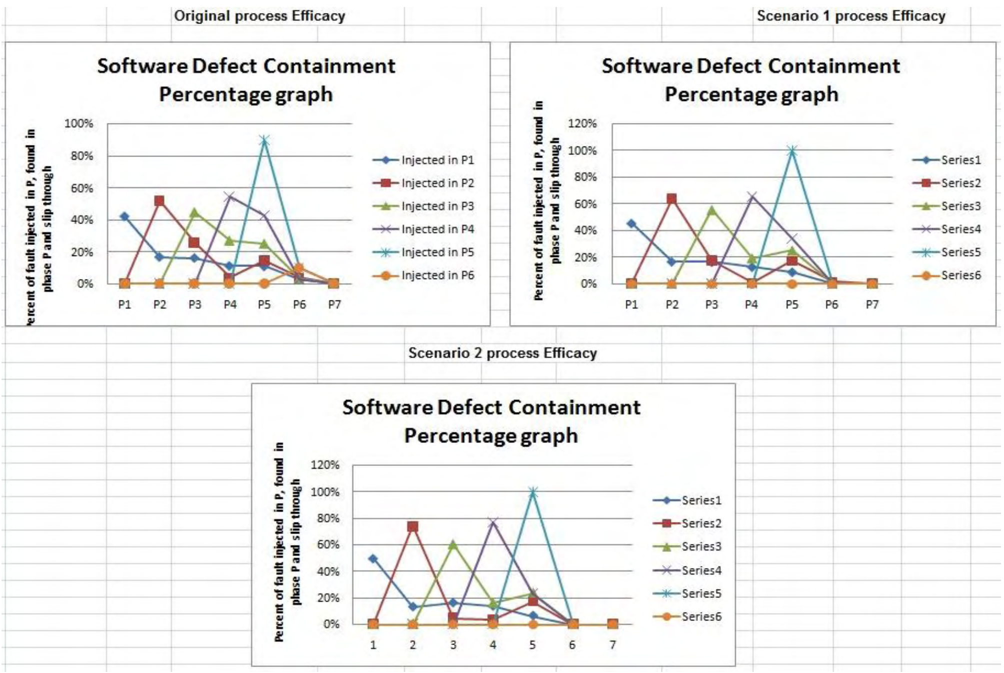


Fig. 13. Calculated Defect Containment Percentage Matrix – for original and anlized scenarios 1 and 2

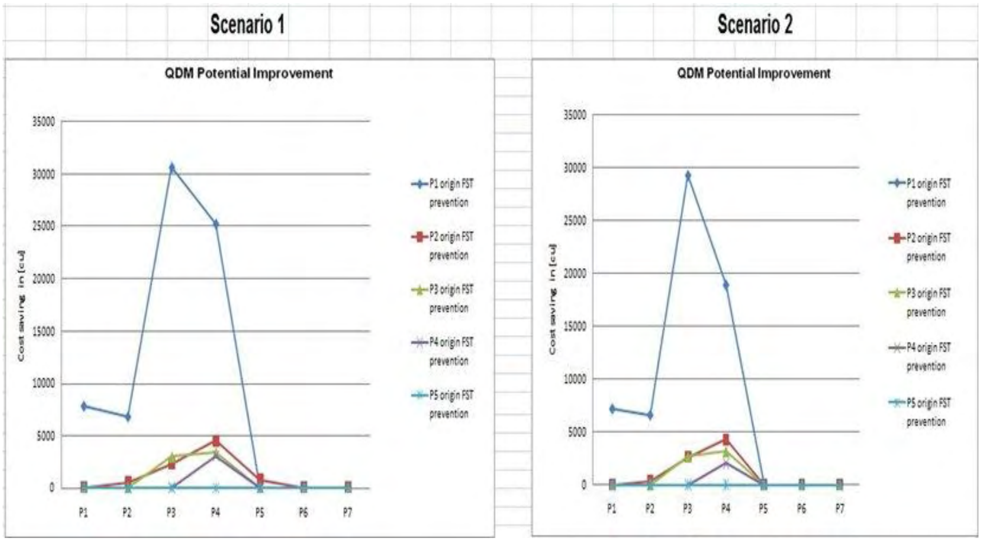


Fig. 14. Graph with calculation of Improvement Potential in [cu] for Scenario 1 and 2

We described ind this section, as answer to the RQ4 and RQ5, a Software Quality Optimization strategy of OptimalSQM framework, which is a continuous, iterative process throughout the application lifecycle resulting in zero-defect software that delivers value from the moment it goes live, with Simulated Defect Removal Cost Savings model.

6. Conclusion

The initial main research question that was posed for the complete research in this project was: How can software testing be performed efficiently and effectively i.e. **Optimal**, that is, do we have a framework model targeted specific software testing domains or problem classes described in the paper? To be able to address the main research question several other research questions needed to be answered first (RQ2–RQ5). Thus, since this project is based upon the main research question, it was worthwhile taking the time to examine the current practice in different projects and see how software quality is measured and, especially, software testing was practiced [1-8] as we described in Section 2. In Section 3 and 4 we described our OptimalSQM framework which presents a set of best practice models and techniques integrated in optimized and quantitatively managed software testing process (OptimalSQM), expanding testing throughout the SDLC. In Section 5, we explained how can Advanced Quantitative Defect Management (AQDM) Model be enhanced (as answer to RQ4 and RQ5) is practically useful for determining which activities need to be addressed to improve the degree of early and cost-effective software fault detection. To enable software designers to achieve a higher quality for their design, a better insight into quality predictions for their design choices, test plans improvement using Simulated Defect Removal Cost Savings model is offered in paper.

Finally, this paper presents and validates a method for measuring the efficiency of the software test process to achieve early and cost-effective software fault detection. That is, it determines how fault statistics can be used for assessing a test process and then quantify the improvement potential of changing the process. The described method assesses a software development organization through the following three steps:

1. Determine which faults that could have been avoided or at least found earlier, i.e. FST.
2. Determine the average cost of faults found in different phases.
3. Determine the improvement potential from the metrics obtained in (1) and (2), i.e. measure the cost of not finding the faults in the right phase.

The practical applicability of the method was determined by applying it on two industrial software development projects. In the studied projects, potential improvements were foremost identified in the the largest improvement potential is in the **LL Design** test phase, i.e. the phase triggered **30661 [cu]** of unnecessary costs in later phases due to a large FST from faults injected in Requirement phase. Therefore, the primary usage of the values is to serve as input to an expected ROI calculation according to given formula, when prioritizing possible improvement actions we can improve DRE cost for ROI=39.2%. That is, the **LL Design** phase inserted, or did not capture faults present at least, too many faults that slipped through to later phases.

References

1. Galin, D.: Software Quality Assurance:From theory to implementation, Pearson Education Limited, ISBN 0201 70945 7, pp. 135-142. (2004)
2. Frost and, A., Campo,M.: Advancing Defect Containment to Quantitative Defect Man, *CrossTalk*, December. pp. 24-28, (2007)
3. Lim, S. So, Y, Cha, S. D. and Kwon, Y. R.: “An Empirical Study on Software Error Detection: Voting, Instrumentation, and Fagan Inspection,” Proceedings of the Asia-Pacific Software Engineering Conference, IEEE Computer Society Press, Dec. (1995)
4. Jones, C.: Estimating Software Costs. 2nd edition. McGraw-Hill, New York.(2007)
5. Lazić, Lj.,Mastorakis, N.:Cost Effective Software Test Metrics, WSEAS TRANSACTIONS on COMPUTERS , Issue 6, Volume 7, June. pp. 599-619, (2008)
6. Lazić, Lj., Mastorakis, N.:Orthogonal Array application for optimal combination of software defect detection techniques choices, WSEAS TRANSACTIONS on COMPUTERS, Issue 8, Volume 7, pp.1319-1336, August.(2008)
7. Lazić, Lj., Mastorakis, N.:Optimizing Test Process Action Plans by Simulated Defect Removal Cost Savings, 11th WSEAS Int.Conf. on AUTOMATIC

CONTROL, MODELLING & SIMULATION (ACMOS'09), Istanbul, Turkey, May 30,pp. 280-287, June 1.(2009)

8. Lazić, Lj.: The Integrated and Optimized Software Testing Process. PhD Thesis, School of Electrical Engineering, Belgrade, Serbia.(2007)
9. Lars-Ola D.: Early and Cost-Effective Software Fault Detection, PhD Thesis, Blekinge Institute of Technology, SWEDEN, Section 2.(2007)
10. McConnell, S. :Professional Software Development, Addison Wesley, ISBN 0-321-19367-9.(2004)
11. Boehm, B. , Abts, C., Brown, A., Chulani,S., Clark, B.,Horowitz,E., Madachy,R., D. Reifer,D., Steece,B.: Software Cost Estimation with COCOMO II, Prentice Hall.(2000)
12. Cohen, C. F., Birkin, S. J., Garfield, M. J. and Webb, H. W.:Management Conflict in Software Testing. Communications of the ACM, 47(1),pp. 76-81. (2004)
13. Dhaliwal, J., Onita C., A framework for aligning Testing and Development, Proceedings of the Workshop on Advances & Innovations in Systems Testing. (2007)
14. Hunter and Blosch:Managing the New IT Risks”. Gartner.(2003)
15. Pettichord, B. :Testers and Developers Think Differently: Understanding and Utilizing the Diverse Traits of Key Players on your Team. Software Testing & Quality Engineering, 2(1), pp. 42-45. (2000)
16. Sabherwal, Hirschheim and Goles:Information systems – business strategy alignment: The dynamics of alignment: Insights form a punctuated equilibrium model. Strategic information management: Challenges and strategies in managing information systems, (Galliers and Leidner, Eds), Butterworh-Heinemann, Oxford pp 311-346. (2003)
17. Lazić Lj., Velasević, D.:Applying simulation and design of experiments to the embedded software testing process. STVR, Volume 14, Issue 4, John Willey & Sons, Ltd., p257-282. (2004)
18. Popovic S, and Lazic, Lj.:Orthogonal Array And Virtualization As A Method For Improvement Configuration Testing, Proceedings of 1st IEEE Eastern European Regional Conference on the Engineering of Computer Based Systems - ECBS-EERC 2009, Novi Sad, pp.148-149. (2009)
19. Lazić, Lj. , Kolašinac, A., Avdić, Dž.:The Software Quality Economics Model for Software Project Optimization. WSEAS TRANSACTIONS on COMPUTERS, Issue 1, Volume 8, pp.21-47.(January 2009)
20. Morasca, S., Serra-Capizzano, S.: On the analytical comparison of testing techniques. In: International Symposium on Software Testing and Analysis (ISSTA ‘04). Association for Computing Machinery, Boston, Massachusetts, USA, pp. 154–164. (2004)
21. Houston, D., Keats, B.:Cost of Software Quality: A Means of Promoting Software Process Improvement, Quality Engineering, 10:3, pp. 563-573, March. (1998)
22. Müller, M.:About the Return on Investment of Test-Driven Development", ICSE'03, Portland, Oregon. (2003)
23. Chen, T., Kuo, F, Merkel,R.: On the statistical properties of testing effectiveness measures. The Journal of Systems and Software 79,pp. 591–601. (2006)
24. Wang, Q. et al.:Estimating Fixing Effort and Schedule based on Defect Injection Distribution, Softw. Process Improve. Pract.; 13: pp.35–50. (2008)

Ljubomir Lazić is an assistant professor of software engineering and computer science at State University of Novi Pazar, Serbia. He received the Ph.D. degree in electrotechnics from School of Electrical Engineering, Belgrade University in 2007. He was a Post-Doctoral Researcher at The WSEAS (The World Scientific and Engineering Academy and Society) of computer science from 2009 to 2010. So far, he has authored over 50 research papers. His current research interests are optimal software project management.

Received: September 23, 2009; Accepted: May 26, 2010.

UDC 007:658:004, DOI:10.2298/CSIS090608006M

Concept of the Exception Handling System for Manufacturing Business Processes

Dragan Mišić[1], Dragan Domazet[2], Miroslav Trajanović[1], Miodrag Manić[1] and Milan Zdravković[1]

[1] Faculty of Mechanical Engineering, Aleksandra Medvedeva 14,
18000 Niš, Serbia
draganm@masfak.ni.ac.rs,traja@masfak.ni.ac.rs,
mmanic@masfak.ni.ac.rs,milan.zdravkovic@gmail.com
[2] Faculty of Information Technology, Tadeuša Košćuška 63,
11000 Beograd, Serbia
dragan.domazet@fit.edu.rs

Abstract: Business processes managed by information systems rarely operate according to the pre defined scenario. Exceptions to the pre defined workflows occur frequently. This especially applies to the production processes, which are very complex and require a constant human involvement. Workflow management systems should be capable of responding adequately to the exceptions caused by the process environment. Moreover, the response should be automatic, if possible, i.e. the workflow should automatically adapt to the new situation, or otherwise, the system administrator should be informed, so that he could take appropriate actions. This paper presents workflow management system MD, which is capable of offering a satisfying solution to the detected exceptions.

Keywords: Workflow Management Systems; Exception Handling.

1. Introduction

Workflow can be defined as the automation of a business process, as a whole or in parts, which passes documents, information or tasks from one participant to another, according to a set of procedural rules.

Workflows are managed by Workflow Management Systems. Workflow Management System (WfMS) is the system that defines, creates and manages the execution of workflows through the use of software running on one or more workflow engines, which are capable of interpreting the process definition, interacting with workflow participants and, when required, capable of invoking the use of other IT tools and applications [25].

Developing a workflow definition which completely encompasses all the variations and features of a business process is very difficult, sometimes even impossible, because of unpredictable events. Workflow definitions are usually developed on a higher, conceptual level. Since the processes defined in such a way are executed in real changing environment, situations in which the execution deviates from the basic business case occur very frequently.

Workflow model, which has been defined in advance, represents the basic, standard business case. Workflow management system should be capable of responding to exceptions, which, in fact, are deviations from the standard (designed) business case.

Some exceptions are predictable since they are known to appear periodically. The solution to that kind of exceptions is known at the moment of developing a business process definition. Other exceptions are unpredictable and they occur due to the unpredictable changes in working environment.

Much research [5, 9, 4, 14, 3] has focused on handling exceptions in workflow management systems. Research has mostly been done in handling predictable exceptions. In order to make the basic process logic clear, exceptions aren't usually integrated into the standard business process definition. Therefore, many different ways of defining and handling those exceptions separately have been suggested.

One of the flaws of existing exception handling systems is the result of exception processing. Actions taken in that case are very rudimentary, and are usually reduced to informing users, exception ignoring, making new attempts to execute the same activity or skipping the activity. Actions leading to the change of the business process definition, such as inserting new activities or deleting the old ones, are rarely supported. Moreover, handling exceptions usually refers to handling run – time exceptions. The consequences that an exception might have on future activities are very rarely taken into consideration [15].

The existing workflow management systems are usually applied in well organized environments, such as banks, insurance companies, hospitals, etc. The advantages of such environments are well defined processes through which usually flow documents presented in a digital form.

Workflow management systems are very rarely used in manufacturing environments. Exceptions are known to occur almost regularly in manufacturing environments (material is not in accordance with specification, machine is out of order, a worker hasn't come to work etc.). According to the workflow definition, the production of a mechanical part is closely connected with the occurrences of various unpredictable situations, which are dealt with on the basis of the engineers' experience. Such processes are very suitable for applying the artificial intelligence tools and expert system, which can naturally model the expert knowledge of the area.

This paper presents the concept of MD WfMS, which is being developed in Faculty of Mechanical Engineering in Nish. This workflow management system is intended for use in production environment. The system is capable of handling the exceptions in a workflow. The system core is the workflow engine. The part which refers to handling exceptions is managed by expert system whose rules are invoked at the moment of exception detection. The work of expert system can result in the modification of either the workflow definition or workflow instance definition.

The old definition will be changed if it has been concluded that it is no longer suitable for the new situation or, in other words, if all the future instances based on it need to be changed. An example of such a situation is when it has been decided that a part of an assembly should be procured from a business partner rather than produced in the factory.

If the exception is only temporary, then for the sake of overcoming it, the process instance rather than the process definition is changed. An example of such a case might be the situation when a casting needs repair, or when a very important machine is out of order so that it is necessary to produce temporarily that part using outsourcing.

The system, in certain situations, is capable of responding to exceptions in advance, i.e. of performing necessary adjustments before the activity in which the exception will occur is taken on.

An example of an exception of this kind might be the case when the machine that is going to be used in future activities, and which is otherwise irreplaceable, is out of order. We shouldn't wait for the activity to come. On the contrary, we should take steps to adapt the system in advance. In MD system, the procedure for handling the exception starts immediately after the exception has been detected (or has been concluded that the machine is out of order), although there is enough time left until the actual occurrence of the exception.

2. Exception Handling Problems and Related Work

An exception in the workflow can be defined as an event which prevents an activity from its being executed properly.

Different workflow management systems handle the exceptions in different ways. In general, methods for exception handling can be divided into:

- methods which have the predictable exceptions built in the process definition
- methods which handle the exceptions by the well structured system

The first method, which is capable of handling exceptions that can be foreseen in advance, is based on the fact that the person who is in charge of a process definition modifies it according to the exceptional situations known at workflow build time. If, for example, an exception occurs during a process, the process control flow is diverted to the attached activities that control the new situation. This approach is good in cases with small number of possible exceptions. A direct specification of many exceptions leads to the complexity of a process model. The question is whether these cases can be considered exception handling processes since the exceptions are built in the process definition. In that case, it is more apt to call these processes well-structured and totally pre defined workflow processes. This kind of approach is used in the WAMO [5, 6] system. For exception handling Sagas [8] and flexible transactions are used. WfMS Opera [9] for exception handling uses programming language primitives (mechanism is similar to exception handling in object oriented programming languages, as Java).

Handling exceptions according to pre defined rules requires the use of a rule-based language. This language is used for exception specification. The rules are usually in the form of ECA (event condition action) rules. The event part defines exception symptoms. The condition part compares these symptoms with the real situation, while the action part defines what it is necessary to do in that case. The existing exception handling systems usually use some special rule based languages, designed specially to satisfy the needs of a particular system. For example, WfMS system WIDE [3] uses language Chimera-Exc. ADOME [4] uses similar approach for its language. METEOR [14] uses the so called "Justified ECA rules (JECA)" where justified refers to defining a special context in which ECA rules for exception handling are applied.

On the level of active rules application, research has been carried out in temporal ECA rules. For example, Active TFL is used for defining rules in Agent Work [18] system. Active rules are applied in assembling e-service [1] (these rules are applied to XML documents). In [20] temporal ECA rules are used for defining E- business software architecture.

Some researchers are trying to apply the previous experiences in handling the similar exceptions [22, 11]. In that case, it is the most natural to use Case-Based Reasoning. One of the problems when using Case-Based Reasoning is how to determine the degree of similarity between the old cases and the new ones [2].

Based on previous analysis it can be concluded that the majority of existing workflow management systems use special languages for exception handling. These languages have been designed to satisfy the needs of particular systems. On the other hand, there are several organizations which deal with the process management system, the best known of which is Workflow Management Coalition (WfMC). This coalition has designed a special language for the definition and specification of a workflow process– XPDL [23] (Xml Process Definition Language). This language defines a Meta model that is used for defining types of constituent components of a process definition. However, XPDL does not contain all elements needed for exception definition.

MD WfMS, uses extended XPDL as the basic language for process definition. The term extended refers to extensions i.e. language constructions which are capable of exception handling. Those constructions are inserted into the standard XPDL. For the core of the system, open source system Shark [7], which had been developed in Java programming language, was chosen. Shark was used for the basic part of this WfMS system i.e. for execution engine. This system was chosen since the process is defined in XPDL which had been recommended by WfMC coalition and which is standard in the area of modeling workflows. The system also uses OMG

(ObjectManagementGroup) recommendations regarding the interface which workflow management system should implement.

The extended version of the basic XPDL, however, encompasses the constructions which are capable of handling exceptions. The extensions are predominantly related to exception detection. After the exception has been detected, the whole process definition is being sent to the expert system whose task is to propose a solution to the detected problem. That solution may be:

- a new changed process definition
- the modification of the process instance or
- the modification of the particular activity.

The process definition is modified when it is necessary to make essential changes, not the temporary ones. An example of such a situation is when the organization decides to procure a part of an assembly from a supplier, and intends to do so in the future, instead of producing the part itself.

The process instance is modified when an exception occurs as a result of sudden circumstances, which do not affect other instances derived from the same process definition. An example of this type which describes the functioning of MD system will be given later in this paper.

Modifying the activity actually means modifying the data of that activity itself. In essence, this kind of modification is the subtype of the process instance modification since neither the new activities are added nor the old ones are deleted. Therefore, the workflow remains unchanged.

The existing workflow management systems are not usually capable of handling the exceptions properly. The primary goal of such systems is defining and managing processes that flow without any exceptions. If the exception handling mechanism is the constituent part of the system, then it means that the whole system has originally been designed to be capable of handling the exceptions. MD system is an example of how a system, which has not been designed to respond to exceptions, can be extended and linked with another system, in this case the expert system in order to handle the exceptions successfully.

3. System Architecture

In MD WfMS (figure 1), workflows are defined by a process manager, i.e. the manager who is responsible for planning and managing the process. Workflows are defined by a person who is in charge of them, usually a business process manager i.e. the manager who is responsible for planning and managing the process. According to that definition, the system administrator with the assistance of an editor enters the process model. If the structure of the organization allows that, it is possible for the process manager to enter the process definitions himself. The definition is entered by means of graphic process editor. Workflow definition doesn't contain the activities that should be carried out in case of unpredictable exceptions.

Therefore, the workflow definition is not loaded with numerous additional activities, which will be carried out in case of an unpredictable situation.

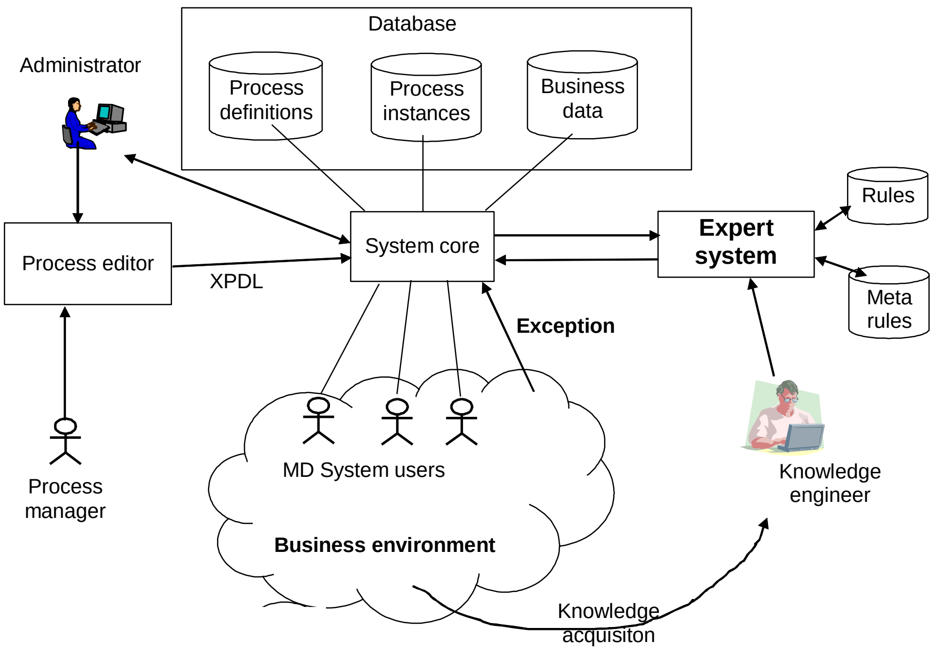


Fig. 1. MD WfMS architecture.

4. Exception Types, detection and handling

4.1. Exception types

According to the possibility of detection in the MD system, we divided the exceptions as following:

- exceptions related to data
- exceptions related to time (violation of deadlines)
- exceptions related to resources
- exceptions signalized by people.

Exceptions related to data are exceptions arising from data values that participate in the processes.

These exceptions can be further divided into complex and simple. Complex exceptions related to data are exceptions that can only be detected tracking several values.

Simple exceptions are the exceptions that can be detected based on only one value. An example is when the value of some parameter does not lie within a previously determined range.

Exceptions related to time are the exceptions that can be detected by measuring time. If the definition of process defines a specific time when certain activity must be done, measuring time enables checking whether those deadlines are met, which can be considered as an exception.

Depending on the type of time measurements used, these exceptions can be divided as following:

- Exceptions based on exceeding the duration of an activity
- Exceptions based on exceeding the deadline of an activity
- Exceptions based on exceeding the waiting time for execution of an activity
- Exceptions based on exceeding the duration of the whole process, or a group of activities
- Exceptions based on exceeding the deadline of the process, or a group of activities

Exceptions based on resources are probably the exceptions that most frequently appear in a workflow. Resources are material resources (machines and equipment necessary for some activities) and human resources.

Exceptions signaled by people: Beside various automatic ways of exception detection, the system will often report situations when it is not able to automatically detect the exception. In such situations human intervention is needed.

4.2. Exception Detection

One of the problems which arise during the application of the exception handling system is how to detect an exception. MD system can detect data exceptions, time limit exceptions with regard to exceeding time limitations, and resource exceptions.

Exceptions related to data can be detected in two ways. Simple exceptions are detected by checking the value of particular data after they have been entered by a system user or calculated by the system.

Complex exceptions related to data are the ones that are the most difficult to detect. These exceptions are the most general type of exceptions, since values and presence of different data and information can in certain situations represent the exception, in some others not, which is very hard to find out. MD system uses ASM [19] (Active Semantic Model) for the detection of these exceptions.

Linking the ASM system with MD system, as well as the way of exception detection, is described in detail in [16]. At this point, we will just mention that at the moment of entering the new information in the system (which does not exist in the model), a conclusion mechanism of ASM [27] (based on artificial intelligence), which is capable of classifying a new situation as an exception, is activated. Therefore, if a certain situation has been classified as an exception, data are sent to the expert system which afterwards makes a decision about the actions that should be taken.

Time exceptions are detected by measuring activity duration, or measuring the time that has passed until the available activity is taken on. Temporal ECA rules might have been used for the description of exceptions related to time, but, since the events whose occurrence is monitored here are quite simple (exceeding the defined duration), there has been no need to define complex conditions related to time intervals. These rules might be included later in further work on the development of this system.

Time limitations that should be monitored are defined by, Limit and Deadline, the elements of XPDL scheme. SimulationInformation is another element within the scope of XPDL scheme which stores information related to the simulation of workflows that are being monitored. According to the original plan, such information should be used by the module for the simulation of the workflow process. Since Shark does not contain this kind of module, SimulationInformation element was used for defining the estimated duration of the activity (element Duration) and estimated waiting time for the activity to be taken on by human resources. These two elements, Duration and Waiting Time, are a part of Time Estimation element. XPDL definitions of these elements are:

```
<xsd:element name="SimulationInformation">
  <xsd:complexType>
   <xsd:sequence>
     <xsd:element ref="xpdl:Cost"/>
     <xsd:element ref="xpdl:TimeEstimation"/>
   </xsd:sequence>
   <xsd:attribute name="Instantiation">
    <xsd:simpleType>
      <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="ONCE"/>
        <xsd:enumeration value="MULTIPLE"/>
      </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
  </xsd:complexType>
</xsd:element>

<xsd:element name="TimeEstimation">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="xpdl:WaitingTime" minOccurs="0"/>
    <xsd:element ref="xpdl:WorkingTime" minOccurs="0"/>
    <xsd:element ref="xpdl:Duration" minOccurs="0"/>
   </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Previously mentioned time limitations are monitored by means of time management module which is the constituent part of MD system. This module functions on the basis of checking violations of time limitations at regular time intervals. A time interval is adjustable.

Besides the information entered at the modeling time of the workflow, MD system uses the information obtained during the execution of the workflow process. Execution time and the acceptance time are monitored for each activity and the data are then stored in the database. At the moment of checking whether an activity has violated the predefined time limit, besides the time defined in the model, the times previously stored in the database (real times) are used too. The mean value of the times from the database is compared with current time.

In case the anticipated time has passed, it is considered that an exception has arisen.

Exceptions related to resources are detected by means of a special resource management module, which is linked with MD system. In case that a resource is no longer available, the resource management module immediately sends an appropriate message to MD system. MD system will automatically check whether that resource is used by any activities in the current process instances. If there is such an activity, it is considered that an exception occurs and the mechanism for its handling is activated. Resources in the process are defined by means of MdResource element, which is integrated in XPDL scheme. This element is a part of Activity element. XPDL definitions of added elements are:

```
<xsd:element name="Activity">
 <xsd:complexType>
    <xsd:sequence>
            ...
         <xsd:element ref="xpdl:MdResources"
                                 minOccurs="0"/>
    </xsd:sequence>
          ...
 </xsd:complexType>
</xsd:element>

<xsd:element name="MdResources">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="xpdl:MdResource" minOccurs="0"
                         maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="MdResource">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element ref="xpdl:Description" minOccurs="0"/>
   <xsd:element ref="xpdl:MdResourceAttributes"
             minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:NMTOKEN"
               use="required"/>
  <xsd:attribute name="Name" type="xsd:string"/>
 </xsd:complexType>
</xsd:element>
```

4.3. Exception Handling

XPDL definition, which has been created by graphical process editor, is sent to the system core (execution engine). The system core is responsible for creating process instance (based on the process definition), and also for the process execution i.e. forwarding activities to corresponding resources.

As soon as an exception is detected, both the process definition and the exception cause are sent to the expert system. We use expert system created via JESS expert system shell [12]. This is the Java rule based system, created in Sandia National Laboratories, from Livermore, California.

In the expert system Meta rules come first.

WfMS system can administer processes from different application domains, such as government, hospitals or manufacturing. Since the domains can be diverse, the predefined rules for solving the run time exceptions needn't be the constituent parts of the expert system. According to that, there are Meta rules which are to decide whether there is the domain knowledge for a particular exceptional case or not.

If an exception related to data is in question, it can happen that a system does not contain enough data for the expert system to draw a conclusion. In such cases the user is granted the right to enter additional data. In such situations, the system user is asked certain questions by defined domain rules. Answering the questions, the user provides the expert system with information necessary for drawing a conclusion. Since the questions are domain specific, certain rules are invoked only if the domain they are defined for is in question.

Considering the casting machining, there are certain rules which are capable of handling exceptions which occur due to the irregularities in casting structure. Therefore, the first question that is posed is whether it is the irregularity of the casting structure. If that is the case, the further questions which enable a more detailed exception description are posed. On the basis of the answers obtained, the expert system is capable of solving the problem.

If an exception has been detected by means of a control parameter, it may become necessary that a human participant enters some additional data, which provides a more detailed description of a particular exception. Therefore, certain rules are developed for asking process specific questions that a human participant has to answer.

Since the expert system is linked with WfMS, the human participant is not even aware of this internal communication. The user is asked questions only if the expert system does not contain enough information to draw a conclusion. If the expert system contains rules for solving a certain problem, then the facts necessary for drawing conclusions are known. The questioning mechanism is devised to enable entering such facts.

The expert system encompasses two defined classes (defftemplate): question (representing questions which the user is asked), and answer (for user's answers).

```
(deftemplate question
          (slot type)
          (slot text)
          (slot ident)
          (multislot valid)
)

(deftemplate answer
          (slot ident)
          (slot text)
)
```

Both the set of posed questions and the set of offered answers depend on the problem that is being solved. For example, in order to demonstrate the application of MD system (pump making), the following questions out of the set of questions were posed:

```
(defrule is-a-hole
=>
(assert (question (type yes-no) (text "Is it a
cavity?") (ident cavity) (valid yes no)))
)
(defrule porousness
          (answer (ident cavity) (text ?d))
=>
          (if (eq ?d yes) then
            (assert (question (type multi) (text "How
large is porousness?") (ident porous) (valid large
medium small)))
          else
            (assert (answer (ident no-solution) (text
no-solution)))
          )
)

(defrule load-pump-rules
          (answer (ident cavity) (text yes))
          (answer (ident porousness) (text ?por))
=>
          (clear)
          (batch pump_rules.clp)
)
```

These questions are posed during the cast processing if a user detects a hole which can’t be repaired by further lathe processing.

If the expert system has the domain knowledge for this kind of exception, then the whole process definition (Java objects) is sent to it. These objects are recorded into the facts base of the expert system. In addition to the process definition, expert system receives the data related to the specific process instance within which the exception occurred, as well as the exception description, no matter whether it was generated automatically or by a user.

The expert system offers two types of solutions: the first type of solutions refers to the specific process instance and the second type of solutions refers to process definition.

Disregarding the solution type, the expert system creates a new process definition. This definition, conditioned by a particular problem, may contain some new activities added by the expert system, or may delete some old ones. The fact that the exception can affect future activities within the same workflow may be also taken into consideration. New process definition is sent back to the WfMS system. Together with the process definition, expert system forwards data defined for particular process instance.

The workflow instance, within which an exception has occurred, is in the meantime suspended, and its execution stopped. The new process instance is automatically created on the basis of the new process definition coming from the expert system. The new process instance is automatically processed to the workflow part within which the exception has occurred. From then on the execution of the workflow continues according to the new workflow definition.

If the newly created solution affects the remaining process instances (for example, the machine that is going to be used in further activities is out of order), then the remaining process instances derived from the old process definition are modified too. This happens only if the particular process instance hasn’t come yet to the activity within which the problems are detected. If that activity has already been executed, the modification of the process definition is not necessary.

The old process definition remains the constituent part of the WfMS and may be used later. Whether the new process definition or the old one will be used depends both on the type of exception and the expert system conclusions. These conclusions are stored in the data base. The data base also contains the new process definition and the old one, information whether the changes are to be applied only to the current process instance, all the new instances or to the instances whose execution has not come to a particular activity yet. The transfer of the information from the expert system to MD system is performed by means of a special class which contains previously mentioned data.

New rules for solving some new problems can be defined within the expert system at any time. New rules can be added to the rule database independently from the WfMS because they are not directly related to the system. Consequently, the WFMS can be applied in new domains and the knowledge about the predictable problems within the system can be modified and adjusted.

The set of solutions which the expert system proposes will grow in time in accordance with the extension of the knowledge domain. In spite of that, new exceptions, for which the solution does not exist, will arise. In that case, WfMS allows the system user to offer a solution himself.

If the problem for which the Expert System doesn't have the solution occurs, the graphic process editor is opened and the actual process definition is automatically loaded. In that case, the user is granted the right to modify the process in order to find a solution to the problem. The procedure after the new process has been defined in this way is similar to the one after the solution has been offered by an Expert system. The new process definition is sent back to the WfMS, which automatically applies new process instance (derived from the new process definition), and the data from the earlier process instance are still present.

Let's take a look at the example of processing mechanical part of great dimensions. Let's assume that there is only one machine available in the factory for processing such a big part. If that machine were out of order, an exception in the process would occur since the activity of lathe processing couldn't be performed. In that case solutions can be different, ranging from waiting for the mechanical part to be repaired, to sending the part to a business partner to process it. If there aren't rules in the system which will handle this exception, the user can choose the option of opening the editor with process definition. Then the user, at his own discretion, can change the process. Since the process instance that is being executed is in question, the activity that is being performed is marked in the open process. It is up to the user to adjust the process to new circumstances. In this case it means that the new activities related to processing the part by a business partner will be added. After he has devised a new process definition, the user should also define the data related to the application of the new process definition. These data are the ones that the expert system issues at the moment of automatic problem solution (whether the new definition applies to all the instances or just to the current one).

5. An example

As an example we will observe pump making process in the pump factory. The simplified schema of the whole process is shown in the figure 2.

The pump impeller is usually made from the casting, which is afterwards processed on cutting machine tools until the final form is obtained. A problem in pump making process can occur due to the inappropriate casting process, which may result in obtaining the casting with cavities. Manufacturing the pump from such a casting cannot be continued until some corrections have been made. Corrections (if it is possible for the part to be corrected) are made by filling the cavities by welding.

Visual irregularity detection in the casting structure is not usually possible until the process comes to cutting. Only after the removing of material layers has started, the irregularities can be detected. In that case the cutting process stops.

Depending on the material porosity, an engineer makes a decision whether the part will be overhauled or declared as waste.

The following activities can be used for overhauling the part: inserting a new element, welding, filling the cavities with glass water or multimetal.

The decision regarding which of the aforementioned activities will be used depends on the cavities dimension and conditions of utilization.

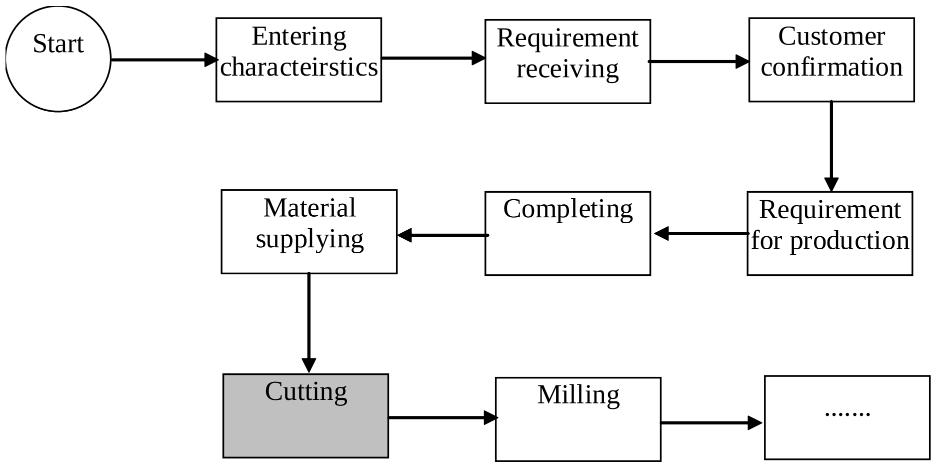


Fig. 2. Example of a pump making process.

Welding process is preceded by digging the material i.e. broadening the hole. This means that two new activities, digging and welding, are added before lathe process. Therefore, the original process is extended with two new activities. After these two activities, the execution of the workflow continues according to the previous plan (figure 3).

Consequently, the XPDL process definition is modified and sent back to the system core which continues the execution of the process according to the new process definition. The new process definition is applied only in that particular instance without affecting other instances of the same process.

The following example refers to exception detection using the control parameter whose value is entered by a user. After the exception has been detected, the questioning mechanism, enabling a more detailed description of the situation, is activated. According to that, in the cavity case, the user first marks that the cavity has been detected, and then he is questioned about its dimension. On the basis of the information obtained, the new rules, which decide the way the problem will be solved, are applied. The modified process definition is sent back to WfMS according to the scheme which has already been presented.

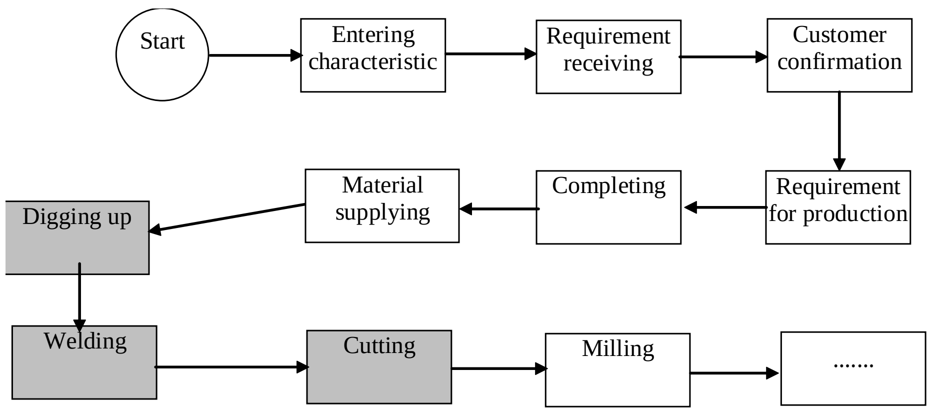


Fig. 3. Changed pump making process.

Knowledge base for resolving this problem is:

```
(defrule welding1 "porousness corrects by welding"
 ?f <- (answer (ident cavity) (text yes)); entered by
user
(answer (ident porousness) (text large)); entered by
user
(answer (ident place) (text accessible));entered by
user
   (Process-data (fluid-type ?type) (temperature ?t))
    (test (or (and (eq ?type water) (> ?t 120)) (eq
?type acid)))
=>
         (assert (action welding))
         (assert (place before current))
   (retract ?f)
)

(defrule welding2 " porousness corrects by welding"
 ?f <-    (answer (ident cavity) (text yes)) ;entered by
user
```

```
          (answer (ident porousness) (text large))
;entered by user
          (answer (ident place) (text accessible))
;entered by user
          (answer (ident part-type) (text rotational))
;entered by user
=>
          (assert (action welding))
          (assert (place before current))
        (retract ?f)
)
(defrule navarivanje3 "porousness corrects by welding"
 ?f <-    (answer (ident cavity) (text yes)) ;entered by
user
          (answer (ident porousness) (text large))
;entered by user
          (answer (ident place) (text accessible))
;entered by user
          (answer (ident stress) (text dynamical))
;entered by user
=>
          (assert (action welding))
          (assert (place before current))
        (retract ?f)
)
(defrule reject1
  (answer (ident cavity) (text yes))
  (answer (ident porousness) (text large))
  (answer (ident stress) (text dynamical))
  (answer (ident place) (text inaccessible))
=>
  (assert (answer (ident reject) (text part-rejected)))
)
(defrule welding-change-process
  (action welding)
  (place before current);place for inserting new
activities.
  (current-activity ?x); current activity id
  (process-definition-id ?processDefId);process Id from
XPDL definition
=>
  (bind ?prevAct (get-previous-activities ?x));list of
previous activities
  (bind ?packageDefinition (fetch PackageDefinition));
XPDL package definition
  (bind ?activities (get-activities-from-process
?processDefId)); all process activities (list of Java
classes(
  (bind ?process (get-process-from-definition
?processDefId)); Java object of process
```

```
  (bind ?transitions (call ?process get
"Transitions")); list of Java objects with transitions
between acitivities
  (bind ?currentAct (call ?activities getActivity (get-
activity-definition-id ?x))); Java object with current
acitivity
  (bind ?prevActObject (call ?activities getActivity
?prevAct)) ;Java object with previous activity
  (bind ?b1 (call org.enhydra.jawe.MisicProba
createActivity "raskopavanje" "raskopavanje"
    ?activities ?process "Opis" 1 ""
"FreeTextExpressionParticipant" 1  1 "" "" ""
nil));Java object for new activity
  (bind ?b2 (call org.enhydra.jawe.MisicProba
createActivity "navarivanje" "navarivanje"
    ?activities ?process "Opis" 1 ""
"FreeTextExpressionParticipant" 1  1 "" "" "" nil))
;Java object for new activity
  (call ?activities add ?b1)
  (call ?activities add ?b2)
  (bind ?tNew1 (call Utility createTransition
?transitions ?prevActObject ?b1)); new transition
  (bind ?tNew2 (call Utility createTransition
?transitions ?b1 ?b2)) ; new transition
  (bind ?tNew3 (call Utility createTransition
?transitions ?b2 ?currentAct)) ; new transition
  (call ?transitions add ?tNew1); adding new transition
to list of all transitions
  (call ?transitions add ?tNew2)
  (call ?transitions add ?tNew3)
  (call Utility removeTransition ?prevActObject
?currentAct ?transitions); removing unnecessary
transition
(bind ?strategy (call org.enhydra.jawe.MisicProba
createStrategyClass ?processDefId ?process
?currentProcess ?currentActivity "change"
"CURRENT_INSTANCE_ONLY")); creating an object of class
;strategy with information about using new process
;instance. Attributes are: old XPDL process definition,
;new XPDL process definition, Java object with current
;process instance, java object with current activity
;instance, change type, and fact that new process
;definition should apply to current instance only
)
```

6. Conclusion

Today's workflow management systems are not so robust because they lack the capability to handle exceptions which lead to the deviation of the workflow from the pre defined workflow model. If a workflow management system is capable of handling the exceptions, it means that the exceptions have become the constituent part of it at build time of workflow definition. Such systems use special process defining languages which have the constructions for exceptions handling built in.

MD system, which is presented in this paper, treats the exception handling a little differently. We have tried to extend the existing WfMS without changing it radically. Therefore, we have linked the system core to the separate expert shell capable of defining rules for handling exceptions. Any system can be extended in similar way.

Now it is possible to define exceptions independently from the main workflow. Consequently, the workflow is no longer overloaded with redundant information. Furthermore, since WfMS can be applied to different environments, it has become easy to add new rules for handling the exceptions under new conditions, and this contributes to scalability of the system. On the other hand, the practical application of the WfMS will lead to the extension of the existing exception handling knowledge. The proposed workflow model makes the extension of the existing system possible without affecting the WfMS core. In that way, the system acquires modularity which enables it to be applicable in various domains. When applying the WfMS in a completely new domain, it is necessary only to define the knowledge base of the Expert system. Modification of the knowledge domain does not affect the system core.

The exception handling system can resolve controversial situations in different ways. Depending on its expert knowledge, the WfMS can be extended in such a way that it becomes capable of handling various exceptions. Therefore, some new activities can be added, or the existing ones modified or deleted. The thorough analysis of the workflow enables us not only to predict the influence the exception will have on future activities, but also to make proper corrective actions in advance.

Acknowledgment

The paper is part of project TR12010 – “Active product semantic data model” that is sponsored by Ministry of Science and Technology for the period of 2008-2010.

References

1. Bonifati, A., Ceri, S., Paraboschi, S.: Active rules for XML: A new paradigm for E-services. The VLDB Journal 10, 1, 39-47 (2001)
2. Burkhard, H.D.,: Case Completion and Similarity in Case-Based Reasoning, International Journal, Computer Science and Information Systems, Volume 1, Number 2. (2004)
3. Casati, F.,: Specification and Implementation of Exceptions in Workflow Management Systems. ACM Transactions on Database Systems, vol.24, no.3, pp.405–451. (1999)
4. Chiu, D. K. W., Li, Q., Karlapalem, K.,: ADOME-WFMS: Towards Cooperative Handling of Workflow Exceptions. ECOOPWorkshop 2000: Advances in Exception Handling Techniques, Lecture Notes in Computer Science, vol.2022, pp.271–288. (2001)
5. Eder, J., Liebhart, W.,: Contributions to Exception Handling in Workflow Management. EDBT Workshop on Workflow Management Systems. Valencia, Spain. (1998)
6. Eder, J., Liebhart, W.,: The workflow activity model WAMO. In Proceedings of the International Conference on Cooperative Information Systems, Vienna, Austria. (1995)
7. Enhydra Shark: Open Source Java/XML Workflow Engine, http://shark.enhydra.org/
8. Garcia-Molina, H., Salem, K.,:"Sagas". Proc. ACM SIGMOD (1987).
9. Hagen, C., Alonso, G.,: Exception Handling in Workflow Management Systems. IEEE Transactions on Software Engineering, vol.26, no.10, pp.943–958. (2000)
10. Hwang, G.H., Lee, Y.C., Wu, B.Y.,: A New Language to Support Flexible Failure Recovery for Workflow Management Systems, Springer-Verlag Berlin Heidelberg. (2003)
11. Hwang, S., Tang, J.: Consulting past exceptions to facilitate workflow exception handling. In: Decision Support Systems 37, 1, pp. 49-69. (2004)
12. JESS, the Rule Engine for the JavaTM Platform, Sandia National Laboratories, http://herzberg.ca.sandia.gov/jess/.
13. Kumar, A., Wainer,J., Zhang, Z.,: Meta-workflows and ESP: A Framework for Coordination, Exception Handling and Adaptability in Workflow Systems. Springer-Verlag Berlin Heidelberg. (2004)
14. Luo, Z., Sheth, A., Kochut, K., Miller, J.,: Exception Handling in Workflow Systems. Applied Intelligence: the International Journal of AI, Neural Networks, and Complex Problem-Solving Technologies, vol.13, no.2, pp.125–147. (2000)
15. Miler, R.: Event-Oriented Dynamic Adaptation of Workflows: Model, Architecture, and Implementation, dissertation, University of Leipzig. (2002)
16. Mišić D., Stojković M., Domazet D., Trajanović M., Manić M. and Trifunović M.: Exception detection in business process management systems, Journal of Scientific and Industrial Research, in press.
17. Muehlen, M.: Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems. Logos, Berlin. (2004)
18. Müller, R., Greiner, U., and Rahm, E.: AGENT WORK: a workflow system supporting rule-based workflow adaptation. Data Knowl. Eng. 51, 2, 223-256. (2004)

19. Stojkovic, M., Manic, M., Trajanovic, M., Korunovic, N.: Semantic Structures In The Product Data Model. Proceedings of International Conference on Product Lifecycle Management, PLM-SP3, 227-234. (2007)
20. Wan, H., Li, L.:. E-business software architecture based on temporal ECA rules and actions conflicts management. Int.Conf on E-Business Engineering (ICEBE), Beijing China, pp 208-211. (2005)
21. Weber, B., Rinderle, S., Reichert, M.: Change Support in Process-Aware Information Systems - A Pattern-Based Analysis, Tech. Rep. TR-CTIT-07-76, CTIT, University of Twente, The Netherlands. (2007)
22. Weber, B., Wild, W., Breu, R.: CBRFlow: Enabling Adaptive Workflow Management Through Conversational Case-Based Reasoning, Springer-Verlag Berlin Heidelberg. (2004)
23. WfMC (XPDL): Workflow Process Definition Interface - XML Process Definition Language. Document Status - 1.0 Final Draft. Document Number WFMC-TC-1025. Workflow Management Coalition 2002.
24. Workflow Management Coalition, The Workflow Reference Model
25. Workflow Management Coalition, Terminology and Glossary, Workflow Management Coalition 1999.
26. Wu, S., Guo, B.: Design of an exception handling algorithm of workflow. ISECS International Colloquium on Computing, Communication, Control, and Management, pp.281-284. (2008)
27. Misic, D., Stojkovic, M., Domazet, D., Trajanovic M., Manic, M., Trifunovic, M. : Exception detection in business process management systems. Journal of Scientific and Industrial Research, pp. 188-193. (march 2010)

Dragan Mišić is research and teaching assistant at Faculty of Mechanical Engineering, University of Niš. He works with workflow management and business process management systems and their application in the manufacturing business environments. He is the author of more than 30 scientific and professional papers.

Dragan S. Domazet is a full professor at the Faculty of Information Technology in Belgrade, of which he is dean and founder. He has worked on the application of object-oriented bases in the product development by applying the STEP standard and on the application of PDM (Product Data Management) systems. His areas of work and research are eLearning systems, analysis and reengineering of business processes, making feasibility studies for the introduction of new information systems, development of systems for collaboration engineering, etc

Miroslav Trajanovic, professor at Mechanical Engineering Faculty, University of Nis, Nis, Serbia, has had got 30 years of experience in application of IT in mechanical engineering and education. He is expert for computer programming, CAD, finite element method and expert systems. He is the author of more than 140 scientific and professional papers. Professor Trajanovic was also project leader for 10 projects mainly in IT and Mechanical Engineering, as well as two FP6 and two FP7 projects.

Miodrag Manić is professor at Faculty of Mechanical Engineering, University of Niš. He is head of Laboratory for Intelligent Manufacturing Systems. Science fields are using of ICT in design of product and manufacturing technology, especialy integrated CAD/CAPP/CAM and CNC programing machine tools. He is participated in eighteen scientific-research projects, he is author of 120 scientific-technical papers, and co-author of two university textbooks.

Milan Zdravković is research and teaching assistant at Faculty of Mechanical Engineering, University of Niš. His work is focused to using semantic web tools, languages and service oriented architectures for facilitating interoperability in inter-organizational collaborative forms, such as manufacturing supply chains.

Received: June 08, 2009; Accepted: March 08, 2010.

UDC 004.451, DOI:10.2298/CSIS090223003D

Reasoning with Linguistic Preferences Using NPN Logic

Goran Devedžić[1], Danijela Milošević[2],
Lozica Ivanović[1], Dragan Adamović[1], and Miodrag Manić[3]

[1] University of Kragujevac, Faculty of Mechanical Engineering
Sestre Janjic 6, 34000 Kragujevac, Serbia
{devedzic, lozica, adam}@kg.ac.rs
[2] University of Kragujevac, Technical Faculty at Čačak
Svetog Save 65, 32000 Čačak, Serbia
danijela@tfc.kg.ac.rs
[3] University of Niš, Faculty of Mechanical Engineering
Aleksandra Medvedeva 14, 18000 Niš, Serbia
mmanic@masfak.ni.ac.rs

Abstract. Negative-positive-neutral logic provides an alternative framework for fuzzy cognitive maps development and decision analysis. This paper reviews basic notion of NPN logic and NPN relations and proposes adaptive approach to causality weights assessment. It employs linguistic models of causality weights activated by measurement-based fuzzy cognitive maps' concepts values. These models allow for quasi-dynamical adaptation to the change of concepts values, providing deeper understanding of possible side effects. Since in the real-world environments almost every decision has its consequences, presenting very valuable portion of information upon which we also make our decisions, the knowledge about the side effects enables more reliable decision analysis and directs actions of decision maker.

Keywords: Fuzzy Cognitive Maps; Negative-Positive-Neutral Logic; Linguistic Preferences; Decision Analysis.

1. Introduction

In the mid-seventies Axelrod [1] proposed representational framework for causal knowledge, namely cognitive maps (CM). Although essentially important this framework has relatively rigid structure. This is due to the fixed ("minus-plus": (-, 0, +) or (-1, 0, +1)) causality measures between concepts. Such structure hardly could capture dynamical behavior of the systems. Further progress in this field provided Kosko in the mid-eighties [2], proposing non-rigid measures of concepts causality. Basic causal relationship between concepts can be described not only as "increasing" or "decreasing", what "plus" and "minus" means in original cognitive maps setting, but also as "increasing to some degree" and "decreasing to some degree". These measures can be expressed and assessed by human experts in numbers as well as by words (with deeper meaning), depending on subjective experience, beliefs and practice. In the most cases this kind of measures usually obey certain mathematical laws which classify them as fuzzy measures. Hence, such framework with non-rigid structure for causal knowledge representation is called fuzzy cognitive maps (FCM). These causal schemes, which allow propagation of causality, represent a system behavior in an adaptive way, depicting situations in given environment more realistic.

In general, systems modeling requires high level of expertise to properly identify and represent complex interrelationships between their elements in order to achieve stable operation. Such state represents equilibrium of the system. One of the well known approach to aggregating multicriteria to provide an overall decision function on system behavior includes diferent classes of ordered weighted aggregation (OWA) operators [8]. However, OWA operators in their definition suggest ordered, permuted position of the weights, instead of direct association with each particular criteria (attribute, system variable). Although reaching equilibrium in the physical system may resemble an aggregation of different criteria, which provides the sort of "optimal balance" of the system, it does not assume an arbitrary layout and interconnection of the system's elements and their corresponding weighted links. In other words, systems are connected sets of elements that behave as a whole, and therefore causally dependent. Furthermore, according to the nature of the physical systems and, consequently, their behavior, the sequence of parameters tuning requires more or less strict interdependent order, which does not assume property of symmetry. Also, the system's components are interlinked, and links represent their weighted dependency, described by so-called connection matrix, where the sum of (all) weights is not necessarily equal to one ($\Sigma w_i \neq 1$). In such situation FCM's representational framework can provide necessary multicriteria analysis of a (complex) systems.

Since FCMs model systems, we cannot expect that precise tuning of one element will not, in some sense, negatively effect other elements disabling their perfect tuning. That is, we can only approximately or more or less accurately for the best, tune the elements in order to achieve optimal operation of a system as a whole. Therefore, it is of a big importance to measure such effects that "negatively" affect the elements (concepts) of a system, directing our actions toward achieving optimality and balance. By the end of eighties of the last century, Zhang et al. [3] presented generic approach to cognitive map development and decision analysis and introduced negative-positive-neutral (NPN) logic, which provide the basics for reasoning with logic values from [-1, 1] interval and modeling of negative or, so-called, "side effect" of decisions.

In the original setting of FCMs theory it is stated that causal weights between concepts are constant and only the concepts values change in time [2], [4]. This holds even in the case of concepts that are difficult to measure, such as social, socioeconomic, geographical, medical, military, political, etc. Values of these mostly perception-based concepts also change in time. However, the strength of influence of one concept over another may change in time as well. This fact represents the leading motivation for the research work directed to develop more adaptive framework of FCMs (based on NPN logic). Such more realistic FCM based model of system behavior should include preferential and time-dependent causal weights. For instance, engineering problems usually deal with measurement-based concepts, some of which are time-varying. But independently of their dynamics and direction of causal relationships, intensity of causal dependence may have its own dynamics too. This issue is recognized as the crucial one for adaptive and reliable behavior of decision making [5-8], [27], [29-31], control problems [9], [10] and agent systems [11]. Several approaches are proposed, which include modifications of the Pool2 algorithm [9], [10], [12], neural networks [13-15] and genetic algorithms [13], [16], [17].

This paper presents more adaptive application of FCMs based on NPN logic modeling framework. The approach itself combines NPN logic based FCMs with linguistic models of causality weights to achieve their quasi-dynamical adaptation to the change of measurement-based values of FCM concepts. Next section briefly reviews notion of NPN logic and NPN relations. The third section introduces fuzzy models of FCM causality weights and incorporates them in the framework of NPN logic based reasoning. Finally, a short discussion of presented approach is provided.

2. Theory of NPN logic and NPN relations

We recall that FCMs are signed, fuzzy weighted and directed graphs with feedback [2], [4]. The concept nodes C_i are fuzzy sets or even fuzzy systems. In general, FCM's concept nodes may stand for states, variables, events, actions, goals, values, trends of the system it models. The links, or edges, define rules or causal flows between the concept nodes. The modeling framework is based on determination of meaningful concepts, connecting them to form a network, and evaluating the direction of effect of target concept excited by the cause concept. In such networks (graphs) the directed link (edge) w_{ij}, from causal concept C_i to target (effect) concept C_j, measures how much C_i causes C_j. Connection n-by-n matrix W contains weights of all edges, representing weighted causation rules of system behavior. The edges w_{ij} take values in the fuzzy causal interval [-1, +1]. When FCM models a physical system (for instance, machining process planning, hydroelectric power station, scoliotic deformity, etc.) each concept node is characterized by a number in the interval [0, 1]. Such concept values are the result either of the normalization of the real value of the system's characteristics or membership degree of the real value (sensory readings, actual measurements) to the fuzzy set which describes the system's characteristics [18]. The latter holds for the concept nodes that represent fuzzy systems, as well. However, one of the most essential parts in a FCM modeling is determination of causal links between the nodes, including their strengths. One must have in mind that initially set weights, acquired in different ways (e.g. assessed by experts), may change and different weight sets can produce the same equilibrium situation [9], [29].

(F)CMs draw causal relationships between system's nodes, describing their mutual dependability and enabling *what-if* inference on a given situation. When value of one or more concept (reference, input) nodes changes, the map is excited and starts adjusting the values of all other nodes, according to the selected threshold function, until it settles down to equilibrium [4], [9], [10], [19]. This updating process uses both, concept values and causal weights.

In the real-world environments almost every decision has its consequences, presenting very valuable portion of information upon which we also make our decisions. FCM modeling framework involves negative edge weights as well, usually in multivalent [-1, +1] interval. In order to measure the magnitude of consequences we need more than three trivalent interval values {-1, 0, +1} of logic variables [1], [20]. NPN fuzzy logic theory is multi-valued logic based on six classes of values [3], [21]. Three individual classes assume values from [-1,0), {0}, and (0,+1] intervals, and three compound classes of values: (0, P) indicates there is no induced negative relationship and positive relationship has a strength P, (N, 0) indicates there is no induced positive relationship and negative relationship has a strength N, and (N, P) indicates that object *i* has both positive and negative relationships to object *j* with negative relationship of a strength N, and positive relationship of a strength P. The third compound value pair *(a, b)* is the most informational and fully describes the side effect, which measures under what mutual conditions between concepts FCM settles down in equilibrium.

Any NPN logic value can be represented as an ordered pair in [-1, 1] × [-1, 1]. The NEG, AND, and OR functions for both NPN crisp and fuzzy logics can be compactly described by the following three logic equations:

$$\mathrm{NEG}(x, y) = (\mathrm{NEG}(y), \mathrm{NEG}(x)) \quad , \tag{1}$$

$$\begin{gathered}(x, y) * (u, v) = (\min(x * u, x * v, y * u, y * v), \\ \max(x * u, x * v, y * u, y * v)) \quad ,\end{gathered} \tag{2}$$

$$(x, y)\ \mathrm{OR}\ (u, v) = (\min(x, u), \max(y, v)) \quad . \tag{3}$$

The star operator (∗) in (2) stands for a general conjunction operator that may be any *T*-norm extended from the interval [0, 1] to [-1, 1]. The extension is made as follows:

$$x * y = \mathrm{sign}(x)\,\mathrm{sign}(y)(|x| * |y|) \quad , \tag{4}$$

where *x* and *y* are singleton NPN values (fuzzy or crisp).

For the sake of briefness we will introduce the following definitions of NPN fuzzy relations, their transitivity and (heuristic) transitive closure, which play important role in reasoning with NPN relations, and skip some other formal definitions, which one can look for in [3], [22-25].

The following definition is an extension of classical fuzzy (binary) relation [22-25], which ensures assigning of NPN compound logic values to a NPN fuzzy (binary) relation as an ordered pair of negative, positive or neutral values:

> ***Definition:*** *An NPN fuzzy (binary) relation* $\mathcal{R}$ *in* $X \times Y$, *where* $X = \{x_i\}$ *and* $Y = \{y_j\}$ *are finite sets, is a collection of ordered pairs or a subset of* $X \times Y$ *characterized by a membership function* $\mu_{\mathcal{R}}(x_i, y_j)$ *that associates with each ordered pair* (x_i, y_j) *a strength of relation between* x_i *and* y_j *using an NPN fuzzy logic value.*

One of the very important sources of imprecision in complex systems is related to a transition behavior [4]. The effect of (imprecise) information propagation through a system may have significant influence on final decision-making, depending on weights of connections between concept nodes of a system's network. Next definition provides formal description of *max-∗* transitivity property of NPN relations:

> ***Definition:*** *An NPN relation* $\mathcal{R}$ *(crisp or fuzzy) in* $X \times X$, *where* $X = \{x_1, x_2, ..., x_n\}$ *is finite set, is NPN (*max-∗*) transitive iff, for all* i, j, *and* k, $0 < i, j, k \le n$,

$$\mu_{\mathcal{R}}(x_i, x_k) \ge \max_{x_j}(\mu_{\mathcal{R}}(x_i, x_j) * \mu_{\mathcal{R}}(x_j, x_k)) \ . \tag{5}$$

Since the connections between system's concepts can be established by different relations we need to compose two or more relations in order to model information propagation, in FCMs usually represented by a fuzzy chain [2], [22-24] , [26]. The (max-∗) composition of two NPN relations $\mathcal{R} \subseteq X \times Y$ and $\mathcal{Q} \subseteq Y \times Z$, denoted by $\mathcal{R} \circ \mathcal{Q}$, is defined by:

$$\mu_{\mathcal{R} \circ \mathcal{Q}} = \max_{y}(\mu_{\mathcal{R}}(x, y) * \mu_{\mathcal{Q}}(y, z)), \quad x \in X, y \in Y, z \in Z \quad , \tag{6}$$

and can be extended to n-fold composition denoted as $\mathcal{R}^n = \mathcal{R} \circ \mathcal{R} \circ \cdots \circ \mathcal{R}$.

> ***Definition:*** *The transitive closure* $\breve{\mathrm{R}}$ *of an NPN relation* $\mathcal{R}$ *(crisp or fuzzy) in* X*, is the smallest* (max-∗) *transitive NPN relation containing* $\mathcal{R}$ *. Since the NPN logics used for transitive closure computation can be considered as a set of rules (heuristics), such closure is called a* heuristic transitive closure (HTC) *of* $\mathcal{R}$ *.*

Using an heuristic path searching algorithm [3] we can find the possible and the most effective paths from one concept to another. That means, we can find the paths between elements (concept nodes) of FCM with the strongest negative and positive side effects that constrain decision making, according to the above two definitions.

3. Fuzzy Modeling of Causal Weights

When dealing with physical systems concepts' values usually are measurement-based. In certain cases experts rely on their experience and perception of current situation, assessing the concepts' values in imprecise manner (e.g. "approximately 5"). Also, some of precisely measured, obtained or assessed concepts values experts use to convert to less precise classification groups assigning to it descriptive degree of belonging. On the other side, the degrees of causal dependence between concepts in practice are almost with no exceptions qualitatively assessed. The meaning of the words of natural language used for degrees of causal relationships depend on context, domain, and nature of the related concepts. In the most cases just a few words are used to quantify a causal dependence (e.g., weak, medium, strong) and a couple of words for modifiers (e.g., very, extremely, a little, not_so, fairly).

Let $\mathcal{W}$ be a set of linguistic labels used for quantification of causal weights, namely $\mathcal{W} = (W_1, W_2, ..., W_k)$, defined over domain X, i.e., $W_i \in \mathcal{P}(X)$, $i \in \mathbf{N}_k$, where $\mathcal{P}(X)$ denote a power set of X. Each fuzzy set W_i, $i \in \mathbf{N}_k$ is defined by its membership function $\mu_W(x): X \to [0,1]$. Also, let $\mathcal{M} = (M_1, M_2, ..., M_r)$ be a set of modifiers, i.e. unary operators (acting on a fuzzy set, transforming a fuzzy set into another one in the same universe), which may modify linguistic weights $W_i \in \mathcal{P}(X)$, $i \in \mathbf{N}_k$ for each $x \in X$ by the equation:

$$^{M}W(x) = M(W(x)) \quad , \tag{7}$$

where $^{M}W \in \mathcal{P}(X)$ denotes linguistic value obtained by applying modifier M to the weight W. The set of all modified weights is denoted as $^{\mathcal{M}}\mathcal{W}$. Typically, modifier *"very"* is defined as $M(a) = a^2$, and *"a little"* or *"not_so"* or *"fairly"* as $M(a) = \sqrt{a}$, where $a \in [0,1]$. Of course, for such modifiers we can use other appropriate operators instead. Generally, modifier is called strong if $M(a) < a$, weak if $M(a) > a$, and identity modifier if $M(a) = a$, for $\forall a \in [0,1]$. Thus, each linguistic weight is a subset of $^{\mathcal{M}}\mathcal{W}$, i.e., $\mathcal{W} \subset {}^{\mathcal{M}}\mathcal{W}$, and has a following structure:

$$\mathcal{S} = \{W, M\} \quad . \tag{8}$$

We may assume that there exist a mapping between values of the concept C_i and the concept C_j. It may be a functional dependency between cause concept C_i and target concept C_j of the form $C_j=f(C_i)$. Function f may have very complex form, empirically defined or unknown. Also, it may be a relational dependency between cause concept C_i and target concept C_j of the form $C_j=\mathcal{R}(C_i)$, where relation $\mathcal{R}$ may be crisp or fuzzy. Experts usually do not provide the definition of such mapping, no matter whether they know it or not. Quite often they rely on recommendation, provided in the form of tables, and to the great extent on their personal experience and belief. Therefore, functional or relational dependence between concepts experts apt to causally describe using linguistic weights. Such linguistic weights are context dependent and often of quasi-dynamic nature. In some cases, while certain value resides within one particular interval of its domain expert will evaluate in one way, but he or she will use different quantifier when concept changes its state to another interval (Fig.1). That is, to achieve the preferred goals of the (physical) system, represented by the equilibrium point (state), causal weights need to be feasibly updated according to the function and operational characteristics of the system.

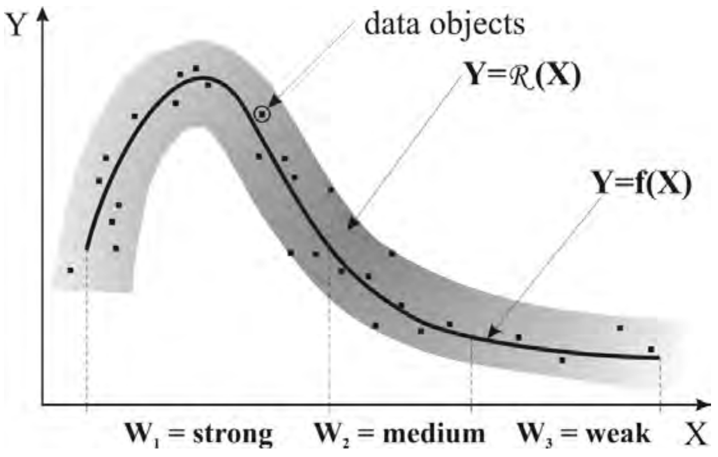


Fig. 1. Possible types of mappings between FCM concept nodes

When deal with measurement-based concept values we usually can gather enough data from direct measurements, archives or recommendation to create data patterns. Also, permissible sets of concepts' values are known. In order to establish fuzzy models of causal weights between concepts we are supposed to cluster given sets of data patterns into classes [25]. The number of classes should be the same as the number of basic linguistic quantifiers which experts use to evaluate system's causal relationships (Fig.1). If needed, basic linguistic quantifiers can be modified by appropriate modifier during any stage of decision analysis process. Thus, the number of clustering classes is $c = |\mathcal{W}|$, and the number of possible linguistic labels in a term set

T = $^{\mathcal{M}}\mathcal{W}$ is $t = |^{\mathcal{M}}\mathcal{W}|$. Schematically, (type-1) fuzzy model based causal effect of concept node C_i to concept node C_j is shown in Figure 2, for both increasing or positive (Fig.2(a)) and decreasing or negative (Fig.2(b)) causal relationship.

Total effect of concept C_i to concept C_j is achieved via all paths from C_i to C_j. The sequence of concepts $\mathcal{C} = (C_i, C_1, C_2, ..., C_j)$ is called chain, namely, NPN fuzzy chain (Fig.3). The strength of the chain is defined by $*$-composition of chain elements strength, i.e. by $*$-composition of pairs (C_k, C_{k+1}), $k \in \boldsymbol{N}_{n-1}$:

$$\mu_C(x) = \mu(C_1, C_2) * \mu(C_2, C_3) * \cdots * \mu(C_{k-1}, C_k) \quad . \tag{9}$$

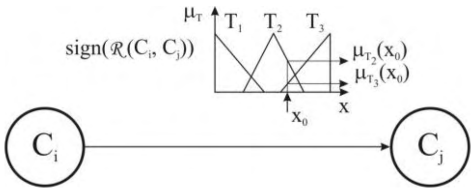


Fig. 2. Fuzzy model based causal weigh

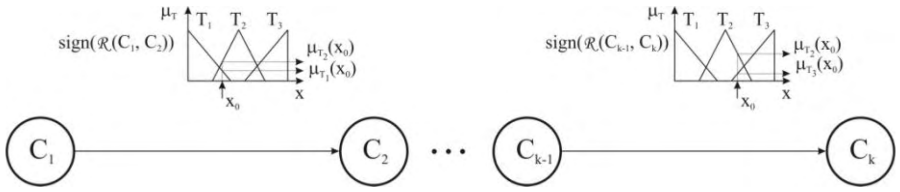


Fig. 3. Fuzzy model based NPN fuzzy chain

Since more than one chain (path) can be established between two nodes in FCM, in general case (Fig.4), for the most important, in the sense of the most causally effective path, we choose the strongest one defined by ***max***-$*$ composition (Eq.6).

The causal connection between concepts is established by fuzzy models, which are fuzzy rules. Therefore we apply the same reasoning mechanism as for SISO (single input, single output) systems [25], [28]. Each causal linguistically weighted link is the fuzzy rule of the form:

$$\begin{array}{lll} \text{IF} & C \text{ is } V & \\ \text{THEN} & {}^{\mathcal{M}}\mathcal{W} \text{ is } {}^{M}W & , \end{array} \tag{10}$$

where *C* is crisp variable represented by a FCM concept defined over domain $\mathcal{D}_C$, and *V* is a value (current state) of variable and $V \in \mathcal{D}_C$; $\mathcal{D}_C$ denotes a fuzzified domain obtained upon permissible set of concepts' real values. Rules (10) are relations $\mathcal{R}_k$, $k \in \mathbf{N}_q$ in a space $\mathcal{D}_C \times X$, defined as:

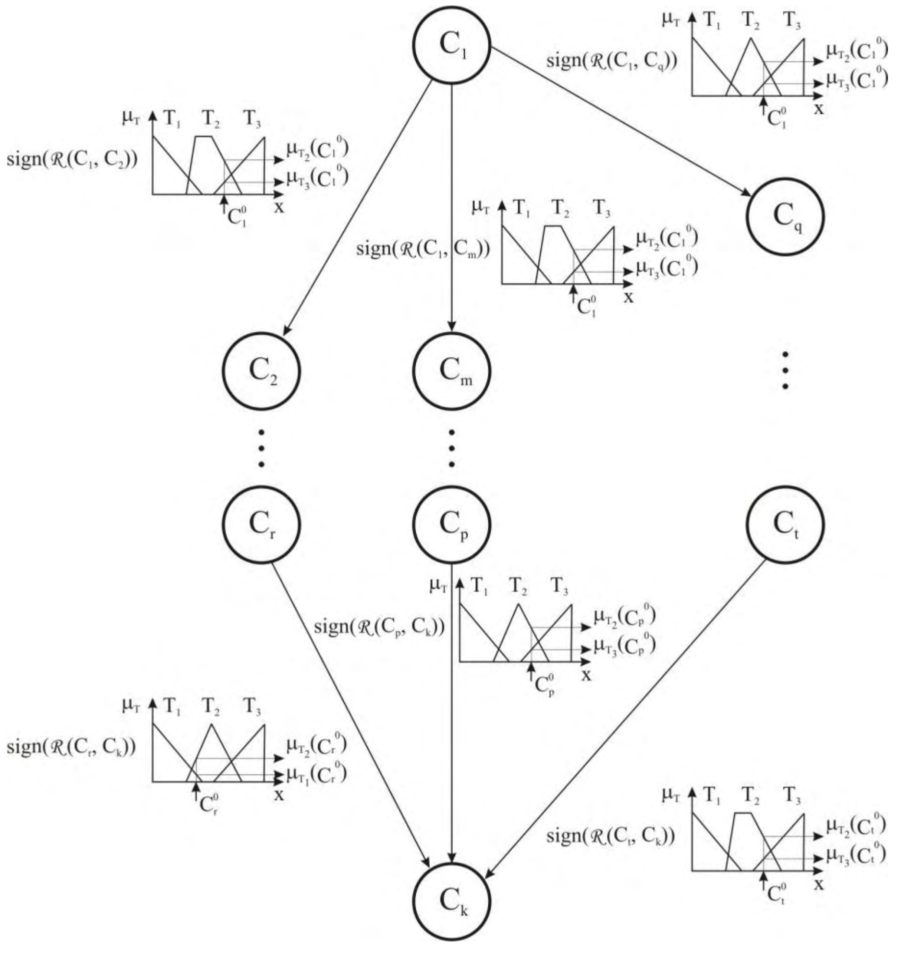


Fig. 4. Fuzzy model based NPN fuzzy multiple chains

$$\mathcal{R}_k(\mathcal{D}_C, X) = \{(V_k, {}^{M}W_k), V_k \in \mathcal{D}_C, {}^{M}W_k \in {}^{\mathcal{M}}\mathcal{W}\}, \tag{11}$$

and membership function

$$\mu_{\mathcal{R}_k}(v, x) = M(\mu_{W_k}(v, x)) \quad , \tag{12}$$

defines a strength of causal link, where *M* is modifying unary operator.

The strength of the NPN fuzzy chain (individual causal path *P*) defined by Eq.(9) is $*$-composition of partial relations (11):

$$\mathcal{R}^{(P)} = \mathop{*}_{k=1}^{q} \mathcal{R}_k \quad , \tag{13}$$

and corresponding membership function is:

$$\mu_{\mathcal{R}}^{(P)}(C_i, C_j) = \mathop{*}_{t} (\mu_{\mathcal{R}_k}(C_i, C_t), \mu_{\mathcal{R}_k}(C_t, C_j)) \quad . \tag{14}$$

The global relation $\mathcal{R}$ represents the strongest individual causal path of concept C_i to concept C_j defined by ***max***-composition (denoted by $\cup$):

$$\mathcal{R} = \bigcup_{j=1}^{r} \mathcal{R}_j^{(P)} \quad , \tag{15}$$

of the total strength:

$$\mu_{\mathcal{R}}(C_i, C_j) = \bigcup_{k=1}^{m} \mu_{\mathcal{R}_k}^{(P)}(C_i, C_j) \quad . \tag{16}$$

As the result of fuzzy relation $\mathcal{R}$ we obtain the lower and the upper bound value of NPN logic value pair *(a, b)*, which count side-effect that defines conditions of equilibrium in a system.

Illustrative example

Let's suppose that a physical system is modeled by the FCM shown in Figure 5. Its concept nodes represent distinctive measurement-based characteristics. Also, mutual dependence between two causal nodes may be of different nature, assuming that representational schemes include functional, relational and empirical modes. Consequently, fuzzification algorithms may vary, respecting the basic requirement to produce a term set of a linguistic causal weights, in this case $\mathcal{W}$ = *(Low, Medium, High)*. We adopt the following set of modifiers: $\mathcal{M}$ = *(Very, Fairly,* $\mathcal{I}$*)*, with usual definitions for modifiers *Very* and *Fairly*, where *Very(a)* = a^2 and *Fairly(a)* = $\sqrt{a}$, *a* $\in$ [0,1], and $\mathcal{I}$ represents identity modifier $\mathcal{I}$*(a)* = *a*, for $\forall a \in$ [0,1]. The corresponding term set is T = $\mathcal{M}\,\mathcal{W}$. For the sake of simplicity, without the loss of generality, the most of the causal weights in the Figure 5 are shown as resulting membership grades, rather than fuzzy models.

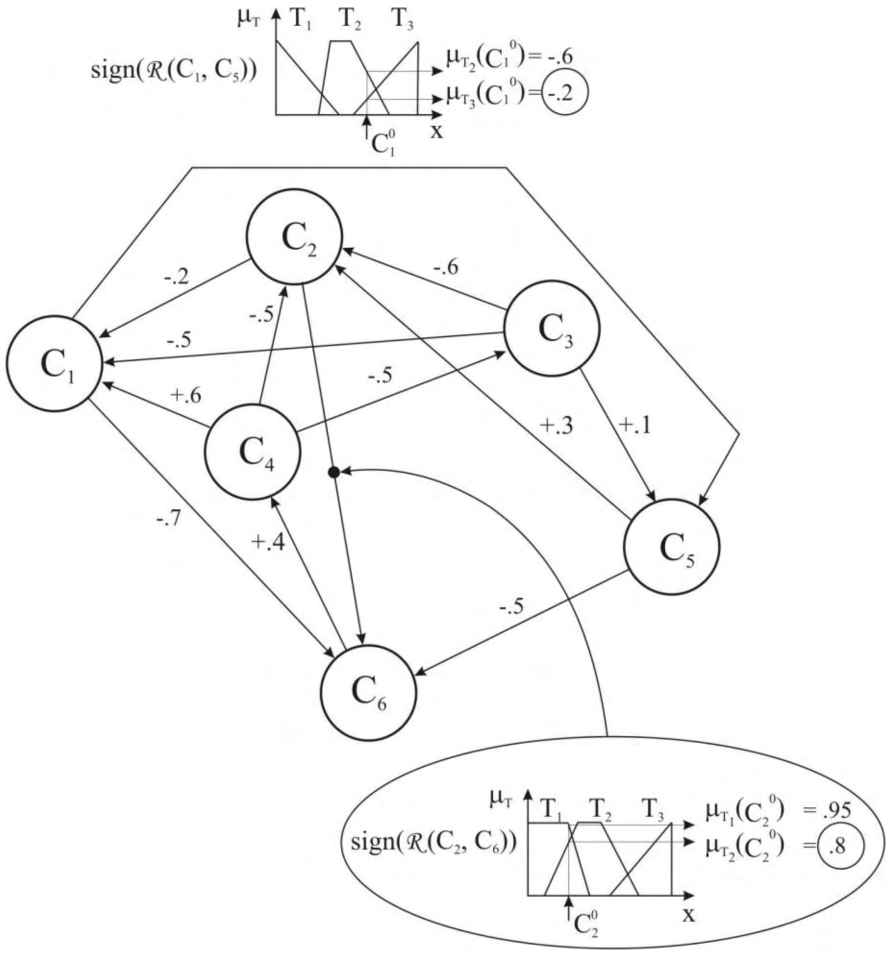


Fig. 5. NPN FCM with linguistic causal weights assessment

Also, let concept node C_1 be a reference (input) node, and concept node C_2 be an output node. Assuming that we are interesting in making a decision on control action over concept node C_2 when excite concept node C_1, we start "what-if" analysis by identifying the fuzzy chains which describe the possible paths from C_1 to C_2 using heuristic path searching algorithm [3]. Among all these paths we calculate the most effective one that provides the largest side effect by applying Eqs.(11)-(16). For the given physical system corresponding connection matrix is:

In this case we have chosen *max-prod* (max-dot) transitivity composition for the output node, defined by Eqs. (5) and (6). Identified the most effective heuristic paths are shown in the Table 1 and the compound values of heuristic transitive *max-prod* closure are shown in the Table 2. If we are interested how to most effectively increase the value (i.e. to perform a control or decision action) of concept node C_2, which can be thought as increasing the speed or temperature, then we conclude that the most effective is the path 1-6-4-2, which provides the highest positive compound value 0.140, with the strongest side effect -0.084, caused through the chain 1-6-4-3-2.

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & -0.2 & -0.7 \\ -0.2 & 0 & 0 & 0 & 0 & 0.8 \\ -0.5 & -0.6 & 0 & 0 & 0.1 & 0 \\ 0.6 & -0.5 & -0.5 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 0.4 & 0 & 0 \end{pmatrix} . \qquad (17)$$

The change of the initial concept nodes' values, affects causal weights to modify according to their mutual physical dependency. Such changes may generate new causal weights using the approach described above. That is, connection matrix (17) changes using the set of modifiers $\mathcal{M}$, as presented by matrix (18). The most effective heuristic paths changed, as well as the compound values of heuristic transitive *max-prod* closure (Tables 3 and 4).

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & -0.2 & -Fairly(0.7) = -0.84 \\ -Very(0.2) = -0.04 & 0 & 0 & 0 & 0 & 0.8 \\ -0.5 & -Very(0.6) = -0.36 & 0 & 0 & 0.1 & 0 \\ 0.6 & -0.5 & -0.5 & 0 & 0 & 0 \\ 0 & Fairly(0.3) = 0.55 & 0 & 0 & 0 & -Fairly(0.5) = -0.71 \\ 0 & 0 & 0 & Fairly(0.4) = 0.63 & 0 & 0 \end{pmatrix} . \qquad (18)$$

Now we can notice that performing a control or decision action over the concept node C_2 is again the most effective through the path 1-6-4-2, but with increased overall benefits. The positive compound value is increased for almost 90% and reached 0.265, while the side effect, increased for about 31% at level of -0.110, is caused through the another path, 1-5-2. In the first setting of the hypothetical physical system the desired effect is caused before the side effect, since the side effect chain is longer, but in the modified setting the side effect is caused before the desired effect, for the same reason.

Table 1: The most effective heuristic paths (initial system's setting)

	1		2		3		4		5		6	
	N	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**
1	1-6-4-1	1-6-4-2-6-4-1	1-6-4-3-2	1-6-4-2	1-6-4-2-6-4-3	1-6-4-3	1-6-4	1-5-6-4	1-5	1-6-4-3-5	1-6	1-5-6
2	2-1	2-6-4-1	2-1-6-4-2	2-1-6-4-3-2	2-6-4-3	2-1-5-6-4-3	2-1-5-6-4	2-6-4	2-6-4-3-5	2-1-5	2-1-5-6	2-6
3	3-2-1	3-1	3-2	3-5-2	3-1-6-4-3	3-1-6-4-2-6-4-3	3-1-5-6-4	3-1-6-4	3-1-6-4-1-5	3-5	3-5-6	3-1-6
4	4-3-2-1	4-1	4-2	4-3-2	4-3	-	4-1-6-4	4-1-5-6-4	4-1-5	4-3-2-1-5	4-1-6	4-1-5-6
5	5-2-1	5-2-6-4-1	5-6-4-3-2	5-2	5-2-6-4-3	5-6-4-3	5-6-4	5-2-6-4	5-2-6-4-1-5	5-6-4-1-5	5-6	5-2-6
6	6-4-3-2-1	6-4-1	6-4-2	6-4-3-2	6-4-3	-	-	6-4	6-4-1-5	6-4-3-2-1-5	6-4-1-6	6-4-1-5-6

Table 2: Compound values of heuristic transitive max - prod closure (initial system's setting)

	1		2		3		4		5		6	
	N	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**	**N**	**P**
1	-0.168	0.027	-0.084	0.140	-0.022	0.140	-0.280	0.040	-0.200	0.014	-0.700	0.100
2	-0.200	0.192	-0.028	0.017	-0.160	0.004	-0.008	0.320	-0.016	0.040	-0.020	0.800
3	-0.500	0.120	-0.600	0.030	-0.070	0.011	-0.020	0.140	-0.017	0.100	-0.050	0.350
4	-0.060	0.600	-0.500	0.300	-0.500	-	-0.168	0.024	-0.120	0.012	-0.420	0.060
5	-0.060	0.058	-0.060	0.300	-0.048	0.100	-0.200	0.096	-0.012	0.024	-0.500	0.240
6	-0.024	0.240	-0.200	0.120	-0.200	-	-	0.400	-0.048	0.005	-0.168	0.024

Table 3: The most effective heuristic paths (modified system's setting)

	1		2		3		4		5		6	
	N	P	N	P	N	P	N	P	N	P	N	P
1	1-6-4-1	1-6-4-2-6-4-1	1-5-2	1-6-4-2	1-6-4-2-6-4-3	1-6-4-3	1-6-4	1-5-6-4	1-5	1-6-4-3-5	1-6	1-5-6
2	2-1	2-6-4-1	2-1-6-4-2	2-1-6-4-3-2	2-6-4-3	2-1-5-6-4-3	2-1-5-6-4	2-6-4	2-6-4-3-5	2-1-5	2-1-5-6	2-6
3	3-2-1	3-5-2-6-4-1	3-2	3-5-2	3-1-6-4-3	3-1-6-4-2-6-4-3	3-1-5-6-4	3-1-6-4	3-1-6-4-1-5	3-5	3-5-6	3-1-6
4	4-3-2-1	4-1	4-2	4-3-2	4-3	-	4-1-6-4	4-1-5-6-4	4-1-5	4-3-2-1-5	4-1-6	4-1-5-6
5	5-2-1	5-2-6-4-1	5-6-4-3-2	5-2	5-2-6-4-3	5-6-4-3	5-6-4	5-2-6-4	5-2-6-4-1-5	5-6-4-1-5	5-6	5-2-6
6	6-4-3-2-1	6-4-1	6-4-2	6-4-3-2	6-4-3	-	-	6-4	6-4-1-5	6-4-3-2-1-5	6-4-1-6	6-4-1-5-6

Table 4: Compound values of heuristic transitive max - prod closure (modified system's setting)

	1		2		3		4		5		6	
	N	P	N	P	N	P	N	P	N	P	N	P
1	-0.318	0.080	-0.110	0.265	-0.067	0.265	-0.529	0.089	-0.200	0.026	-0.840	0.142
2	-0.040	0.302	-0.011	0.004	-0.252	0.002	-0.004	0.504	-0.025	0.008	-0.006	0.800
3	-0.500	0.017	-0.360	0.055	-0.132	0.033	-0.045	0.265	-0.032	0.100	-0.071	0.420
4	-0.007	0.600	-0.500	0.180	-0.500	-	-0.318	0.054	-0.120	0.001	-0.504	0.085
5	-0.022	0.166	-0.081	0.550	-0.139	0.224	-0.447	0.277	-0.033	0.054	-0.710	0.440
6	-0.005	0.378	-0.315	0.113	-0.315	-	-	0.630	-0.076	0.001	-0.318	0.054

4. Conclusions

We have presented some of preliminary results of the research work related to the adaptive causality weights assessment. It is based on application of linguistic preferences and their fuzzy models. In this approach system behavior analysis uses linguistic causality weights constructed upon measurement-based concepts values and their known dependences. This way causality weights are assessed indirectly, activating fuzzy models by current state concepts values. In this approach the nature of concept dependency is captured by clustering measurement-based data into appropriate groups. To each group we assign a label of expert's linguistic preference used to describe the system behavior. Linguistic models quasi-dynamically tune the causation weights and allow for their propagation through NPN logic fuzzy chains. The lower and upper bound of NPN logic compound values enable measurement of side effect, which in turn provides the basis for deeper understanding of system behavior and more reliable decision analysis.

Acknowledgement

Research work presented in the paper is funded by the Serbian Ministry of Science within the projects TR-12002 and TR-12010.

References

1. R. Axelrod: "Structure of Decision: The Cognitive Maps of Political Elites", Princeton University Press, Princeton NJ, 1976.
2. B. Kosko: "Fuzzy Cognitive Maps", Int. J. of Man–Machine Studies, Vol.24, pp.65-75, 1986.
3. W.R. Zhang, S.S. Chen, J.C. Bezdek: "Pool2: A Generic System for Cognitive Map Development and Decision Analysis", IEEE-SMC, Vol.19, No.1, pp.31-39, 1989.
4. B. Kosko: "Neural Networks and Fuzzy Systems - A Dynamical Systems Approach to Machine Intelligence", Prentice Hall, Englewood Cliffs, New Jersey, 1992.
5. F. Herrera, E. Herrera-Viedma: " Linguistic decision analysis: steps for solving decision problems under linguistic information", Fuzzy Sets and Systems, Vol.115, pp. 67-82, 2000.
6. F. Herrera, E. Herrera-Viedma, Luis Martínez: " A fusion approach for managing multi-granularity linguistic term sets in decision making", Fuzzy Sets and Systems, Vol.114, pp. 43-58, 2000.

7. M. Delgado, F. Herrera, E. Herrera-Viedma, L. Martínez: "Combining numerical and linguistic information in group decision making", Journal of Information Sciences, Vol.107, pp. 177-194, 1998.
8. R.R. Yager: "On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking", IEEE Transactions on Systems, Man, and Cybernetics", Vol.18, No.1, pp.183-190, 1988.
9. T.L. Kottas, Y.S. Boutalis and M.A. Christodoulou, Fuzzy Cognitive Networks: A General Framework``, Inteligent Desicion Technologies, vol. 1, no. 4, 2007, pp. 183- 196.
10. Y. Boutalis, T. Kottas and M. Christodoulou, ``Adaptive Estimation of Fuzzy Cognitive Maps With Proven Stability and Parameter Convergence", IEEE Transactions on Fuzzy Systems, 2009, doi: 10.1109TFUZZ.2009.2017519.
11. C.Y. Miao, A. Goh, Y. Miao, Z.H. Yang: "Agent that Models, Reasons and Makes Decisions", Knowledge-Based Systems, Vol.15, pp.203-211, 2002.
12. T.L. Kottas, Y.S. Boutalis, G. Devedžić, B.G. Mertzios: "A new method for reaching equilibrium points in Fuzzy Cognitive Maps", IEEE Conference on Intelligent Systems, Varna Bulgaria, June 2004.
13. M. Ghazanfari, S. Alizadeh, M. Fathian, D.E. Koulouriotis: "Comparing simulated annealing and genetic algorithm in learning FCM", Applied Mathematics and Computation, Vol.192, Issue 1, pp.56-68, 2007.
14. E.I. Papageorgiou, C.D. Stylios, P.P. Groumpos: "Active Hebbian learning algorithm to train fuzzy cognitive maps", International Journal of Approximate Reasoning, Vol.37, pp.219–249, 2004.
15. E.I. Papageorgiou, C.D. Stylios, P.P. Groumpos: "Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links", International Journal of Human-Computer Studies, Vol.64, pp.727–743, 2006.
16. E.I. Papageorgiou, P.P. Groumpos: "A new hybrid method using evolutionary algorithms to train Fuzzy Cognitive Maps", Applied Soft Computing, Vol.5, pp.409–431, 2005.
17. W. Stach, L. Kurgan, W. Pedrycz, M. Reformat: "Genetic learning of fuzzy cognitive maps", Fuzzy Sets and Systems, Vol.153, pp.371–401, 2005.
18. M. Schneider, E. Schneider, A. Kandel, G. Chew: "Automatic Construction of FCMs", Fuzzy Sets and Systems, Vol.93, pp.161-172, 1998.
19. B. Kosko: "Fuzzy Engineering", Prentice Hall International, Inc., Upper Saddle River, New Jersey, 1997.
20. K. Nakamura, S. Iwai, T. Sawaragi: "Decision Support Using Causation Knowledge Base", IEEE-SMC, Vol.12, No.6, pp.765-777, 1982.
21. W.R. Zhang, S.S. Chen, W. Wang, R.S. King: "A Cognitive Map Based Approach to the Coordination of Distributed Cooperative Agents", IEEE-SMC, Vol.22, pp.103-114, 1992.
22. L.A. Zadeh: “Fuzzy Sets”, ", Information and Control, Vol.8, pp. 338-353, 1965.
23. L.A. Zadeh: “Similarity Relations and Fuzzy Orderings”, Information Sciences, Vol.3, pp.177-200, 1971.
24. L.A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning. Information Sciences, vols. 8, 8, 9 (Part I–III) (1975):199–249, 301–357, 43–80.
25. G. Klir, B. Yuan: “Fuzzy Sets and Fuzzy Logic – Theory and Applications”, Prentice Hall PTR, Upper Saddle River, New Jersey, U.S.A., 1995.
26. Kandel, L. Yelowitz: “Fuzzy Chains”, IEEE-SMC, Vol.4, No.5, pp.472-475, 1974.
27. G. Devedžić: “Fuzzy Cognitive Maps Based Decision Support System”, EUROFUSE 2001, Granada, Spain, April 25 - 27, 2001.

28. Y. Miao, Z.-Q. Liu: "On Causal Inference in Fuzzy Cognitive Maps", IEEE Transactions on Fuzzy Systems, Vol.8, No.1, pp.107-119, 2000.
29. K.-M. Osei-Bryson: "Generating consistent subjective estimates of the magnitudes of causal relationships in fuzzy cognitive maps", Computers & Operations Research, Vol.31, pp.165–1175, 2004.
30. G. Montibeller, V. Belton: "Qualitative operators for reasoning maps: Evaluating multi-criteria options with networks of reasons", European Journal of Operational Research, Volume 195, Issue 3, pp.829-840, 2009.
31. S.-M. Zhou, F. Chiclana, R.I. John, J.M. Garibaldi: "Type-1 OWA operators for aggregating uncertain information with uncertain weights induced by type-2 linguistic quantifiers", Fuzzy Sets and Systems, Vol.159, pp.3281–3296, 2008.

Goran Devedžić is professor at Faculty of Mechanical Engineering, University of Kragujevac, Serbia. His research interests focus on the advanced product and process development, industrial and medical application of soft computing techniques, and bioengineering. He has authored/co-authored more than 100 research papers, published in international and national journals or presented at international and national conferences, as well as three books on CAD/CAM technology and 3D product modeling.

Danijela Milošević is assistant professor at Technical Faculty at Čačak, University of Kragujevac, Serbia. She holds a PhD in Computer Science and her research interests include ontologies, knowledge representation, intelligent tutoring systems, user modeling and bioengineering. Author/co-author of over 70 research papers, two textbooks, and was actively involved in a number of research projects addressing e-Learning tools and technologies, adaptive instructional design and development of user model ontologies. She is a member of IEEE computer society.

Lozica Ivanović is assistant professor at Faculty of Mechanical Engineering, University of Kragujevac, Serbia. Her main research interests include industrial design and product alternatives ranking. She has authored/co-authored more than 20 research papers, published in international and national journals or presented at international and national conferences.

Dragan Adamović is associate professor at Faculty of Mechanical Engineering, University of Kragujevac, Serbia. His research interests include engineering materials, manufacturing parameters selection, and alternatives ranking. She has authored/co-authored more than 70 research papers, published in international and national journals or presented at international and national conferences, as well as eight books on engineering materials and welding.

Miodrag Manić is professor at Faculty of Mechanical Engineering, University of Niš, Serbia. His research interests focus on the manufacturing processes, production systems, and industrial application of soft computing techniques. He has authored/co-authored more than 100 research papers, published in international and national journals or presented at international and national conferences, as well as two books on CIM systems and CNC technology.

Received: February 23, 2009; Accepted: November 27, 2009.

UDC 621.391:004.4, DOI:10.2298/CSIS090710007B

QoS Testing In a Live Private IP MPLS Network with CoS Implemented

Živko Bojović[1], Emil Šećerov[2], and Vlado Delić[2]

[1]Telekom Srbija, Takovska 2, 11000 Beograd, Serbia
zivko@telekom.rs
[2]Faculty of Technical Sciences, Trg D. Obradovića 6
21000 Novi Sad, Serbia
{secerov, tlk_delic}@uns.ac.rs

Abstract. This paper describes a testing conducted on a private IP/MPLS network of a Telecom operator during service introduction. We have applied DiffServ and E-LSP policies for bandwidth allocation for predefined classes of service (voice, video, data and VPN). We used a traffic generator to create the worst possible situations during the testing, and measured QoS for individual services. UML considerations about NGN structure and packet networks traffic testing are also presented using the deployment, class and state diagrams. Testing results are given in tabular and graphical forms, and the conclusions derived will be subsequently used as a basis for defining the stochastic traffic generator/simulator.

Keywords: Network testing; IP/MPLS; DiffServ; Traffic generator; Worst case; UML.

1. Introduction

Development of telecommunication market and technologies caused transformation of networks specialized for dedicated service into multiservice networks (voice, video, data, L2 and L3 virtual private networks-VPNs) across the common infrastructure and unique service handling. Decisive influence to this transformation had the implementation of new technologies in the networks of telecommunication operators (see Fig. 1) and adoption of internet protocol (IP) core in next generation networks (NGN).

Real-time network services, e.g. telephony, video or multimedia service, which will prevail in the near future, are very dependent on variation of quality of service (QoS) parameters (delay, jitter and packet loss). The NGN convergence toward an unique network architecture offers capability of a flexible, on-demand and dynamic, bandwidth allocation for specific service classes. This fact means that there is no need to oversize bandwidth allocated for voice and video traffic, while the data service bandwidth suffers from deficiency of network resources [1].

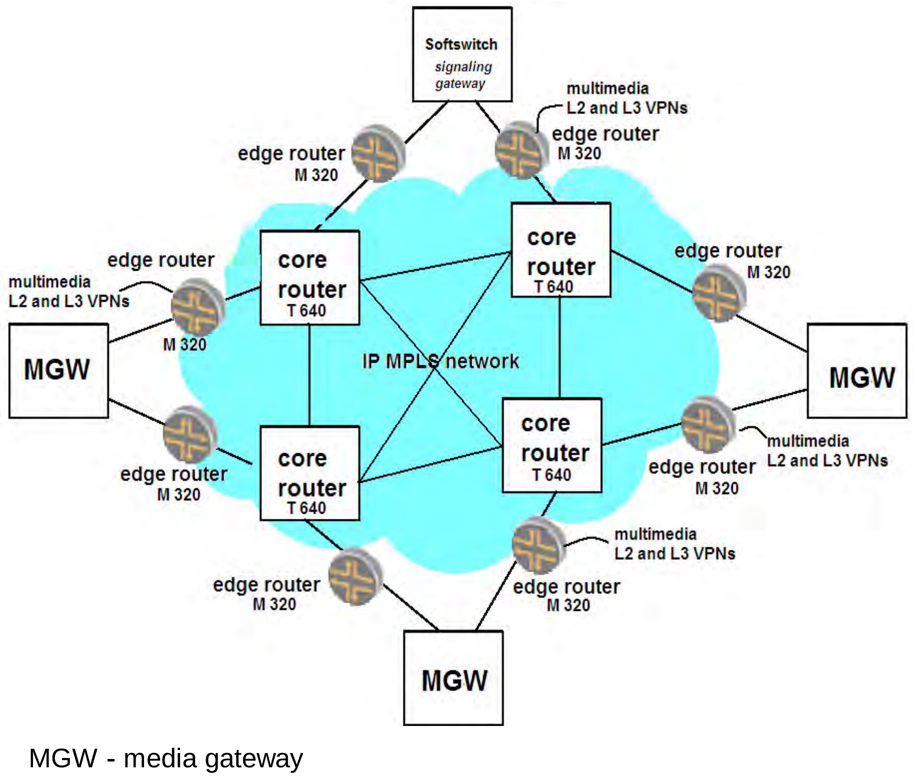


Fig. 1. An example of NGN implementation

QoS and costs are determining factors during network design and implementation, implying that desired QoS should be obtained at minimum possible costs. Per service costs should be optimized according to the usage of network resources. Specific service QoS, which network operators offer to the end users, is the basis for Service Level Agreement (SLA) contracts between the operators and end users. In actual IP networks, the prerequisite for QoS policy is implementation of Multi Protocol Label Switching (MPLS) mechanism for bandwidth allocation. Commonly used MPLS supplied with Traffic Engineering (TE) extension enable precise control, including the bandwidth allocation, of network established information paths [2]. MPLS bandwidth control mechanism prevents traffic overload condition that leads toward increasing the delay, jitter and packet loss. TE MPLS extension efficiently uses network resources and supports regular and balanced per service bandwidth allocation. Efficient, on demand and dynamic, bandwidth usage anticipates and prevents from network overload caused by burst nature of NGN traffic. Finally, it also can guarantee QoS agreed by SLA.

However, standard MPLS-TE does not reserve per service class bandwidth. Bandwidth allocation policy is used for aggregated services traffic flows. Since that network operators commonly use IETF (Internet Engineering

Task Force) Differentiated Service (DiffServ) standard that supports much wider Class of Services (CoS) than MPLS-TE [3]. MPLS DiffServ–TE combine advantages of both MPLS extensions (DiffServ and TE) to achieve strict QoS guaranties while optimizing the network resources. MPLS DiffServ-TE aware the network with CoS configuration enabling per service class resource reservation, service granularity for end user and QoS guaranties defined in SLA.

Today, Class of Service support has become an indispensable function in many of the large service provider networks because of the competitive nature of the Internet and the diversity in customer needs. CoS support in traditional IP routed cores has been provided by a variety of queuing and scheduling mechanisms. This paper presents some experience in testing CoS translation in IP/MPLS based packet network. Effects of relative and fixed bandwidth allocation for different CoS in the bursty traffic conditions are discussed based on experimental results. Paper is organized as follows. Section 2 gives short overview of related work in the area of telecommunication traffic and sources for the proposed UML (Unified Modeling Language) view of the network. Section 3 is the continuation of the introduction and presents an overview UML specification of NGN. Section 4 outlines some of the techniques put forth by the IP network to MPLS CoS mapping over arbitrary layer-2 technology. Sections 5 describe the traffic classes in the network that is the subject of the experiment. Section 6 describes the test environment, the tools used in conducting the experiments and shows the results of experiments. It also gives the UML view of test network topology and state diagram for statistical MUX that is important for understanding the dynamics of packet networks scheduling process. Section 7 is the conclusion with some final comments.

2. Related Work

The first papers in this field of IP communication were written with the aim to improve the knowledge of MPLS concepts, installation, migration, operation, inspection and troubleshooting in accordance with applicable RFC documents [1], [2]. The effort was made to make a general review of MPLS and to explain its functioning in telecommunication networks. In other words, the intention was to point out all aspects of implementation, integration and installation of MPLS infrastructure in the networks of telecom operators. The next group of papers was based on implementation of Quality of Service mechanisms as a critical element to solving the problems of handling the increasing volume and disparate types of traffic seen on today's public and private networks [3]. Whenever a mission critical application - whether voice, video, or data - is being delivered over a network, Quality of Service traffic shaping and policing can assist network managers in maintaining the best network performance and good-put for applications during periods of network congestion [7].

As far as we know in this moment there are no available papers or other sources of knowledge that propose UML view for the telecommunication networks structure and behavior. According to this we are trying to initiate work in this ICT(Information and Communication Technologies) convergence field and the main UML sources for this paper were [4], [5] and [6].

QoS offers the following core principles: Classification and Marking, Policing, Queuing and Dropping. Class of Service is Quality of Service method used to manage the network during congestion as implemented in Junos (Juniper Operating System) software [8]. It allows the network administrator to identify the order of priority that the incoming traffic should be processed for transmission when the switch or network is congested. As an example, a weighted round robin (WRR) method prevents head-of-line blocking. Juniper routers, with implemented Junos, are reliable, high-performance network operating system for routing, switching and security features [9]. Starting from the previously mentioned, the desire of the authors was that, in the vibrant MPLS network with a specific configuration and implemented in the Junos software within the JUNIPER routers, examine the behavior of these networks in some critical cases. In that sense, we made the test scenarios whose contents and results will be listed below the text block.

3. UML Specification of NGN

In the figure 2 there are two compartments: above is the provider core network (CN), while below is access network (AN). In the CN there are four types of nodes: Media Gateway (MGW), Edge Router, Core Router and Softswitch. MGW is basically realized in hardware and its purpose is media traffic conversion and maintaining connections (between two users or conference call). It supports different media types for voice (PCM, ADPCM, CELP, etc.) and video (MPEG2, MPEG4, etc.) and has the access functionality. It also collects different user traffic and signaling from internet data traffic (WEB, ftp, e-mail, etc.), IP phones, IPTV and circuit switched network (analog, ISDN-Integrated Services Digital Network and GSM-Global System for Mobile-phones). Edge router is the IP network front end communication node used to shape users traffic demands. Core router is traffic switch whose throughput is determined with provider traffic forecast and Qos guarantied by the network. Softswitch is common name for access node of pure IP users. Although in the figure 2 Softswitch is represented as a single node it is typically realized as several hardware and software building blocks. Communication paths between MGW, Edge router and Softswitch are IP/gigabit Ethernet, where several routers can be connected to single MGW. Edge router is also connected to at least two Core routers. With this redundant connectivity two engineering goals are achieved: reliability and load sharing. For the purpose of maximal reliability core routers are fully connected, that is marked with cardinality “all” in the figure 2. Access network supports all of the today available user devices, but it is expected that IP access will prevail in the near future.

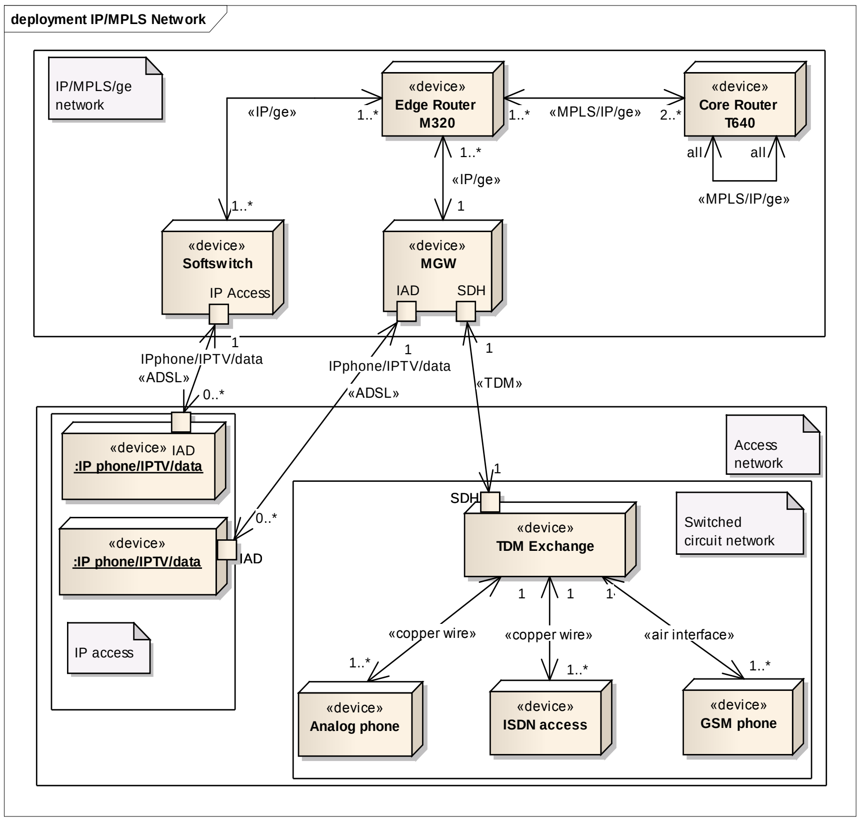


Fig. 2. Deployment diagram of the considered IP/MPLS network topology

Figure 3 shows the M320 Edge Router Device with it components and gigabit Ethernet Port. Software heart is Scheduler (IP packets scheduler), that relies on Junos and MPLS/IP networking components, realizes selected scheduler algorithm (e.g. deficit round robin) supplied with user defined parameters in SchedulerPolicy table. This table, as shown in fig. 3, has only rudimentary data, while in the real application besides the scheduling queues (BE_Queue, etc.), CoS parameters (latency, packet loss, bandwidth parameters and time constraint) should be defined.

Figure 4 is the deployment diagram for the media Gateway. Components that converts user TDM (Time Division Multiplex) to IP traffic are organized around Packetizing module, while SignalingConversion component converts SS7 (Signaling System No. 7) into SIP (Session Initiation Protocol) signaling. IP streaming component supports different media types for data, voice and video IP traffic. The communication paths between Scheduler, gigabit

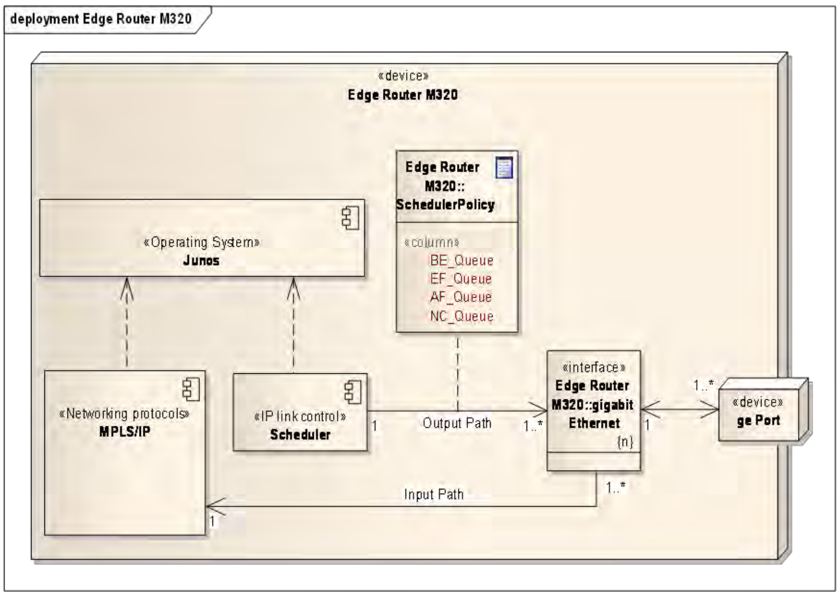


Fig. 3. Deployment diagram for the Edge router

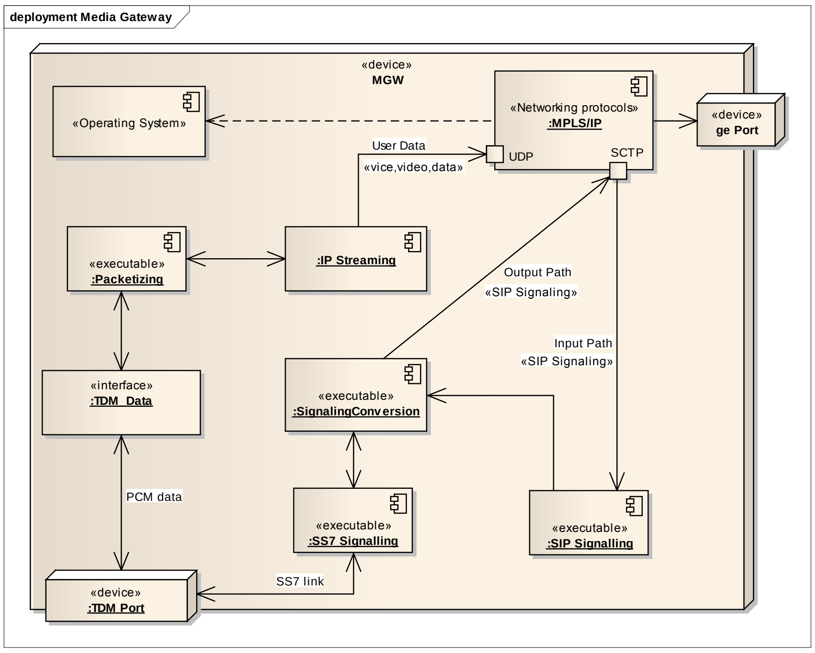


Fig. 4. Deployment diag ram for the media gateway

Ethernet interface and IP streaming should be consider in details for more realistic operating system component meaning that media gateways can be purchased from different manufactures. That is opposite to M320 and T640 routers in figure 3, where Junipers routers were assumed. Connectivity between devices from different manufacturers should be guaranteed by open standards for Ethernet, MPLS/IP and signaling protocols.

4. CoS Policy Implementation

One of the basic trends in communications is network and service convergence, i.e. a unique network architecture that simultaneously offers different types of services: voice, video, data, VPNs and others. The different types of service need different QoS requirements considering delay, jitter, and packet loss. In order to provide quality of service from end to end, providers usually apply model based upon IETF DiffServ standard, since it supports a vast number of classes of service [7]. For this reason, the CoS mechanisms are implemented into routers, implying the following steps (Fig. 5):

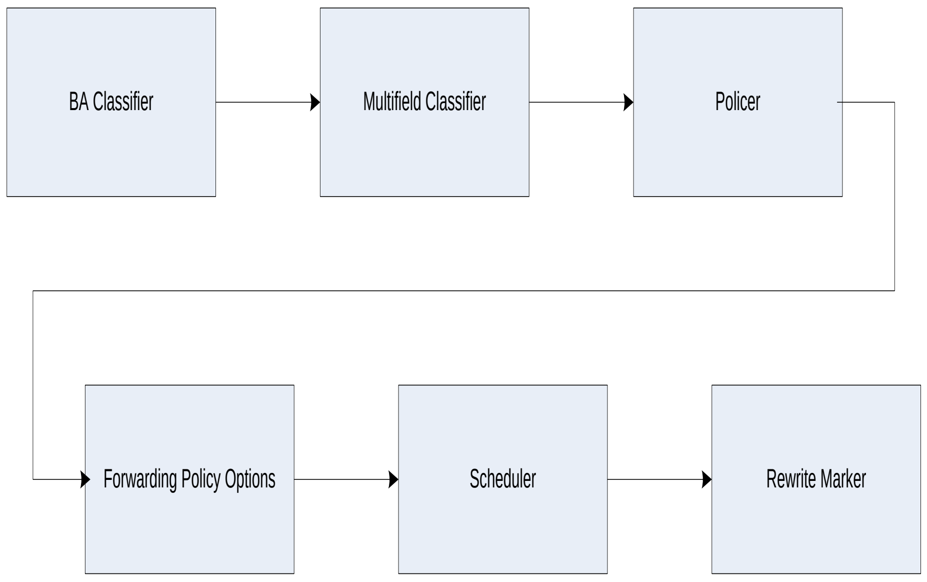


Fig. 5. The steps in implementation of CoS mechanism

packet classification (*Behavior Aggregate and Multifield*) at input interface;
policing application for the traffic flow (*Policer*);
policing application for the traffic transmission according to the CoS attributes (*Forwarding Class Policer*);

scheduling the transmission at the output interface (*Scheduler*);
marking the bits at the output interface (*Rewrite Marker*).

Figure 6 is the black box approach for deployed routers combined with some assumption details about QoS policy implementation. It extends Scheduler stub from figure 3 and it is based on CoS steps shown in figure 5. Input port (Rx IP Driver) is at the left top corner and Output port (Tx IP Driver) is in the down right corner suggesting MPLS/IP packets traverse through the scheduler modules/classes. There is one input queue and four output queues for each CoS, aggregated in OutputTrafficQueue. Traffic scheduler is an active class (thread) activated in router power on procedure. It uses other classes (BE and Multifield Classification, Policer and RewriteMaker) in successive packet scheduling steps. Classification of packets has two stages: BA (Behaviour Aggregate) and Multifield Classification. Data part in figure 6 is defined in CoS_Attributes and RoutingTable. Although it is not shown in the figure 6, to avoid overpolution with shapes, CoS_Attributes class should be instantiated for each of the traffic class (EF, AF, NC and BE). RoutingTable data depends on routing algorithm and actual traffic paths.

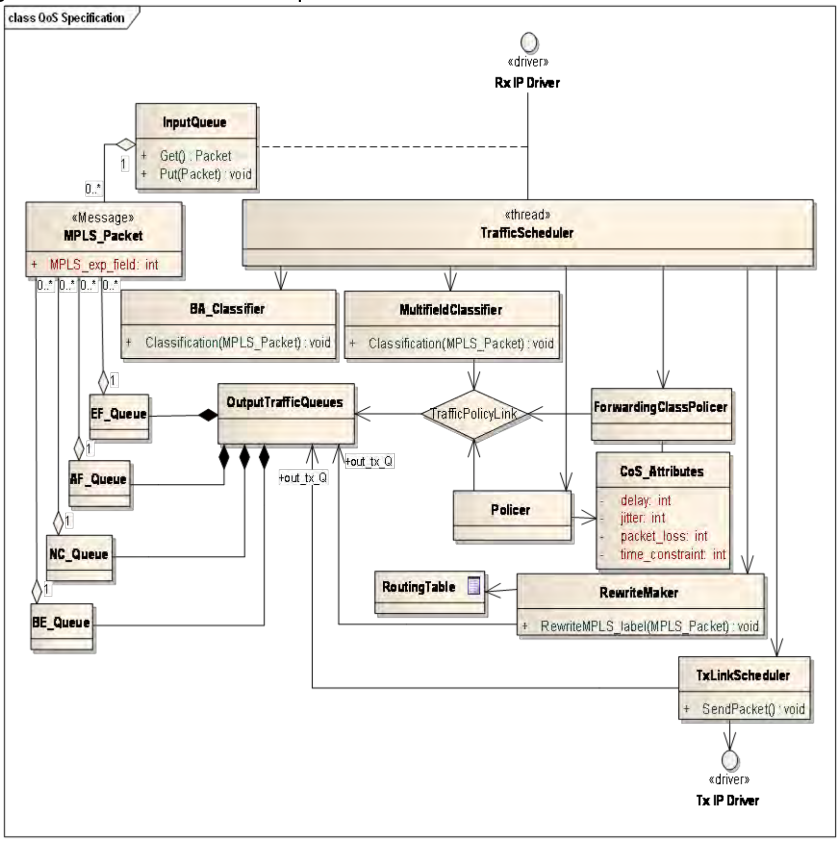


Fig. 6. Black box view of router QoS implementation

5. Traffic Classes in the Network that is the Subject of Experiment

As it is well known, IP traffic from the aspect of MPLS network and core routers can be classified into: EF (Expedited Forwarding), NC (Network Control), AF (Assured Forwarding) and BE (Best Effort) traffic classes [8]. These classes have their own forwarding requirements in each of the network routers. In the testing process itself, every packet should be assigned to one of four predefined data forwarding classes (FCs):

EF class provides the lowest values of packet loss, delay and jitter. It also secures the bandwidth for services in this class, end to end. Voice and video packets, with real-time traffic requirements, belong to this class.

AF class allows the definition of group values for certain traffic subclasses, including three different packet rejection possibilities. Traffic packets from layer-2 and layer-3 VPNs belong to this service class.

BE class does not provide reliable types of service profiles; it usually applies a more aggressive packet rejection profile, e.g. random early detection (RED). It encompasses data packets.

NC class usually has a high priority since it supports network control protocols operation.

Each of the listed classes is suitable for a specific service category. For the transfer of traffic sensitive to delays, jitters and losses, such as voice and video transmission in real time, it is necessary to assign EF service class. For common Internet traffic, BE class offers a satisfactory service, whereas transmission of higher priority traffic should be assigned the AF class. NC service category should be chosen for the exchange of packets necessary for the operation of routing protocol in the network.

There are several different methods of packet classification, i.e. assigning packets to different service classes based on the value contained in the CoS fields or based on the conditions examining the values of other fields within the packet header. In our case the router shall not undergo any configuration adjustments, but rather a BA classifier shall be implemented in the traffic generator.

The considered network contains the *Juniper Networks* routers (M320 and M10i in Fig. 7) configured to support up to 4 FCs (Forwarding Classes) [9]. The FCs configuration with Junos software is listed at listing 1:

```
[edit class-of-service forwarding-class]

user@RouterA# show

queue 0 BE;

queue 1 EF;

queue 2 AF;
```

```
  queue 3 NC;
```

Listing 1. Network routers scheduling configuration

It can be seen that traffic is grouped into four scheduling queues that support FSs explained earlier.

6. QoS/CoS Test Environment

The goal of the testing was to observe the quality of transmission, the transmission priorities and packet rejection rate in a case of traffic congestion. For the previously mentioned CoS configuration, it is necessary to demonstrate the scheduler shape for different classes, so the router scheduler print-out is shown in Fig. 8. The measurements are done using the part of the considered network whose topology is presented in Fig. 7. The reasons of such a topology are as follows:

The capacity of particular interfaces: the Agilent Router Tester is connected to the edge routers via GE (Gigabit Ethernet) interface, while the traffic was supposed to be routed over the link of lesser capacity (STM-1 - 155Mbps), in order to simulate the congestion and analyze the provoked packet rejection in the router,

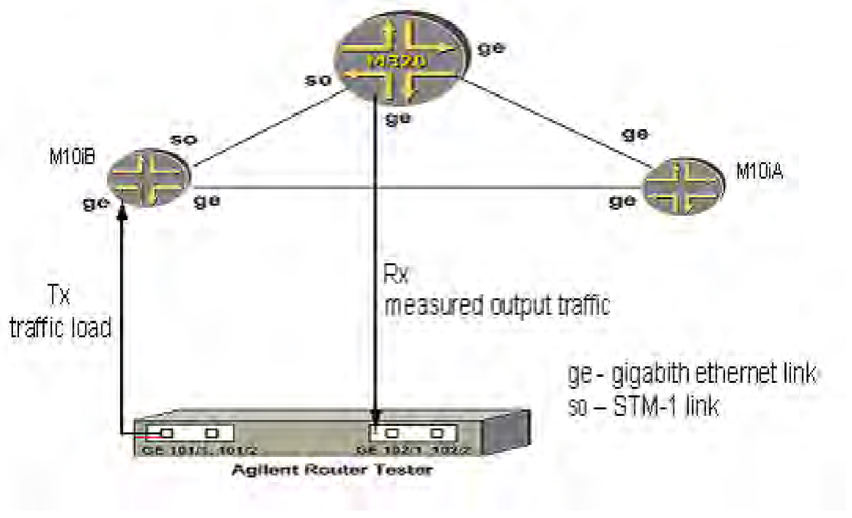
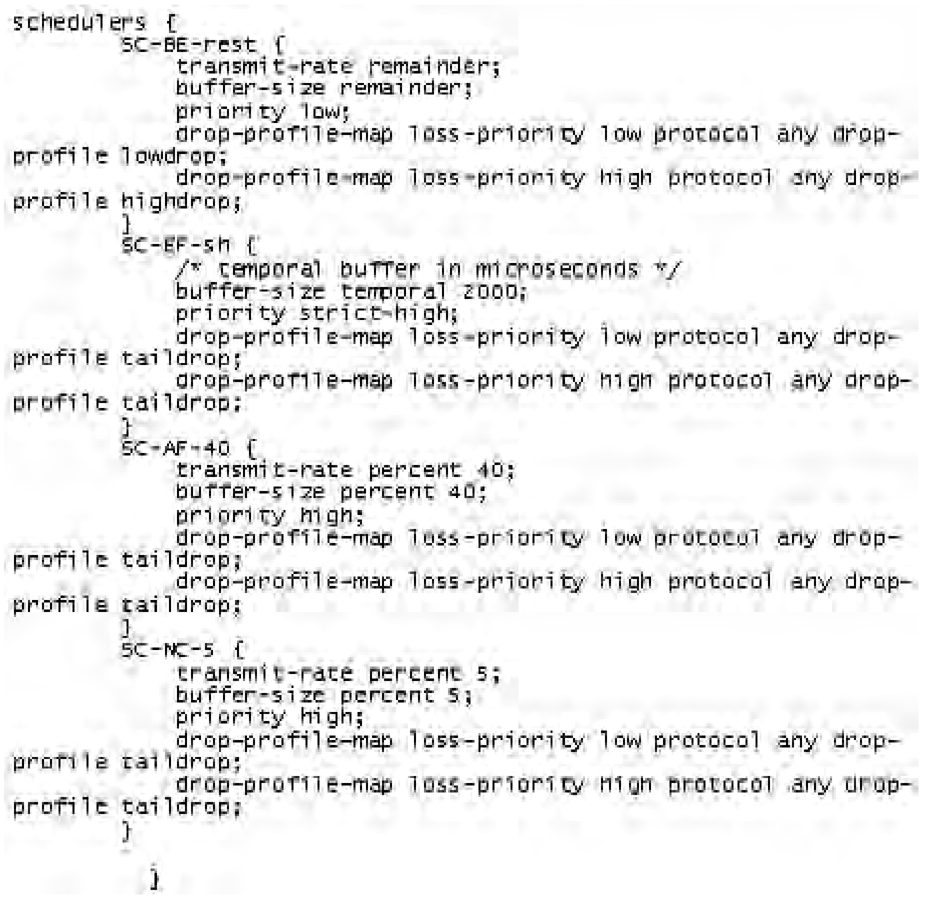


Fig. 7. Test topology

The test topology is made as simple as possible, since the structure of other considered network elements (routers and links) is designed following the same pattern and the same results would be obtained in any part of the network.



```
schedulers {
        SC-BE-rest {
            transmit-rate remainder;
            buffer-size remainder;
            priority low;
            drop-profile-map loss-priority low protocol any drop-
profile lowdrop;
            drop-profile-map loss-priority high protocol any drop-
profile highdrop;
        }
        SC-EF-sh {
            /* temporal buffer in microseconds */
            buffer-size temporal 2000;
            priority strict-high;
            drop-profile-map loss-priority low protocol any drop-
profile taildrop;
            drop-profile-map loss-priority high protocol any drop-
profile taildrop;
        }
        SC-AF-40 {
            transmit-rate percent 40;
            buffer-size percent 40;
            priority high;
            drop-profile-map loss-priority low protocol any drop-
profile taildrop;
            drop-profile-map loss-priority high protocol any drop-
profile taildrop;
        }
        SC-NC-5 {
            transmit-rate percent 5;
            buffer-size percent 5;
            priority high;
            drop-profile-map loss-priority low protocol any drop-
profile taildrop;
            drop-profile-map loss-priority high protocol any drop-
profile taildrop;
        }

          }
```

Fig. 8. The router scheduler print-out

6.1. UML Specification of test network

Test network, whose topology is in the figure 9, deploys three routers, one M320 and two M10. M10 router is Juniper router that has similar traffic capabilities as T640 core router. Assumption made about this topology: this small network has the basic traffic properties like the IP/MPLS network in figure 1. STM-1 link, with 155Mbps capacity is used to provoke overload condition (traffic congestion) with available tester (Agilent Router Tester) whose traffic is limited with 1 gigabit Ethernet interfaces. Test Scenario (left down corner) script defines traffic load for different CoS classes as it will be explained in the paper sections that follows for different QoS tests. Rx Traffic Measurements object collects QoS of traffic that passed the communication route defined by test topology. The difference between the measured QoS and CoS attributes in fig. 6 is in CoS time constraint attribute used in traffic scheduling but without significance for end users.

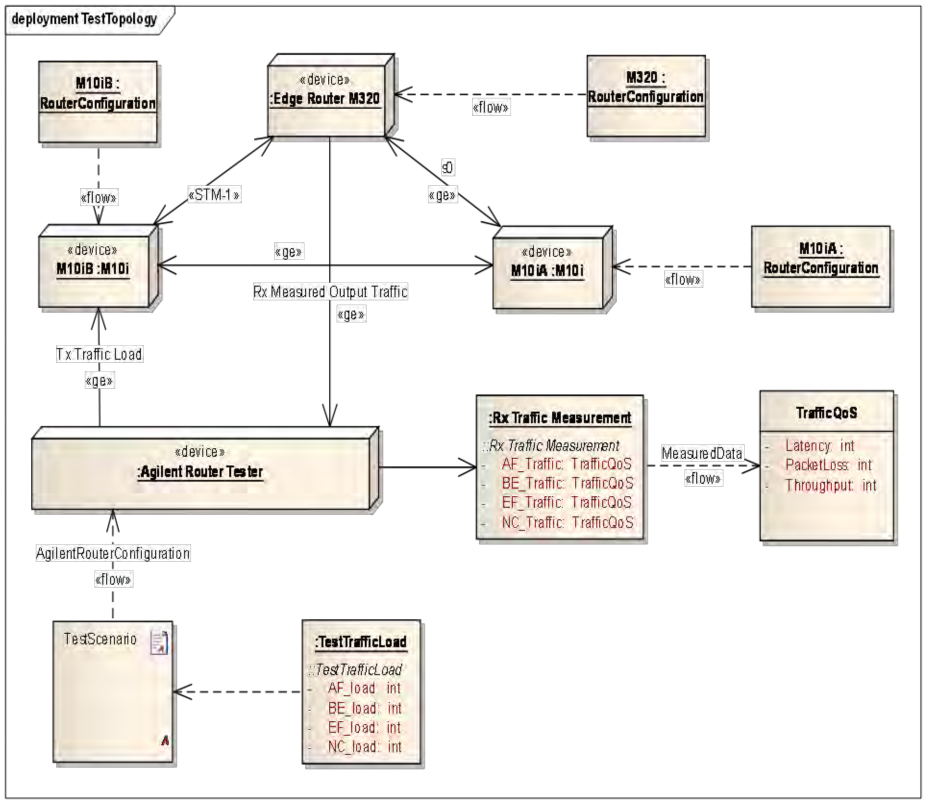


Fig. 9. Topology of test network

6.2. Specification of MUX

In packet networks traffic distribution is stochastic in time and packet length. Figure 10 is a possible specification of statistical MUX, consisted of two concurrent threads: Listening and Sending. They communicate via InputQueue shared data supplied with PutPacket and GetPacket methods. Output port is simple serial interface to communication link, e.g. optical line. InputPort is e.g. optical line interface, while InputHW(input Hardware) block can be realized as DMA(Direct Memory Access) device that triggers listening thread each time when end of packet is detected.

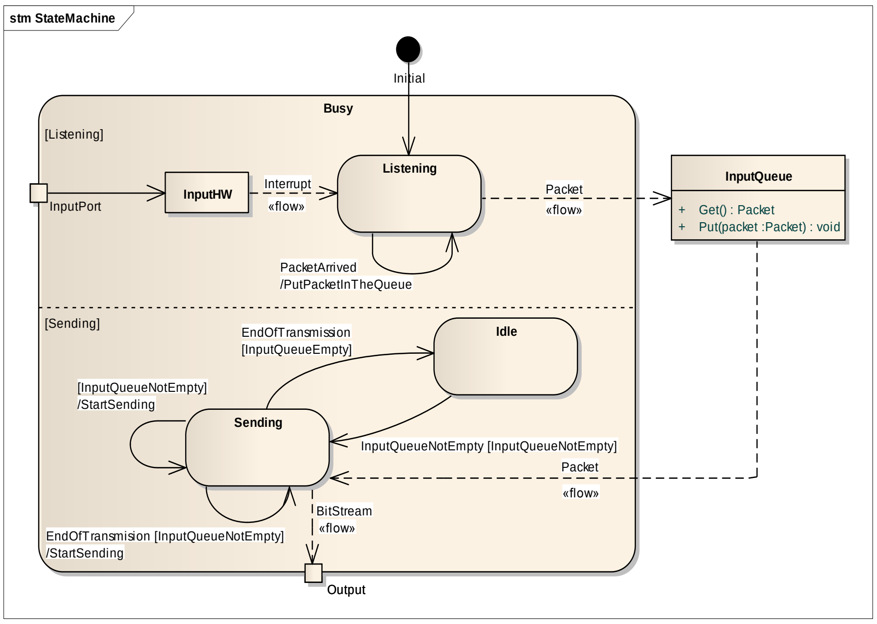


Fig. 10. Statistical MUX

The traffic generator was *Agilent Router Tester* that generates different classes of traffic with the predefined traffic distribution and with fixed packet length of 1000 bytes. At the same time Agilent router tester measured the output traffic on Rx traffic link shown in Fig. 7. The aim was to approve the expected performances of CoS mechanism in a case of congestion: EF class (voice traffic) should not show evidence of delay and packet loss, nor its excessive traffic should affect AF and NC classes; on the other hand, the excessive amount of AF and/or BE class should not affect EF class in the case of congested link, not even if the EF traffic itself is oversized. Each of the GE links, including the Agilent generated input traffic link, had the allocated bandwidth distribution network routers, as follows:

NC - 50Mbps - 5%;
AF - 400Mbps - 40%;
BE - 275Mbps - 27.5%;
EF - 275Mbps - 27.5%.

6.3. QoS/CoS – Test 1

The first test compares the priorities forwarding policy of EF and BE classes that occupy 55% of the whole bandwidth. Considering the link of lesser capacity (STM-1, 155Mbps bandwidth), between the M10iB and M320 routers that is a cause of congestion, network overload was provoked. EF and BE classes were allocated, in M10iB and M320 routers, for 40Mbps of bandwidth each, with a possibility to extend to any part of temporary unused bandwidth of other classes. Also, EF class can use the BE bandwidth at any time. In order to provoke the congestion, 140Mbps of BE and EF maximal traffic were generated by Agilent router tester. Packets were generated with constant packet interarrival time (PIAT), fixed length (1000 bytes) and linear increase of input traffic load. The results are shown in Fig. 11 and Table 1. As expected, due to high priority class, no EF packet is lost, while BE packet suffer a high rejection ratio of 92%. Obtained results clearly lead to the conclusion that router scheduling policy is properly implemented for this traffic condition.

Table 1. Test 1 results: EF and BE priority classes traffic

CoS	Tx Test Packets	Rx Test Packets	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)	Rx Packet Loss	Average Latency (us)
BE	140000,00	11534,00	124,44	10,25	128466,00	41793,86
EF	140000,00	140000,00	124,44	124,44	0,00	244,64

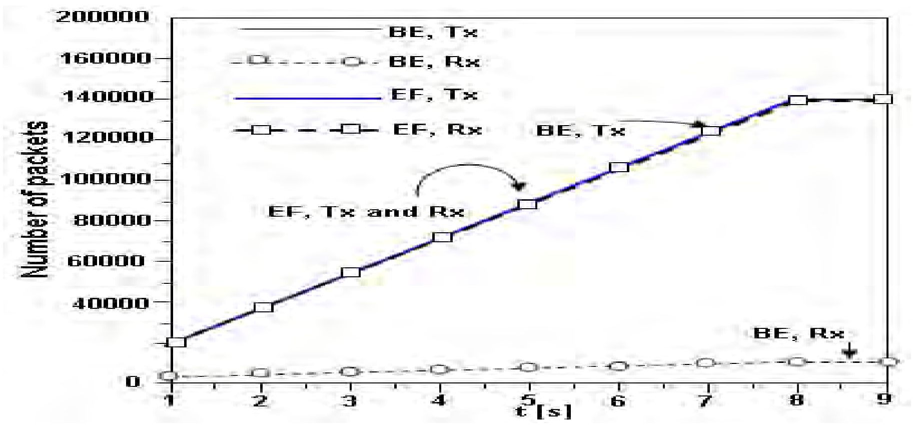


Fig. 11. Test 1 result: EF and BE priority classes traffic

6.4. QoS/CoS – Test 2

The second test compares the priorities of AF and BE classes in a case of congestion. AF class occupies 40% of effective bandwidth or 60Mbps in the case of STM-1 link while the bandwidth of BE class was set to 140Mbps. Again, the results were as expected: only the BE packets suffered from the rejection (Fig. 12 and Table 2).

Table 2. Results of QoS/CoS test 2 for the AF=60Mbps and BE=140Mbps

CoS	Tx Test Packets	Rx Test Packets	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)	Rx Packet Loss	Average Latency (us)
AF	89250,00	89250,00	54,92	54,92	0,00	245,12
BE	208250,00	135344,00	128,15	83,29	72906,00	16082,15

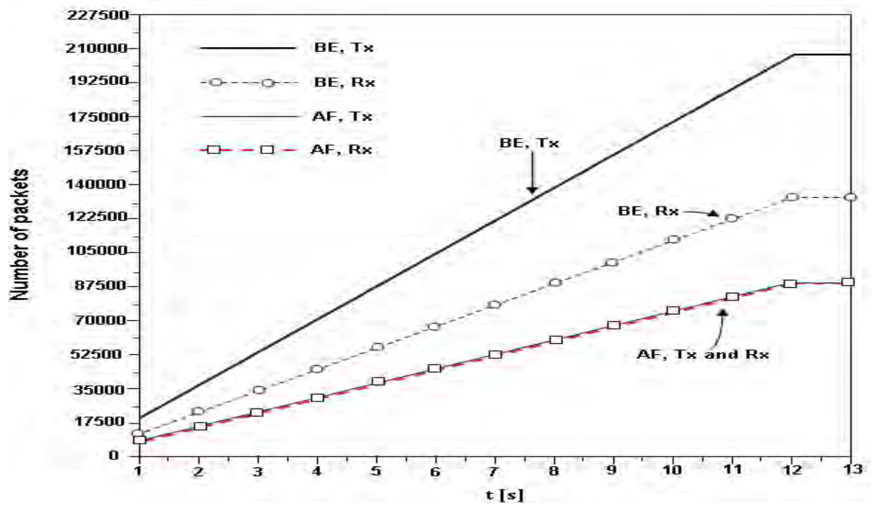


Fig. 12. Results of QoS/CoS test 2 for the AF=60Mbps and BE=140Mbps

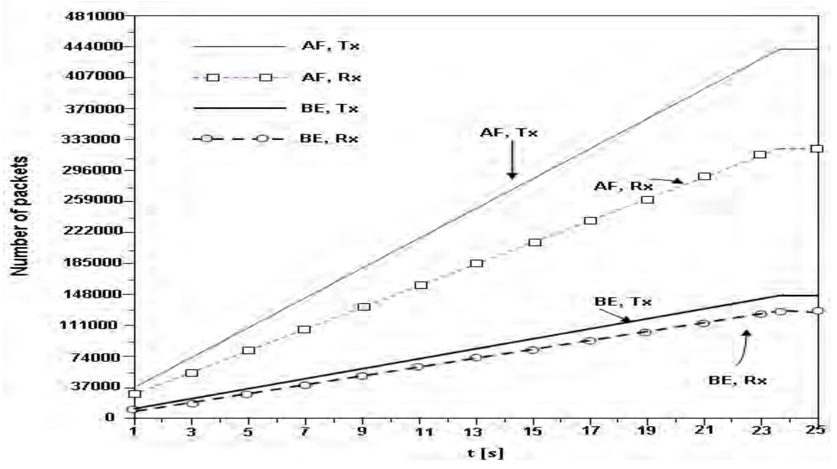


Fig. 13. Results of QoS/CoS test 2 for the AF=150Mbps and BE=50Mbps

An opposite scenario is shown in Fig.13 and Table 3: 150Mbps of AF class and 50Mbps of BE priority class was allocated at STM1 link. Since BE does not exceed its predefined limits, it suffers minimal but inevitable losses since

AF shares part of its bandwidth. On the other hand, AF suffers considerable losses, since it exceeds the link limits.

Table 3. Results of QoS/CoS test 2 for the AF=150Mbps and BE=50Mbps

CoS	Tx Test Packets	Rx Test Packets	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)	Rx Packet Loss	Average Latency (us)
AF	442500,00	322816,00	141,60	103,30	119684,00	38594,56
BE	147500,00	122881,00	47,20	39,32	24619,00	21577,42

6.5. QoS/CoS – Test 3

The aim of the third test is to compare AF and EF classes of service. Although EF CoS is of high priority, it cannot disturb AF class if the latter remains within its own allocated bandwidth. To approve this, two scenarios were tested. For the first one (Fig. 14 and Table 4) the parameters were AF=150Mbps and EF=50Mbps of generated input traffic; results of the test show that only AF class packets out of predefined bandwidth were rejected.

Table 4: Results of QoS/CoS test 3 for the AF=150Mbps and EF=50Mbps

CoS	Tx Test Packets	Rx Test Packets	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)	Rx Packet Loss	Average Latency (us)
AF	318750,00	214909,00	141,67	95,52	103841,00	38564,30
EF	106250,00	106250,00	47,22	47,22	0,00	244,66

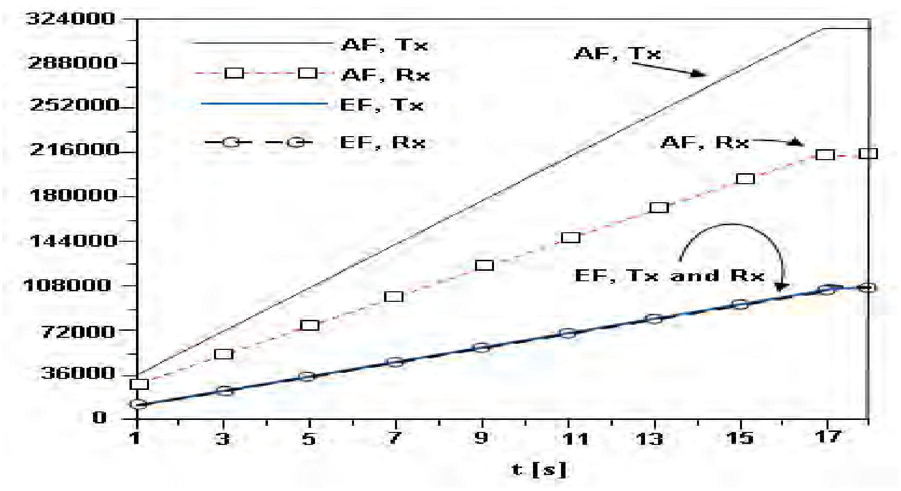


Fig. 14. Results of QoS/CoS test 3 for the AF=150Mbps and EF=50Mbps

The second set of parameters was AF=50Mbps and EF=150Mbps (Fig. 15 and Table 5). In spite of high priority class, the EF packets were rejected as the maximal allocated bandwidth (80Mbps) was exceeded.

Table 5. Results of QoS/CoS test 3 for the AF=50Mbps and EF=150Mbps

CoS	Tx Test Packets	Rx Test Packets	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)	Rx Packet Loss	Average Latency (us)
AF	80000,00	80000,00	45,71	45,71	0,00	246,09
EF	240000,00	161282,00	137,14	92,16	78718,00	617,85

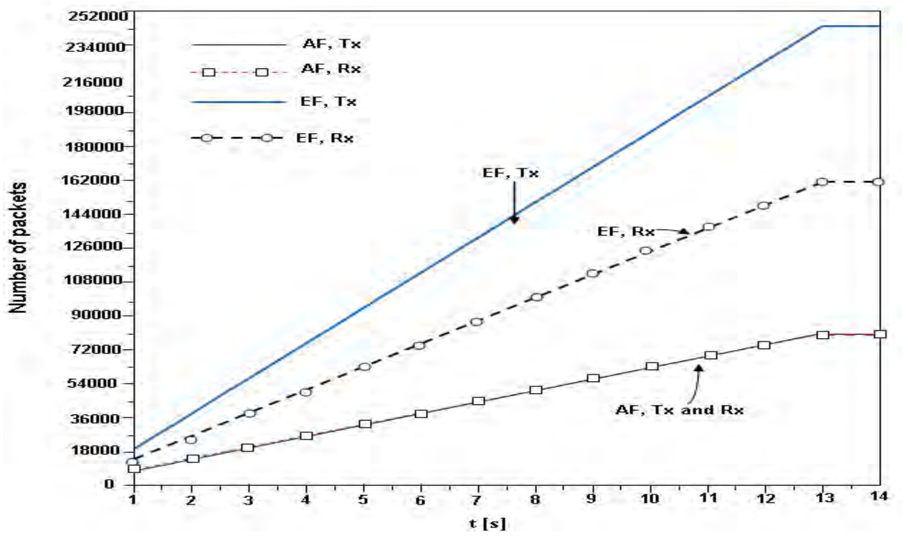


Fig. 15. Results of QoS/CoS test 3 for the AF=50Mbps and EF=150Mbps

6.6. QoS/CoS – Test 4

The aim of this test is to observe the network congestion in case of three classes of traffic, EF, AF and BE. The test has approved the expected results that AF class is not affected within its own bandwidth, that while EF and BE classes share the remaining resources, and also that EF class is not affected by any class, nor it can affect any other class, except BE. The testing input traffic load parameters were: EF=80Mbps, AF=80Mbps and BE=80Mbps. From Fig. 16 it could be noticed that EF packets are not rejected, that AF packet are not rejected within its own predefined bandwidth and that only the remaining part of full STM-1 bandwidth is allocated to BE.

Table 6. Results of QoS/CoS test4 for EF=AF=BE=80Mbps

CoS	Tx Test Packets	Rx Test Packets	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)	Rx Packet Loss	Average Latency (us)
AF	266000,00	202013,00	76,00	57,72	63987,00	71932,13
BE	266000,00	34809,00	76,00	9,95	231191,00	70279,57
EF	266000	266000	76	76	0	273,69

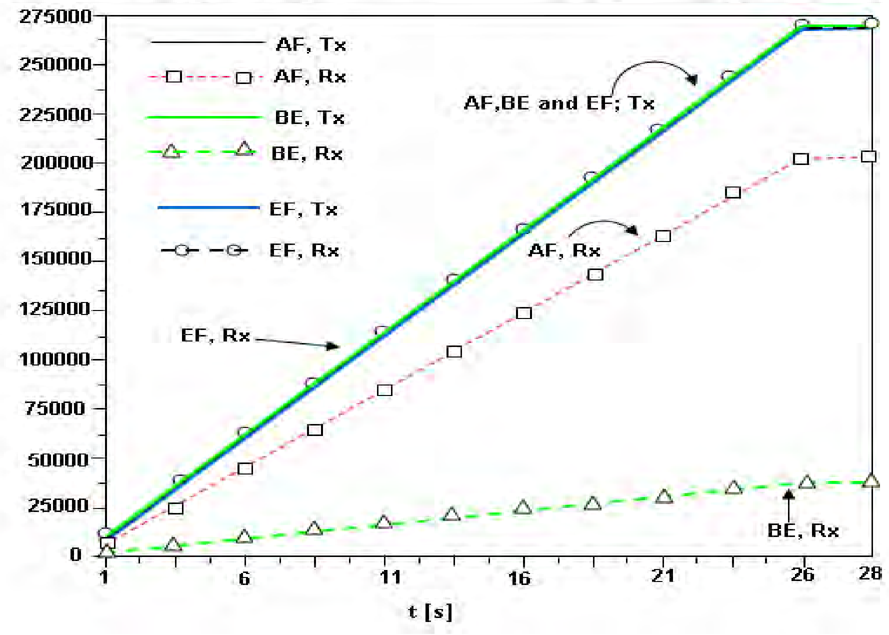


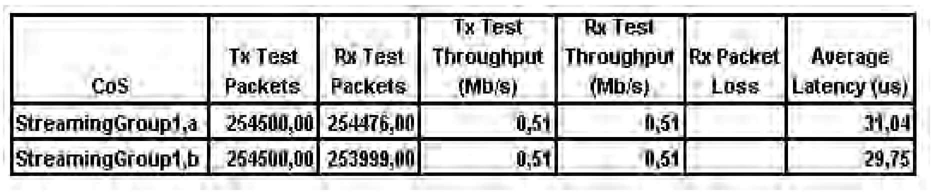
Fig. 16. Results of QoS/CoS test 4 for the EF=AF=BE=80Mbps

6.7. Voice Packet Latency Test

The delay time is essential for QoS of the voice traffic. For this reason, average delay time of a packet is measured - for a single router and along the STM-1 path. 250 Mb voice traffic was generated, with packet length that equals to 64 byte. The *Agilent Router Tester* was employed again, at the network test topology shown at Fig. 7. Two tests were executed. In first one the propagation time along the three successive routers M10iB → GE → M10iA → GE → M320 directly coupled implementing an optical link yielded delay time of 30μs, as it can be seen in the table 7. Each router equally contributes to the delay time, so it may be concluded that the delay time for a single router is 10μs.

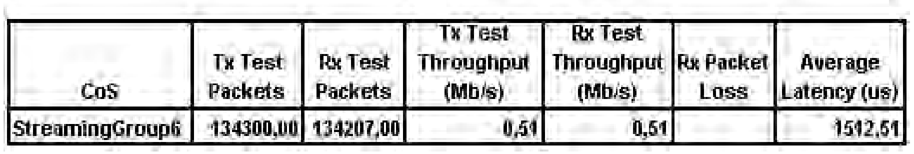
In the second test we used the same path was, but routers were connected with GE over STM-1 links. The total propagation time measured along the path M10iB → STM-1 → M10iA → STM-1 → M320, yielded a delay time that is less than 2ms (table 8), well below the standard recommendation values for voice signal (50ms).

Table 7. Results of measurements for the voice packet latency test 1



CoS	Tx Test Packets	Rx Test Packets	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)	Rx Packet Loss	Average Latency (us)
StreamingGroup1,a	254500,00	254476,00	0,51	0,51		31,04
StreamingGroup1,b	254500,00	253999,00	0,51	0,51		29,75

Table 8. Results of measurements for the voice packet latency test 2



CoS	Tx Test Packets	Rx Test Packets	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)	Rx Packet Loss	Average Latency (us)
StreamingGroup6	134300,00	134207,00	0,51	0,51		1512,51

7. Conclusion

In this paper we presented a number of tests conducted in public IP/MPLS network and presented the obtained results. The aim was to verify the policy by which different bandwidths would be allocated to different classes of services in the IP/MPLS network. Namely, we generated different traffic cases that increased the probability of overloading the network with ensuing increase of packet loss rate. The tests were conducted partially in specific parts of the network, but since configured policy for bandwidth allocation is replicated in all routers, the assumption regarding testing is that the same results would be obtained in testing of the whole network. Completed tests include user traffic (EF, AF and BE service class), whereas control traffic (NC service class) is not included in the tests.

IP/MPLS network testing, described in this paper, was based on Telecom operator experiences in circuit switching network and packet networks (X.25, frame relay and IP data network). Proposed worst case test set exhausts all of the combinations for the EF, AF and BE traffic that can cause network congestions. Heuristic assertion for the network behavior in real IP traffic conditions is that the bandwidth allocation will be proper and expectable if it is same in the worst case traffic conditions. Besides the traffic overload tests we conducted latency test that verifies QoS for VoIP service.

During the testing, commercially available generators for input traffic load with the following characteristics were used: fixed packet length, constant packet, interarrival time and linear increase of traffic load. In the analysis of the obtained results this is a restrictive factor because in the real traffic network load implies variable packet length, packet interarrival time and burst traffic generating. For achieving more realistic results which will represent the policy adopted in the network in greater quality, it is necessary to use a generator that generates input traffic with variable lengths, packet interarrival times and burst periods. It is certain that each operator has its own business case, which implies that besides basic service classes such as voice, video, data and VPNs, other service classes are also created (e.g. emergency calls). In other words, this means that further research should be extended to the testing of all 8 service classes which would include testing of all potentials of the IP/MPLS network. Experience obtained in described IP network testing and verification of network bandwidth allocation policy is a sufficient basis to define IP traffic simulator and/or traffic generator. Idea for this is explained in section 4.2 that describes statistical MUX specification. It can be expected that more realistic results would be accomplished if the simulator/generator had a possibility to produce traffic with stochastic features, packet interarrival time and packet length. In the future work the traffic generator should support typical traffic shapes, with stochastic features, for different network services, e.g. VoIP (Voice over IP), IPTV (IP television), web browsing, etc.

At the end we want to emphasize that, besides the above fact, executed tests and measurements procedures give sufficient information about considered IP/MPLS network behavior in the overload traffic condition, since they represent the worst case of input traffic load.

References

1. E. Rosen, A.Viswanathan, R. Callon: Multiprotocol Label Switching Architecture, IETF RFC 3031. (Jan 2001)
2. Ina Minei, Julian Lucek: MPLS-Enabled Applications (Chapter 2 -Traffic Engineering with MPLS (MPLS-TE) (p 37-67)), [Online], Available: http://www3.interscience.wiley.com/cgi-bin/bookhome/, (Published Online: 21 Feb 2006)
3. K. Nichols, S. Blake, F. Baker, D. Black: RFC 2474 - Definition of the Differentiated Services Field (DSF), (December 1998)
4. OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, OMG Document Number: formal/2007-11-02 Standard document URL: http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF
5. Enterprise Architect-Free UML Tool Downloads for OOA/D Software Development, http://www.sparxsystems.com.au/products/ea/downloads.html
6. Allen Holub's UML Quick Reference, http://www.holub.com/goodies/uml/
7. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss: RFC 2475 - An Architecture for Differentiated Services, (December 1998)
8. JUNOS MPLS Applications Configuration Guide: Configuring CoS for MPLS [Online], Available:http://www.juniper.net/techpubs/software/junos/junos80/ swconfig80-cos/html/cos-mpls.html, (Published Online: 2007)

9. Juniper: Supported CoS Standards and features, http://netscreen.com/techpubs /software/junos/junos91/swref-hierarchy/supported-cos-standards-and-features.html

Živko Bojović is employed at Telekom Srbija at the position Manager of Department for Investment development and Maintenance in Belgrade, Serbia. He received his Master Degree (2001) in Telecommunications Science from the University of Novi Sad, Faculty of Technical Sciences. Since 1999 he has been with Telekom Srbija in Belgrade Mr. Bojović participated in several science projects. He published more than 9 scientific and professional papers. His main research interests include speech technologies, service based on speech technologies, and services based on implementation of Internet Protocol in area of telecommunication network. He can be contacted at: zivko@telekom.rs.

Emil Šećerov is holding the assistant professor position at the Faculty of Technical Sciences, Novi Sad, Serbia since year 2000. He received MSc from the University of Novi Sad in 1993 and a PhD degree from the same university in 1998, both in electrical engineering. His area of interest are: programming and software engineering, operating systems and system software and telecommunication and computer networks. He can be contacted by e-mail: secerov@uns.ac.rs.

Vlado Delić is holding the associate professor position at the Faculty of Technical Sciences, Novi Sad, Serbia. He received MSc from the University of Belgrade in 1993 and a PhD degree from University of Novi Sad in 1997, all in electrical engineering. Particularly interested in audio signal processing and human-machine speech communications. He established a R&D group that is the leader in RTDI and applications of ASR&TTS in South Slavic languages. Chapters in several books of communications and signal processing. More than 100 scientific papers in the last 10 years. He has been bestowed several prestigious awards for his creative work and contribution to innovations in the region. He co-chairs the biennial conference DOGS. Professional Memberships: AES, IEEE.

Received: July 10, 2009; Accepted: December 15, 2009.

UDC 519.876.3, DOI:10.2298/CSIS100118012G

Cost of Cooperation for Scheduling Meetings

Alon Grubshtein and Amnon Meisels

Dept. of Computer Science
Ben Gurion University of the Negev
P.O.B 653, Beer-Sheva, 84105
Israel
{alongrub,am}@cs.bgu.ac.il

Abstract. Scheduling meetings among agents can be represented as a game - the Meetings Scheduling Game (MSG). In its simplest form, the two-person MSG is shown to have a price of anarchy (PoA) which is bounded by 0.5. The PoA bound provides a measure on the efficiency of the worst Nash Equilibrium in social (or global) terms. The approach taken by the present paper introduces the *Cost of Cooperation* (CoC) for games. The CoC is defined with respect to different global objective functions and provides a measure on the efficiency of a solution *for each participant* (personal). Applying an "egalitarian" objective, that maximizes the minimal gain among all participating agents, on our simple example results in a CoC which is non positive for all agents. This makes the MSG a *cooperation game*. The concepts are defined and examples are given within the context of the MSG.
Although not all games are cooperation games, a game may be revised by adding a mediator (or with a slight change of its mechanism) so that it behaves as a cooperation game. Rational participants can cooperate (by taking part in a distributed optimization protocol) and receive a payoff which will be at least as high as the worst gain expected by a game theoretic equilibrium point.

Keywords: Multi-Agent Systems, Cooperation, Meeting Scheduling.

1. Introduction

Scheduling meetings between two or more people is a difficult task. Despite the advances offered by modern world electronic calendars these are often limited and serve as passive information repositories. As a result, a dedicated person is usually hired to handle this task. Previous attempts to automate the meetings scheduling problem (MSP) employ one of two extreme approaches: the cooperative approach and the competitive one. Cooperative methods for solving MSPs perform a distributed search for an optimum of a global objective [8]. The competitive approach investigates game theoretic equilibria of suitable games and designs strategies for the competitive agents [2]. One previous attempt to combine the two approaches was introduced in [4], which empirically attempted to introduce selfishness into a cooperative protocol for the Meetings Scheduling

Problem (MSP). Instead of searching for a solution which is socially optimal and potentially harmful for some agents, alternative global objectives were examined. Such solutions maintain an important goal - they provide acceptable gains to each one of the agents [4].

The present paper attempts to introduce cooperation into a self interested scenario by using a simplified *meetings scheduling game* (MSG). The use of game theory for studying self interested interactions provides a sound mathematical basis for the analysis of strategic situations under a set of (commonly) acceptable axioms (cf. [12]). A fundamental result of game theory is the existence of equilibrium points which define an action (or strategy) profile that is stable (i.e., no participant can gain by deviating from this profile). There are many definitions of various types of stable points, and unless otherwise specified we will assume the common "pure strategy Nash Equilibrium" (PSNE, or simply NE). A NE is a set of actions (assignments), one for each agent, in which no agent can gain by unilaterally changing its value [12].

Investigating the simple MSG, one shows that the NE of the MSG may be different than its optimal points. This is similar to routing games (cf. [15]). Consequently, the widely accepted efficiency measure, known as the *Price of Anarchy (Stability)* [7, 13, 16, 12, 15], is different than unity for the simple MSG. For scheduling meetings and its underlying game, we present user oriented criteria, and formulate a game property based upon it - the *Cost of Cooperation* (CoC). This property can motivate selfish users to cooperate by presenting *each one* of the players a *guaranteed* cooperative gain which is higher than its worst equilibrium value.

2. Meetings Scheduling by Agents

The Meeting Scheduling Problem (MSP) involves a (usually large) set of users with personal schedules and a set of meetings connecting these users. It is an interesting and difficult problem from many aspects. In the real world, agents have a dual role. They are expected to be both self interested (i.e. seek the best possible personal schedule) but also cooperative (otherwise meetings will not be scheduled).

MSP users have personal calendars which are connected by constraints that arise from the fact that multiple groups of the overall set of users need to participate in meetings. Meetings involve groups of users and each meeting includes at least two participants. Meetings can take place in different places and carry a different significance for their participants. All participants have their personal schedules and need to coordinate their meetings. Each user is represented by an agent and the set of agents coordinate and schedule all meetings of all users. The final result of such an interaction is a set of updated personal schedules for all agents/users (a global calendar). In this work we use utilities to represent the significance that users attach to meetings and to certain features of the resulting schedule.

Two very different approaches were studied for solving MSPs. The first employs Distributed Constraints Satisfaction (DCSPs) and Optimization (DCOPs) [8, 10, 19]. In these studies, agents are expected to be *fully cooperative* and follow the protocol (algorithm) regardless of their private gains. The outcome of such protocols is a solution whose **global utility** is optimal (or consistent in the case of a DCSP). The global utility does not necessarily account for the quality of the personal schedules of the participants. Indeed, due to the combinatorial nature of the MSP it is often the case that a globally optimal solution includes a low quality solution for at least one participant [4].

An alternative approach is offered by researchers in the field of *Game theory*. Here, agents are rational, *self-interested*, entities which have different (and often conflicting) utility functions (cf. [12]). A large share of the game theoretic research related to MSPs emphasizes the underlying mechanism of the interaction [3, 2, 14]. The basic assumption of these studies is that the gain of agents from their scheduled meetings can be represented by some universal currency. Moreover, the assumption accepts a uniform exchange rate for some monetary means and *unscheduled* meetings. These fundamental assumptions seem very unrealistic for scheduling meetings among people. However, if one is ready to accept this monetary model many game theoretic results and mechanisms can be applied. One important example is that mechanisms for discouraging cheating, that are known for combinatorial auctions, can possibly be used for agents that schedule meetings [14].

The approach of the present study examines simple game theoretic mechanisms to restrict or predict agents behaviors in an inherently cooperative environment.

3. The meetings scheduling game (MSG)

Imagine a pair of self-interested users, each with her own preferences about her personal schedule. Many agreements need to be reached in order to generate a viable and consistent schedule. One way to enforce agreements is to pose the problem as a game with rules that enforce schedules that are then accepted by the participants. In a way, this is analogous to routing by a player that accepts the delay as a given result of the route it selected. One can think of this as the mechanism of the game.

An equilibrium point is defined with respect to a given game, so we begin by defining a simple MSG. The game includes two players coordinating a meeting. The meeting can be scheduled in the morning or in the evening. Each player places her *bid* for scheduling by specifying her preferences (or expected gains) from each schedule. This list of bids takes the form of a tuple $b_i = \langle x, y\rangle \in \{0, 1, ...B\} \times \{0, 1, ...B\}$ for the two time slots, where the first value (x) corresponds to the morning time slot and the second number (y) corresponds to the evening time slot. A higher number indicates a higher preference towards having the meeting at the corresponding time slot (a stronger desire to meet at that time slot). Note that the bids made by the players do not necessarily represent a player's real preference. That is, players may act strategically and attempt to "deceive" their opponent into an action which may result in a better or stable personal outcome. As we shall soon demonstrate, in the case of our simple example, bidding "truthfully" is a dominating action (the action that results in the best payoff for every action taken by the opponent).

Once both bids are presented, a time slot is selected by the *game's mechanism* according to its decision rule, and the participants are informed of the decision. There are many possible decision rules, but for now we use the "utilitarian" approach [12] - the time slot selected is the one with the highest sum of bids on it (or when sums are equal, outcomes have equal probabilities). In other words, the selected time slot has the highest total (reported) gain for the players. The end payoff of each participant is defined by the player's utility function (preference) for a time-slot.

We limit ourselves to the case where players are never indifferent to the results, i.e., they will always prefer some time slot over the other. Any bid of the form $\langle x, x \rangle$ is thus excluded (it results in a scheduling based on the opponent's preference).

The first approximation of the MSG assumes that when the meeting is held at a time that is not preferred by a player, her gain from it will equal zero. We will later remove this assumption. We distinguish between the payoffs of players when the meeting is held at the player's most desired time, and say that player 1's payoff in such a case is m, while player 2's payoff for an analogous case is k. This enables us to treat users that are essentially different.

Figure 1 depicts the simplest possible MSG for two players - the "rows" player (player 1) and the "columns" player (player 2). The outcome of a joint bid is composed of two values: the left hand value represents player 1's gain while the right hand value represents player 2's gain from the joint action. The descriptive power of a strategy is limited to two values of preference, either 0 or 1 (preferred). In this example player 1 prefers to have the meeting in the morning. When player 2 also prefers the morning time-slot both players have the same dominant strategy - $\langle 1, 0 \rangle$. That is, bidding $\langle 1, 0 \rangle$ will *always* yield a higher payoff for both players. Such a strategy profile is said to be stable since no player has an incentive to deviate from her bid.

When player 2 prefers the evening time-slot her dominant strategy becomes $\langle 0, 1 \rangle$ (player 1's dominant strategy remains $\langle 1, 0 \rangle$). This can immediately be translated into an equilibrium point resulting from playing the dominant strategy. When the desires of both players are the same, the equilibrium payoff is m, k, and when these desires conflict the equilibrium payoff is $\frac{m}{2}, \frac{k}{2}$ (i.e., expected values).

If a NE, or a stable point, is the expected outcome of an interaction, a valid question to ask would be "How globally efficient or inefficient is this solution?". This question is the focus of a large body of work on "the Price of Anarchy" (PoA) [12, 13, 7, 16, 15]. The PoA is a measure of the inefficiency of an equilibrium point, and is defined as the ratio between the cost of the *globally* worst

p_1 \ p_2	$\langle 0,1\rangle$	$\langle 1,0\rangle$
$\langle 0,1\rangle$	$0,0$	$\frac{m}{2},\frac{k}{2}$
$\langle 1,0\rangle$	$\frac{m}{2},\frac{k}{2}$	m,k

(a)

p_1 \ p_2	$\langle 0,1\rangle$	$\langle 1,0\rangle$
$\langle 0,1\rangle$	$0,k$	$\frac{m}{2},\frac{k}{2}$
$\langle 1,0\rangle$	$\frac{m}{2},\frac{k}{2}$	$m,0$

(b)

Fig. 1: Payoff matrices of a simple MSG. Player 1 (Rows) prefers a morning meeting, and player 2 (Columns) preferences can be either the same (a), or the opposite (b)

NE, to the cost of the *globally* optimal solution. Studies of the price of anarchy originate with routing games (cf. [15]).

It is interesting to see that the same question can be formulated for MSGs. What is the PoA for MSGs ? In reality, a player does not know her opponent's private preferences and this uncertainty should be reflected in our analysis. However, we will consider the above simple example as two different games for the sake of simplicity. In the first game the desires of both opponents align and in the unique equilibrium point the payoffs are m, k. If one is to examine the overall satisfaction of a solution then the global utility in this case is $m + k$. Clearly this is also the optimal solution's utility when considering a cooperative mechanism (optimizing the sum of gains depicted in the payoff bi-matrix). Hence, $PoA = 1$.

When the desires of the opponents conflict, the global gain in the unique equilibrium point is $\frac{m+k}{2}$. This is not necessarily the optimal solution: when $m < k$ the optimal solution's global utility is k. In this case the PoA moves further away from unity as k increases, [1] and we have:

$$PoA = \frac{m}{2k} + \frac{1}{2}$$

That is, *the Price of Anarchy for the simplest MSG* is bounded by $\frac{1}{2}$.

This behavior is also expected in more realistic forms of the two-players two time-slots MSG in which the players may express their strategies in terms of multiple preference values. In such a case, the limit on the maximal preference of each player - the value B - is greater than 1. The resulting MSG has at least one equilibrium point and one can find the worst possible value that it can have. Next, comes the formulation of these results in the form of lemmas and the outlines of their proofs.

Lemma 1. *Every participant in the MSG described above has one dominating strategy, or bid, which is composed of the value B for the preferred time slot, and 0 for the remaining time slot.*

[1] Unlike the routing minimization problem [15, 16], MSP is a maximization problem, hence the PoA is always smaller than 1. This may lead to some confusion with readers already familiar with the PoA but is consistent with previous work

The proof is simple and requires showing that if an agent assigns B to the preferred time slot and 0 otherwise (i.e., each player's action would be either $\langle B, 0\rangle$ or $\langle 0, B\rangle$), her opponent can at most force a draw, which will result in a fair toss of coin. If both players assign this action, no unilateral deviation will result in a higher payoff.□

Lemma 2. *There are only two possible equilibrium outcome values to the above MSG,* (m, k) *and* $(\frac{m}{2}, \frac{k}{2})$

By examining all five possible outcome values, and noting that a player can always impose a draw, one can easily rule out unstable assignments which lead to $(0, 0)$, $(m, 0)$ or $(0, k)$. Thus we are left with (m, k) and $(\frac{m}{2}, \frac{k}{2})$. □

From these two simple lemmas, it is clear that when preferences coincide, the worst equilibrium payoff is (m, k). In this case the price of anarchy (PoA) is 1. In fact, this is the only equilibrium value due to the dominance of strategies. When the preferences for time slot of the two players are in conflict, the worst equilibrium payoff of both players is $(\frac{m}{2}, \frac{k}{2})$. If (without loss of generality) $m < k$, then the PoA decreases to a value lower than 1 as k increases (and is bounded below by $\frac{1}{2}$).

We now proceed to generalize this game, and allow players to assign any non-negative gain when their less preferred time slot is selected by the game mechanism. More specifically, we define for player one the payoff for a non optimal time slot as $0 \leq x \leq m$, and for player two $0 \leq y \leq k$, as demonstrated in the example in figure 2.

p_1 p_2	$\langle 0,1\rangle$	$\langle 0,2\rangle$	$\langle 1,0\rangle$	$\langle 1,2\rangle$	$\langle 2,0\rangle$	$\langle 2,1\rangle$
$\langle 0,1\rangle$	m, y	m, y	$\frac{m+x}{2}, \frac{k+y}{2}$	m, y	x, k	$\frac{m+x}{2}, \frac{k+y}{2}$
$\langle 0,2\rangle$	m, y	m, y	m, y	m, y	$\frac{m+x}{2}, \frac{k+y}{2}$	m, y
$\langle 1,0\rangle$	$\frac{m+x}{2}, \frac{k+y}{2}$	m, y	x, k	$\frac{m+x}{2}, \frac{k+y}{2}$	x, k	x, k
$\langle 1,2\rangle$	m, y	m, y	$\frac{m+x}{2}, \frac{k+y}{2}$	m, y	x, k	$\frac{m+x}{2}, \frac{k+y}{2}$
$\langle 2,0\rangle$	x, k	$\frac{m+x}{2}, \frac{k+y}{2}$	x, k	x, k	x, k	x, k
$\langle 2,1\rangle$	$\frac{m+x}{2}, \frac{k+y}{2}$	m, y	x, k	$\frac{m+x}{2}, \frac{k+y}{2}$	x, k	x, k

Fig. 2: Payoff matrix for a more general MSG with B=2. Player 1 prefers an evening meeting, and player 2 prefers a morning meeting.

The lemmas hold in this case (the tie's payoff is slightly revised). This can easily be understood by noticing that the best response to an opponent's strategy basically remains the same. Just as before, ties which result in a random selection by the mechanism (coin flip) and a value of $\frac{m+x}{2}$ or $\frac{k+y}{2}$, are always preferred over losing (x or y respectively).

As a result, when the preferences of the two agents are the same, the worst NE (which is also the best) has a value of $m + k$ which is also optimal - leading to a PoA of 1. When preferences contradict, the value of the stable point is $\frac{m+x}{2} + \frac{k+y}{2}$. This is not necessarily optimal. For example, if $\frac{m-x}{2} < \frac{k-y}{2}$ than the optimal result can be $x + k$, resulting in

$$PoA = \frac{m+y}{2(x+k)} + \frac{1}{2}$$

Combining these results we reach several intermediate conclusions. The first is that when players have contradicting preferences, the PoA is bounded below by $\frac{1}{2}$, and depends on the relationship between the players payoffs. The second conclusion is that when the two players have the same preferences, the PoA is 1. Our analysis also indicates that the PoA for the MSG depends on several factors:

- The agents' private payoffs (i.e. m, k, x, y). This motivates the use of a mechanism which maps payoffs to a uniform or universal scale which can further bound this value. Such a scale can be the quality of a schedule as described in [4]. Indeed, when the individual payoffs are equal, the PoA becomes unity.
- The *mechanism* of the MSG. While Figure 2 depicts an MSG with a "utilitarian" mechanism an "egalitarian" one which selects a time slot that maximizes the minimal bid is depicted in Figure 3. The different mechanisms leads to different games, with different stable solutions.
- The PoA may also be affected by the *designer*'s perception of optimality. For example, given the MSG of Figure 2 and $\frac{m-x}{2} < \frac{k-y}{2}$, the solutions yielding the gain $\langle x, k \rangle$ are optimal when considering the "utilitarian" approach (maximal combined payoffs), and the resulting PoA is the same as described above. If, however, one's perception of optimality is that it maximizes the lowest gain (e.g.,"egalitarian"), then the optimal solution gain for this MSG (Figure 3) is $\langle \frac{m+x}{2}, \frac{k+y}{2} \rangle$ and the PoA becomes unity. Optimizing different objective functions can produce substantially "better" results [4].

The MSG toy example can become more realistic along several dimensions:

1. The maximal preference bid (B) - while this value increases the assignment space of each player, we have shown that it does not affect the behavior of the two-players, two-time-slots game.
2. The number of participants. Although not presented here, adding more players does not change the fact that these have a (possibly weak) dominant strategy.
3. The number of time slots. Adding more time slots requires several modifications to the game described above (for example, in case of a tie, will the mechanism randomly pick any of the possible time slots? just those that were tied? Can two time slots receive the same bid value? etc). Even after adding the required adjustment, finding the PoA still relies on the utility that each agent associates with a time slot.

p_1 p_2	$\langle 0,1\rangle$	$\langle 0,2\rangle$	$\langle 1,0\rangle$	$\langle 1,2\rangle$	$\langle 2,0\rangle$	$\langle 2,1\rangle$
$\langle 0,1\rangle$	m,y	m,y	$\frac{m+x}{2},\frac{k+y}{2}$	m,y	$\frac{m+x}{2},\frac{k+y}{2}$	m,y
$\langle 0,2\rangle$	m,y	m,y	$\frac{m+x}{2},\frac{k+y}{2}$	m,y	$\frac{m+x}{2},\frac{k+y}{2}$	m,y
$\langle 1,0\rangle$	$\frac{m+x}{2},\frac{k+y}{2}$	$\frac{m+x}{2},\frac{k+y}{2}$	x,k	x,k	x,k	x,k
$\langle 1,2\rangle$	m,y	m,y	x,k	m,y	x,k	$\frac{m+x}{2},\frac{k+y}{2}$
$\langle 2,0\rangle$	$\frac{m+x}{2},\frac{k+y}{2}$	$\frac{m+x}{2},\frac{k+y}{2}$	x,k	x,k	x,k	x,k
$\langle 2,1\rangle$	m,y	m,y	x,k	$\frac{m+x}{2},\frac{k+y}{2}$	x,k	x,k

Fig. 3: Payoff matrix for the same MSG as the one depicted in figure 2 with a different decision rule (an "egalitarian" mechanism).

4. The number of meetings. Note that the natural scenario of incremental addition of meetings defines a different game which is an iterated game.

p_1 p_2	$\langle 0,0,1\rangle$	$\langle 0,1,0\rangle$	$\langle 1,0,0\rangle$	$\langle 1,0,1\rangle$	$\langle 1,1,0\rangle$	$\langle 0,1,1\rangle$
$\langle 0,0,1\rangle$	x	$\frac{x+z}{2}$	$\frac{x+y}{2}$	x	$\frac{x+y+z}{3}$	x
$\langle 0,1,0\rangle$	$\frac{x+z}{2}$	z	$\frac{y+z}{2}$	$\frac{x+y+z}{3}$	z	z
$\langle 1,0,0\rangle$	$\frac{x+y}{2}$	$\frac{y+z}{2}$	y	y	y	$\frac{x+y+z}{3}$
$\langle 1,0,1\rangle$	x	$\frac{x+y+z}{3}$	y	$\frac{x+y}{2}$	y	x
$\langle 1,1,0\rangle$	$\frac{x+y+z}{3}$	z	y	y	$\frac{y+z}{2}$	z
$\langle 2,1\rangle$	x	z	$\frac{x+y+z}{3}$	x	z	$\frac{x+z}{2}$

Fig. 4: Payoff matrix of the row player (p_1) in a three time slot max-sum MSG. The player's ordinal preferences are $evening > morning > noon$

As an example of a simple change consider a three time slot MSG, with a Max-Sum mechanism. Let us assume that users are allowed to add any bid representing different preferences (that is, bids which have at least two different values such as $\langle B,0,0\rangle$). Let us also assume that in the case of a tie over several time slots, the mechanism uniformly selects one of the tied slots. Figure 4 depicts the gains of the row player (p_1) in such a game. The player prefers to have the meeting in the evening, and if that is not possible she would rather have it in the morning. She least prefers to have it at noon. That is, the agent's utility from a meeting will be z for the least preferred time slot, x for the most preferred time slot, and y for the remaining time slot.

It can be shown that if both of the following conditions are satisfied for such an MSG:

- $x + z > 2y$
- $x + y > 2z$

Then, the bid assigning B to the preferred time slot and 0 to the remaining time slots (denoted b) is a (possibly weak) dominant strategy for the players.

Proof. Similarly to our previous lemmas, one can see that any player playing the suggested bid will divert the game towards some sort of tie involving her most preferred time slot, or towards the time slot itself (her opponent will never exceed the value B on any of her other time slots). Following our requirements on payoffs we can sort the possible outcomes:

$$z < \frac{y+z}{2} < y < \frac{x+y+z}{3} < \frac{x+z}{2} < \frac{x+y}{2} < x$$

Since the highest outcomes are those involving the required time slot, to complete our proof one needs only to show that for any given bid by the opponent, playing the suggested bid will result in the highest outcome (or at least as high as any other outcome). This, however, is immediate by the bids structure: If a three slots tie exist, any bid other than b will break the tie resulting in a worst outcome for the player (one of its less preferred time slot). Similarly, if a two slots tie exists then changing b will either result in a three slots tie (which produces a lower quality solution for the player), or worst yet, it will result in a less preferred "pure" outcome.
Thus, playing b is always a (possibly weak) dominant strategy in the MSG described above.

One can think of a "real" MSG as a graphical game [6], in which each participant is connected with others who have at least one common meeting with her. While this formulation provides a more detailed view of a large scale interaction, it provides little theoretical insights. Furthermore, finding an optimum of such a graphical game is a task which received little attention until only recently [5].

4. The Cost of Cooperation

Following this detailed tour of the Price of Anarchy for MSGs, an interesting question arises: what exactly is the meaning of the PoA for MSGs? In other words, what is the meaning of "anarchy" in the context of MSGs. After all, "anarchy" for an MSG is the natural situation because agents want their best possible personal calendar. The "cooperative" method of optimizing some global goal is not very natural. Why would self interested users wish to optimize some global objective ? There is no sense of "public savings" or "minimization of overall cost", as in the case of routing games. Players in an MSG are only interested in cooperation as a form of mediation between them. This implies that examining efficiency in terms of some global value does not fit the agent's point of view.

The present paper proposes a different measure. Instead of addressing the "social" cost, the focus here is to quantify the cost (or gain) of a single agent, from participating in any cooperative protocol.

Definition 1. *An agent's* Cost of Cooperation*(CoC) with respect to $f(x)$ is defined as the difference between the lowest gain the agent can have in any of the NEs of the underlying game (if any, otherwise zero) and the lowest gain an agent receives from a (cooperative) protocol's solution x that maximizes $f(x)$.*

The intuition behind this definition is quite simple. A user about to interact with other users can *expect that the combined spontaneous action taken by all participants will result in a NE*. Since there can be more than one such stable points the user anticipates the worst case for herself, i.e., the personal lowest gaining NE. For her to join a cooperative protocol (and use a multi user distributed optimizing protocol), she uses knowledge about her possible payoffs. This information is provided by the CoC. When the CoC value is negative, the user knows that she *stands to gain* from joining the cooperative protocol.

As a simple example, consider the famous Prisoners' Dilemma game as depicted in Figure 5. In this problem, a single stable action profile exists - $\langle D, D\rangle$. When both participants select this action, each has a gain of 1. If one examines the socially optimal solution that maximizes the overall gain (e.g., the *sum of all personal gains*) - $\langle C, C\rangle$ - the gain for each agent is 4. This corresponds to a CoC of -3 for each participant, which implies that both agents gain from following this cooperative solution.

p_1 p_2	Cooperate	Defect
Cooperate	4, 4	0, 6
Defect	6, 0	1, 1

Fig. 5: The Prisoners' Dilemma game matrix

We define the $f(x)$-CoC for a set of users as a tuple, in which every element CoC_i corresponds to the CoC of agent A_i. Defining the CoC for an entire system can be used to motivate the participation of users in the cooperative protocol. A negative CoC (i.e., gain), provides a self-interested user with a strong incentive to join the cooperative protocol. One may even venture to think of this as a form of *Individual Rationality* [12].

4.1. Cooperation Games

Let us define special strategic situations in which the CoC vector of all participants has non positive values (we shall refer to these as CoC solutions).

Definition 2. *A game is a* Cooperation game[2] *if there exists a solution (strategy profile) to the game in which each player's payoff is as high as its worst equilibrium payoff. That is, there exists a solution for which the CoC (with respect to some $f(x)$) of all agents is non positive.*

Note that if there exists a solution to the game in which each player's payoff is as high as its worst equilibrium payoff, the objective function one may optimize will simply accept solutions in which the gains of each agent are higher than the worst NE gains, and reject all other solutions.

As an example consider the simple MSG of Figure 2. It is a cooperation game. If a cooperative mechanism optimizing the Max-Min objective function is used, all agents are guaranteed to have a non positive CoC (although this scenario may be treated as a degenerate one - all CoC values are zero).

Our definition of a cooperation game implies that any game with at least one PSNE is also a cooperation game (one can always take one of the worst NEs and present it as a solution with non positive CoC). However, we will usually focus on games in which the CoC is negative for all players.

It is important to note that a CoC solution does not imply stability. This can easily be understood by examining the prisoners' dilemma in Figure 5. An all-non-positive CoC solution in this case corresponds to the tuple $\langle C, C\rangle$. However, game theory does not treat this solution as a viable one since each player has a strong incentive to deviate from it. If on the other hand, players are treated as agents which follow a protocol (may not deviate from it) one need not worry about the stability of the solution. In this case, the agents' underlying users simply accept the cooperative framework since it guarantees a solution which is at least as good as that achieved by "playing out" the interaction *for each and every one of them*.

Figure 6 provides a visual representation of these ideas. In this figure, the space of possible outcomes of a given game is drawn. Each dot represents a possible outcome of the game along with the payoff each participant associates with that outcome. The social choice solution which maximizes the overall gain to participants is indicated in the bottom right corner ("Max-Sum"), while the set of all pareto efficient solutions is indicated by a line connecting these. A non positive CoC solution is any outcome situated in the colored area (it is easy to see why there always exists at least one pareto efficient solution within this area).

Although every game admits at least one (possibly mixed) stable point some games are not cooperation games. Consider for example the game in Figure 7. In this game, there exists only one stable solution (the mixed strategy $\langle\frac{1}{3}, \frac{1}{2}\rangle$). The expected outcome of each player in this mixed solution results in a payoff of 3 to each participant. However, there is no outcome with the desired property of non positive CoC to both players in this game.

Given a general game, one may want to change it into a cooperation game (if it is not one already). This can be achieved by changing the mechanism of

[2] Not to be confused with cooperative games

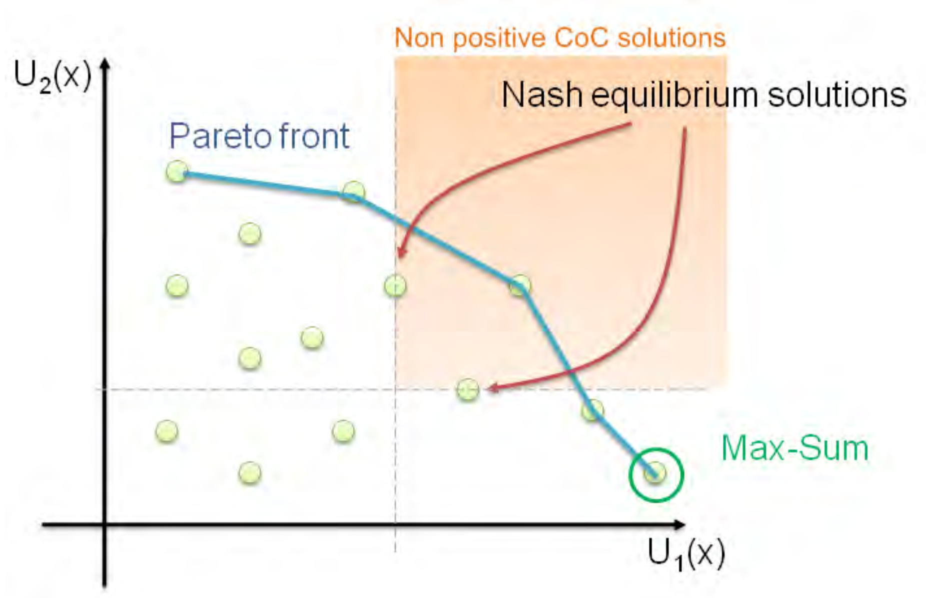


Fig. 6: A cooperation game

p_1 \ p_2	Left	Right
Up	2, 5	4, 1
Down	6, 1	0, 3

Fig. 7: A simple game. This game is not a cooperation game - the worst (expected) equilibrium outcome for each player is 3 and $\frac{7}{3}$, and no outcome can satisfy a non positive CoC tuple.

the interaction itself, or by adding an interested party to the interaction (that is, modify the interaction to introduce at least one PSNE). Some work in the direction of the latter approach that uses mediators was recently reported in [17, 11]. *Mediators* are introduced as parties wishing to influence the choice of action (e.g., strategy) of participants which are not under their control. Mediators cannot enforce payments by the agents or prohibit strategies, but can influence the outcome of an interaction by offering to play on behalf of some (or all) of the participants. By doing so a mediator commits to a pre-specified behavior [17].

An interesting property of Routing mediators [17] is that they are capable of possessing information about the actions taken by agents. While the authors of [11] apply mediators for the sake of a strong equilibrium, in our two player strategic situation, one may use the following routing mediator to generate a simple PSNE. The mediated game includes mediated actions and each agent may either play its game as before, or let the mediator play for it.

The following mediator is then used: the mediator may receive a message from the user asking it to play for her or not. When the mediator plays for an agent, it will always choose to assign the bid which will result in the minimal payoff to the agent's *opponent*. If both agents use the mediator, the mediator assigns the interaction which results in the lowest value that is at least as high as any value that results from the play of the agent's opponent.A player may always choose not to use the mediator (apply one of the original actions available to her).

p_1 p_2	Left	Right	Med
Up	2, 5	4, 1	2, 5
Down	6, 1	0, 3	0, 3
Med	6, 1	4, 1	4, 1

Fig. 8: A mediated version of the game in Figure 7

Figure 8 depicts the result of applying the above mediator to the game of Figure 7. In the new mediated game there are three new PSNE - $\langle Med, Left\rangle$, $\langle Med, Right\rangle$ and $\langle Med, Med\rangle$. Furthermore, since the new worst gains for the participant are 4 to player 1, and 1 to player two, the solution $\langle Down, Left\rangle$ has a non positive CoC tuple (player 1 gain's from the interaction is higher than her "expected" NE gain, while player 2's gain is not lower than her expected worst gain). The proposed mediator will always add at least one PSNE:

Theorem 1. *When applying the mediator described above to a two player game, the transformed interaction will include at least one PSNE in the strategy profile $\langle M, M\rangle$.*

Proof. The mediator described above always uses the lowest value to the opponent. Thus, the "M" column (row) is based on the lowest value of the row (column) player. Once the values are inserted to each cell but the last - $\langle M, M\rangle$ - one needs to show that there exists a joint action which has a left hand value as high as the highest value in all the left hand values of the "M" column, and a right hand value as high as the highest right hand value in the "M" row.

Let r be the row in the "M" column with the highest left hand value, and let us denote that value with x. Similarly, let c be the column in the "M" row with the highest right hand value - y. By the construction of the "M" column we know that x is the lowest value in row r. The same is true for y - according to the construction of row "M" we know that it is the lowest value in the column c. Examining the values in the intersection of row r and column c we conclude that the left hand value must be at least as high as x and the right hand value must be at least as high as y.

Since x and y are the highest values in the "M" column and row (respectively), the left (right) hand value of $\langle r, c\rangle$ is higher than all other values in the "M" column (row). As a result, adding these values to the cell indicated by $\langle M, M\rangle$, we receive a PSNE in that joint interaction.□

4.2. CoC solutions

As mentioned in the previous section, a CoC solution does not imply stability and should not be confused with solution concepts such as Aumann's strong Nash equilibrium [1]. A strong Nash equilibrium strives to achieve a solution which is deviation proof by any possible coalition of players. While such a concept has many desired properties it requires agents to rationalize over outcomes, does not provide any guarantees to individual players (except having lower gains in case of a deviation) and rarely occurs in most games. A CoC solution on the other hand is unstable in the game theoretic sense but as demonstrated in the previous section often exists in a multi agent interaction.

The approach taken by the present paper introduces cooperation, or a cooperative protocol, to a self interested interaction. This approach is different than that taken by previous works. Specifically, the attempt to introduce a party to a given game is extensively discussed in [11]. Although the mediator presented by Monderer and Tennenholtz is limited, its presence alters the initial interaction (as demonstrated). The changes introduced by a mediator alter the very course of the game. While a simple MSG includes two phases: action selection and payoff collection the introduction of a mediator can be translated as an additional interim phase. In this phase, the players wishing to employ the mediator notify it, a calculation is made by the mediator, and this information is sent onward to the game mechanism along with the rest of actions selected by those players who have not used the mediator. The game's mechanism then selects the time of the meetings. Using a cooperative protocol on the other hand is equivalent to stating that the players have no knowledge on how to choose their actions and are only interested in a solution which provides some guarantees to the quality of the solution. As such, they request the game mechanism to make its choice for them.

Providing guarantees to participants was also discussed in [18]. There, the author defines a mixed strategy t as a C competitive safety level strategy if the ratio between the expected payoff of the agent in a Nash equilibrium to its expected payoff in t is bounded by C. While we share the idea of providing a guarantee to participating agents, our approach examines a cooperative, multi agent approach which does not require stability. This is in contrast to the competitive safety level strategy introduced in [18] which provides a sound recommendation to each one of the participants separately.

Finally, it should be noted that the concept of CoC defines a subspace of the original search space. In this subspace (colored area of Figure 6) there may be several CoC solutions. That is, one may use existing solution concepts (such as the pareto front) over this area and still provide a strong incentive for players to join.

5. Related work

Research into distributed cooperative optimization has seen a growing interest in the past decade. Specifically, the large body of work on distributed algorithms for Distributed Constraint Optimization Problems (DCOPs) (cf. [9]) makes this approach suitable and widely accepted for formulating and solving many diverse MAS problems ([4, 20, 8]). DCOP agents exchange messages pretaining to their state and their valuation of it. The end goal of the agents is a globally optimal solution (a set of all agents' states). These agents are cooperative and will refrain from changing their state even if some consider it sub optimal (i.e., can improve their personal gain by changing their local state). A recent paper introduced a variation of DCOPs - Asymmetric DCOPs (ADCOP) [5] - which may be even better suited to problems such as the one discussed in our work. ADCOPs form a richer framework for cooperative optimization and allows search in game-like structures such as the one presented in Figures 4 and 8.

Cooperative optimization provides a rich set of algorithms for solving many problems, however, it is deemed ill fitting to many problems in which agents are mainly interested in their personal gain. In such cases, Game Theory is invoked (cf. [12]). It allows one to mathematically define a stable solution to an interaction involving a set of rational self interested agents. However, there has been very little work on distributed algorithms for finding such stable solutions [6].

The present work attempts to draw upon the advantages of both of these approaches. Although it considers a set of cooperative agents which may be thought of as a coalition (of all players), it is quite different from the coalitions discussed by Aumann's strong Nash equilibrium [1]. A strong Nash equilibrium strives to achieve a solution which is deviation proof by any possible coalition of players, but rarely exist. CoC solutions, on the other hand, provide no such guranatee, but are often present in many problems. Moreover, unlike other cooperative optimization approaches, CoC solutions maintain some form of Individual Rationality (IR).

An interesting approach is provided by Monderer and Tennenholtz in [11], who propose to alter a given interaction through a trusted third party - the mediator. Agents may communicate with the mediator which in turn provides some form of limited cooperation. However, unlike the present work which attempt to provide guarantees to participants, their work attempts to enrich the set of situations where stability against deviations by coalitions may be obtained.

Tennenholtz also attempts to provide guarantees to participants in [18]. Examining the ratio between the expected payoff in a NE to the expected payoff from a mixed strategy, a C competitive safety level strategy is defined. However, the end goal of this analysis provides a recommendation to each self interested participant. The method proposed by the present study assumes a cooperative protocol and is not limited to stable solutions.

The cooperative approach to MSPs attempts to maximize (minimze) a given objective function [19, 8, 10] and does not attempt to provide any guarantee on the outcome (or provide IR). In contrast, the competitive approach to MSPs attempts to design the mechanism of the interaction so that some stable solution is acheived [3, 2]. It assumes agents will interact in a selfish manner and end up in a stable state, although how agents may do so is not discussed and is often ignored. The method of the present paper is not limited to stable solutions, apply known cooperative algorithms, and provide some subjective guarantee to each participant.

6. Discussion

The present paper discusses two opposite extreme approaches that are inherent to many multi agent scenarios - cooperation and competition. In order to investigate these, a simple scheduling problem was formulated in the form of a simple game and analyzed. The maximal preference bid and general payoffs were incrementally added and their impact on the stable points and the PoA of the interactions were examined.

The analysis of a simple MSG leads naturally to a new measure that quantifies the cost/gain to agents from participating in a cooperative protocol. The Cost of Cooperation (CoC) as defined in the present paper further defines a game property that is termed “Cooperation game”. Participants in a cooperation game may be better off cooperating in a search protocol for an optimal solution than playing (selfishly) out the game.

These ideas may be applied to a large range of problems where participants are competitive but may need to reach some agreement. One may consider voting games, for example. In voting games, each agent has prefrences over the set of outcomes and a set of actions (votes for candidate solutions). A candidate solution is selected by the game mechanism based on a pre defined rule. The candidate solution which maximizes the overall utility (often referred to as the “social choice”) may also be one in which some agents receive a very low utility. Being self interested, agents will always choose to deviate from such solutions. On the other hand, stable solutions may result in a rather low overall satisfaction (utility). A CoC solution can be used to maximize the overall utility in a manner which assures agents’ gains are at least as high as those acheived in a fully competitive environment.

In the future our ideas can be used to identify classes of problems for which a simple objective function guarantees a non positive CoC for all players. Thus, one may use a distributed protocol for MAS and achieve better personal results than either letting participants play out the game, or search for a NE for them. We hope that through the CoC ideas we may be able to devise new objective functions which can provide guarantees on a wide set of problems.

7. Acknowledgment

The research was supported by the Lynn and William Frankel Center for Computer Sciences at Ben-Gurion University and by the Paul Ivanier Center for Robotics Research and Production Management.

References

1. Aumann, R.J.: Subjectivity and correlation in randomized strategies. Journal of Mathematical Economics 1(1), 67–96 (March 1974)
2. Crawford, E., Veloso, M.: Mechanism design for multi-agent meeting scheduling. Web Intelligence and Agent Systems 4(2), 209–220 (2006)
3. Ephrati, E., Zlotkin, G., Rosenschein, J.: A non manipulable meeting scheduling system. In: Proc. Intern. Workshop on Distributed Artificial Intelligence. Seatle, WA (1994)
4. Gershman, A., Grubshtein, A., Meisels, A., Rokach, L., Zivan, R.: Scheduling meetings by agents. In: Proc. 7th International Conference on Practice and Theory of Automated Timetabling (PATAT 2008). Montreal (August 2008)
5. Grubshtein, A., Grinshpoun, T., Meisels, A., Zivan, R.: Local search for distributed asymmetric optimization. In: Proc. of the 9th International Conference on Autonomous Agents and Multiagent Systems(AAMAS 2010). Toronto, Canada (May 2010)
6. Kearns, M.J., Littman, M.L., Singh, S.P.: Graphical models for game theory. In: UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence. pp. 253–260. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
7. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: in Proc. of the 16th Annual Symposium on Theoretical Aspects of Computer Science. pp. 404–413 (1999)
8. Maheswaran, R.T., Tambe, M., Bowring, E., Pearce, J.P., Varakantham, P.: Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling. In: Proc. 3rd Intern. Joint Conf. on Autonomous Agents & Multi-Agent Systems (AAMAS-04). pp. 310–317. NY, New York (2004)
9. Meisels, A.: Distributed Search by Constrained Agents: Algorithms, Performance, Communication. Springer Verlag (2007)
10. Modi, J., Veloso, M.: Multiagent meeting scheduling with rescheduling. In: Proc. 5th workshop on distributed constraints reasoning DCR-04. Toronto (September 2004)
11. Monderer, D., Tennenholtz, M.: Strong mediated equilibrium. Artificial Intelligence 173(1), 180–195 (2009)
12. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press (2007)
13. Papadimitriou, C.: Algorithms, games, and the internet. In: STOC '01: Proc. of the 33rd annual ACM symposium on Theory of computing. pp. 749–753 (2001)
14. Petcu, A., Faltings, B., Parkes, D.: M-DPOP: Faithful distributed implementation of efficient social choice problems. Journal of AI Research (JAIR) 32, 705–755 (2008)
15. Roughgarden, T.: Selfish Routing and the Price of Anarchy. The MIT Press (2005)
16. Roughgarden, T., Tardos, É.: How bad is selfish routing? J. ACM 49(2), 236–259 (2002)
17. Rozenfeld, O., Tennenholtz, M.: Routing mediators. In: IJCAI. pp. 1488–1493. Hyderabad, India (2007)
18. Tennenholtz, M.: Competitive safety analysis: robust decision-making in multi-agent systems. Journal of Artificial Intelligence Research 17(1), 363–378 (2002)
19. Wallace, R.J., Freuder, E.: Constraint-based multi-agent meeting scheduling: effects of agent heterogeneity on performance and privacy loss. In: Proc. 3rd workshop on distributed constrait reasoning, DCR-02. pp. 176–182. Bologna (2002)
20. Zhang, W., Xing, Z., Wang, G., Wittenburg, L.: Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. Artificial Intelligence 161:1-2, 55–88 (January 2005)

Alon Grubshtein is a PhD student in the department of Computer Science at Ben-Gurion University working under the supervision of Prof. Amnon Meisels. His research is focused on competitive and cooperative agents and Disrtibuted Constraint Reasoning. Alon Grubshtein is currently employeed in Deutsche Telekom's Labarotories at Ben-Gurion University as a junior researcher.

Amnon Meisels has been a faculty member of the department of Computer Science at Ben-Gurion University for the last 20 years. Served as head of the Computer Science dept. from 2004 to 2006. The main field of research of Amnon Meisels is Constraints Processing and in the last 7 years mostly Distributed Constrained Search. His book "Distributed Search by Constrained Agents", was published by Springer Verlag in January 2008. Amnon Meisels has been part of the organizing committee of the series of workshops on distributed constraints reasoning (DCR series) every year since 2003. Important research milestones of the DisCSP group at BGU that is headed by Prof. Meisels in the last 7 years include: (a) *Asynchronous forward-checking Concurrent search - algorithms for DisCSPs*, (b) *Asynchronous ordering heuristics for DisCSP algorithms*, (c) *Concurrent run-time measures for experimental evaluation of DisCSP algorithms*, (d) *Privacy preserving search methods for DisCSPs*, and (e) *Asynchronous Forward-bounding for DisCOPs - establishing the phase transition*

Received: January 18, 2010; Accepted: May 3, 2010.

UDC 519.8:004.4, DOI:10.2298/CSIS100209016D

TEAMLOG in Action: a Case Study in Teamwork

Barbara Dunin-Kęplicz[1][2]*, Rineke Verbrugge[3]**, and Michał Ślizak[4]

[1] Institute of Informatics,
Warsaw University,
Banacha 2, 02-097 Warsaw, Poland
keplicz@mimuw.edu.pl
[2] Institute of Computer Science,
Polish Academy of Sciences,
Ordona 21, 01-237 Warsaw, Poland
Barbara.Dunin-Keplicz@ipipan.waw.pl
[3] Department of Artificial Intelligence,
University of Groningen,
P.O. Box 407, 9700 AK Groningen, The Netherlands
rineke@ai.rug.nl
[4] Institute of Informatics,
Warsaw University,
Banacha 2, 02-097 Warsaw, Poland,
Michal.Slizak@mimuw.edu.pl

Abstract. This article presents a case study of a theoretical multi-agent system designed to clean up ecological disasters. It focuses on the interactions within a heterogeneous team of agents, outlines their goals and plans, and establishes the necessary distribution of information and commitment throughout the team, including its sub-teams. These aspects of teamwork are presented in the TEAMLOG formalism [20], based on multi-modal logic, in which collective informational and motivational attitudes are first-class citizens. Complex team attitudes are justified to be necessary in the course of teamwork. The article shows how to make a bridge between theoretical foundations of TEAMLOG and an application and illustrates how to tune TEAMLOG to the case study by establishing sufficient, but still minimal levels for the team attitudes.

Keywords: TEAMLOG, motivational attitudes, multi-agent modelling.

1. Defining teamwork

When modeling complex automated systems it is often easier to think of them as a collection of agents, each capable of making its own decisions, rather than collections of modules with multiple, often unclear interdependencies. The notion of agency that is essential in the field of multi-agent systems (MAS) allows the designer to skip over the complexities of relationships between components,

* Supported by the MNiSW grant N N206 399334.
** Supported by the NWO Vici grant 016-094-603.

while still having the possibility to prove all required characteristics of the system, for example *liveness* and *correctness*. One of the most advanced models is the Belief-Desire-Intention (BDI) model of agency.

The BDI agent model comprises agents' individual beliefs, goals, and intentions. However, when a group of agents needs to work together in a planned and coherent way, agents' individual attitudes are not sufficient; the team needs to present a collective attitude over and above individual ones. Therefore, when constructing BDI systems, firstly a model of an agent as an *individual, autonomous* entity [35] has to be constructed. Then, a key point is to organize agents' cooperation in a way allowing the achievement of their common goal, while preserving, at least partly, their individual autonomy (see [31, 28, 42, 14, 16, 25, 34, 20] for some logical approaches to teamwork).

1.1. TEAMLOG **applied**

As a *static, descriptive* theory of collective motivational attitudes, TEAMLOG has been formed on the basis of individual goals, beliefs and intentions of strictly cooperative agents [14, 16, 22, 20]. It addresses the question what it means for a group of agents to have a *collective intention*, and then a *collective commitment* to achieve a common goal. While collective intention transforms a loosely-coupled group into a strictly cooperating team, collective commitment leads to team action, i.e., to coordinated realization of individual actions by committed agents according to a plan. The social plan can be constructed from first principles, or may be chosen from a repository of pre-constructed plans. Both collective intentions and collective commitments allow to fully express the potential of strictly cooperative teams [14, 16]. The bilateral and collective notions cannot be viewed as a sort of sum of individual ones.

In this case study we focus on *disaster response* and, more specifically, on decontamination of a certain polluted area. Our goal is to illustrate how to adjust a multi-agent system model created using the TEAMLOG theory to this specific use-case, with a focus on the theoretical performance of a potential implementation. Our contribution is to show how the costly infinitary definitions of collective attitudes can be reduced in a real-world situation, while maintaining the team cohesion necessary to achieve the goals. Whence the title "TEAMLOG in action" and the focus on full cooperation, also in a complex case, reflected in agents' individual, social (i.e. bilateral) and collective attitudes. Cases of competition are explicitly excluded by TEAMLOG definitions.

A theory of individual and group beliefs has been formalized in terms of epistemic logic [23, 32, 33]. General, common, and distributed knowledge and belief were defined in terms of agents' individual knowledge and belief. Different axiom systems express various properties of knowledge and belief, while the corresponding semantics naturally reflect these properties.

When modeling group attitudes, agents' awareness about the overall situation needs to be taken into account. Awareness is understood here as a limited form of consciousness: in a way typical for MAS, it refers to the state of an agent's beliefs about *itself* (*intra-personal*), about *others* (*inter-personal*) and about *the environment* (*group awareness*). Thus, various epistemic logics and different gradations of group information (from distributed belief to common knowledge) are adequate to formalize agents' awareness [23, 16, 33].

In TEAMLOG, group awareness is naturally expressed in terms of *common belief* (C-BEL_G), fully reflecting collective aspects of agents' behavior. Due to its infinitary flavor, this concept has a high complexity: its satisfiability problem is EXPTIME-complete [22]. There are general ways to reduce the complexity by restricting the language, by allowing only a small set of atomic propositions or restricting the modal context in formulas, as proved in [22, 21]. However, when building MAS applications, it may be more profitable to use domain-specific means to tailor TEAMLOG to the circumstances in question, calling for weaker forms of awareness [18]. In this case study we will show how to adjust team attitudes to ensure a lower complexity.

Collective intention constitutes a basis for preparing a plan, reflected finally in *collective commitment*. Various versions of collective commitments are applicable in different situations, dependent on organizational structure, communicative and observational possibilities and so on. These cause differences in agents' awareness both with respect to the *aspects* of teamwork of which they are aware, and the *kind* of awareness present within a team, ranging from distributed and individual belief, up to common knowledge. In this context, plan representation is a separate issue. It is assumed that any method of describing plans is acceptable, as long as it states the sequences of actions to be performed and allows us to project these actions onto individual agents. While a formal definition of social plan is available in TEAMLOG, the syntax used in the case study is considerably richer, and still very intuitive. It can easily be translated into combinations of definitions presented in [19, 15].

1.2. From real-world data to teamwork

Formal approaches to multi-agent systems are concerned with equipping software agents with functionalities for reasoning and acting. The starting point of most of them is the layer of beliefs, in the case of TEAMLOG extended by goals, intentions and commitments. These attitudes are usually represented in a symbolic, qualitative way. However, one should view this as an idealization. After all, agent attitudes originate from real-world data, gathered by a variety of sources at the *object level* of the system. Mostly, the data is derived from sensors responsible for perception, but also from hardware, different software platforms, and last but not least, from people observing their environment. The point is that this information is inherently quantitative. Therefore one deals with a meta-level duality: sensors provide quantitative characteristics, while reasoning tasks performed at the *meta-level* require the use of symbolic representations and inference mechanisms.

Research in the framework of TEAMLOG is structured along the lines depicted in Figure 1. The object-level information is assumed to be summarized in queries returning Boolean values. This way it is possible to abstract from a variety of formalisms and techniques applicable in the course of reasoning about

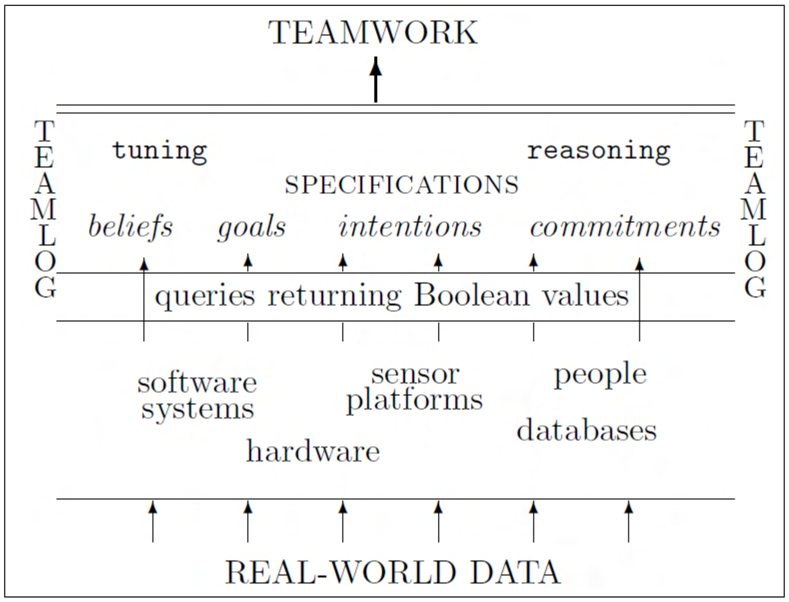


Fig. 1. The object- and meta-level view on teamwork

real-world data. This abstraction is essential, since the focus of TEAMLOG is on the meta-level, including formal specification and reasoning about teamwork.

This paper is structured into several sections. The first one introduces TEAMLOG and the problem in general terms. Next, in section 2, some definitions and assumptions regarding the environment are presented, including an outline of the interactions within a team and between sub-teams. This is followed in section 3 by definitions of social plans to be executed. In section 4 we take a closer look at the actions defined in our plan library and automated planning. In section 5 the TEAMLOG theory is presented. In section 6 we explore the minimal requirements for successful teamwork arising from the case study. Section 7 discusses related work, followed by a conclusion which sums up the article.

2. The case study: ecological disasters

Disaster management is a broad discipline related to dealing with and avoiding risks [41]. This involves several important tasks: *preparing for disaster* before it occurs, *disaster response* (e.g. emergency evacuation and decontamination), and *restoration* after natural or human-made disasters have occurred. In general, disaster management is the continuous process by which all individuals, groups, and communities manage hazards in an effort to avoid or ameliorate the impact of disasters resulting from them. Actions taken depend in part on perceptions of risk of those exposed [6]. Activities at each level (individual, group, community) affect the other levels.

The case study deals with response to ecological disasters caused by specific poisons, by means of heterogeneous multi-agent teams. The use-case is theoretical, in order to focus on those aspects of the problem that influence team action. The teams work in situations where time is critical and resources are scarce; it has been shown that teams consisting of software agents, robots and humans are a good choice in such critical situations [30, 37]. The example has been constructed by the authors, who took inspiration from the literature on disaster response [6, 41], heterogenous teams [30, 37], and unmanned aerial vehicles [8, 7].

Here, the main goal ($safe$) of teamwork is maintaining a given region REG safe or to return it to safety if it is in danger. The possible threats are two kinds of poison, X_1 and X_2, which are dangerous in high concentrations. The situation when both toxins mix is extremely hazardous, because if their concentration is high enough, they will react to form a chemical compound. This compound ($X_1 \oplus X_2$), once created, may explode if temperature and air pressure are high. Three functions f_1, f_2 and f_3 reflect the influence of temperature $t(A)$, air pressure $p(A)$ and concentrations $c_1(A)$ and $c_2(A)$ of poisons X_1 and X_2 at location A on the possible danger level at that location. The function ranges are divided into three intervals, as follows:

The first poison X_1:
- *safe*$_1$ iff $f_1(p(A), t(A), c_1(A)) \in [0, v_1]$;
- *risky*$_1$ iff $f_1(p(A), t(A), c_1(A)) \in (v_1, n_1]$;
- *dangerous*$_1$ iff $f_1(p(A), t(A), c_1(A)) \in (n_1, \infty)$;

The second poison X_2:
- *safe*$_2$ iff $f_2(p(A), t(A), c_2(A)) \in [0, v_2]$;
- *risky*$_2$ iff $f_2(p(A), t(A), c_2(A)) \in (v_2, n_2]$;
- *dangerous*$_2$ iff $f_2(p(A), t(A), c_2(A)) \in (n_2, \infty)$;

The chemical compound $X_1 \oplus X_2$:
- *safe*$_3$ iff $f_3(p(A), t(A), c_1(A), c_2(A)) \in [0, v_3]$;
- *risky*$_3$ iff $f_3(p(A), t(A), c_1(A), c_2(A)) \in (v_3, n_3]$;
- *explosive* iff $f_3(p(A), t(A), c_1(A), c_2(A)) \in (n_3, \infty)$;

The proposition $safe$ is defined as $safe :=$ *safe*$_1 \wedge$ *safe*$_2 \wedge$ *safe*$_3$ and is also referred to as a goal. There are also thresholds ϵ_1 and ϵ_2: when the concentration of a poison X_i exceeds ϵ_i, the respective function f_i is computed.

2.1. Starting point: the agents

This model reflects cooperation between humans, software agents, robots, unmanned aerial vehicles ($UAVs$) as discussed in [8, 7], and a helicopter steered by a pilot.

The whole process is controlled by one $coordinator$, who initiates cooperation, coordinates teamwork between different sub-teams of the full team G, is responsible for dividing the disaster zone into sectors and assigning a sub-team to each sector to perform clean-up. Several sub-teams $G_1, \ldots G_k \subseteq G$ of similar make-up work in parallel, aiming to prevent or neutralize a contamination. Each of these sub-teams G_i consists of:

- one UAV_i - responsible to the coordinator for keeping assigned sectors safe. This agent cannot carry a heavy load, but it carries a computer, therefore it has considerable computational capabilities for planning and is capable of observing and mapping the terrain;
- n_i identical neutralizing robots $rob_{i_1}, \ldots, rob_{i_{n_i}}$ - responsible to their UAV_i for cleaning up a zone.

Next to the sub-teams there is a rather independent member of G:

- one regular helicopter steered by the human $pilot$, who can independently choose the order of cleaning up assigned areas, is directly accountable to the coordinator and can communicate as an equal with the $UAVs$.

See Figure 2 for the team structure.

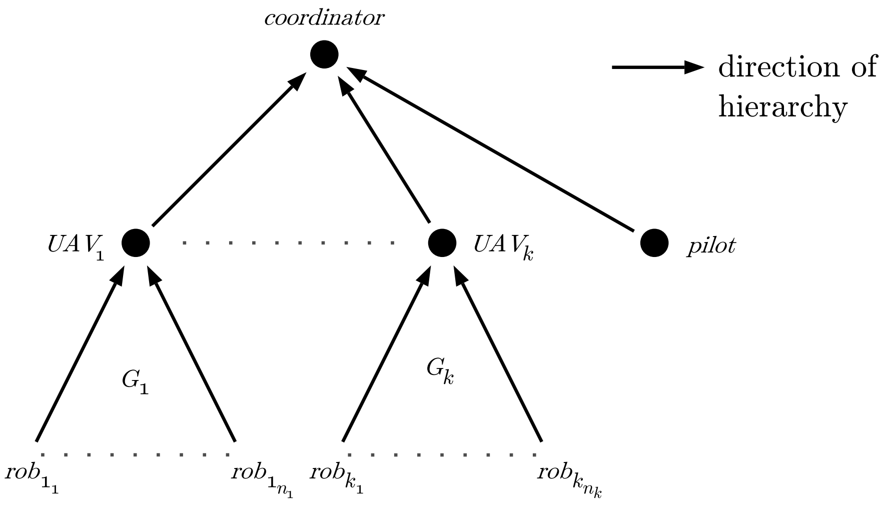


Fig. 2. Hierarchical team structure of the ecological disaster management team G.

2.2. Cooperation between sub-teams

The entire disaster zone is divided into sectors by the coordinator, based on terrain type, sub-team size, population density and hot spots known in advance (see Figure 3). Sub-teams are responsible for (possibly many) sectors. The leader UAV_i of a sub-team G_i prepares a plan to keep its sectors safe. Each plan is judged based on a fitting function fit, which takes into account:

1. available robots, including their current task, load, capacity and position;
2. whether the plan relies on the help from other sub-teams;
3. task priorities;
4. the minimum amount of time it takes to implement;
5. the minimum number of robots it requires.

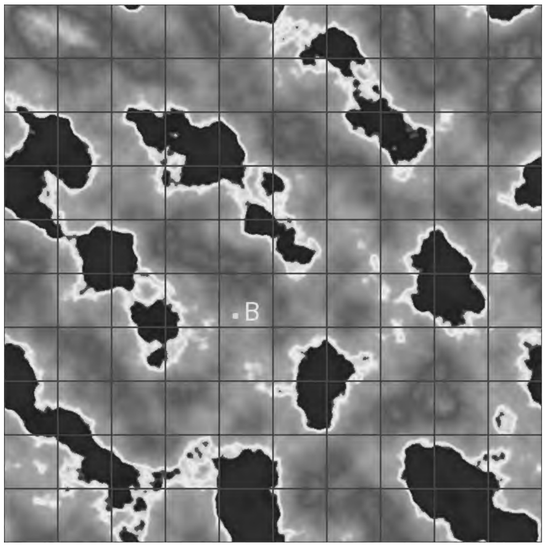


Fig. 3. Example terrain split, with base for robots B marked.
The darkest regions consist of water and are an obstacle to the cleanup robots.

The $UAVs$ communicate and cooperate with one another. If performing tasks requires more robots than are currently available in their own sub-team G_i, its leader UAV_i can call for reinforcements from another UAV_j, for $j \leq k, j \neq i$. Of course for UAV_j in question, fulfilling its own sub-team G_j's objectives has priority over helping others from G_i.

2.3. A bird's-eye view on cases

To maintain the goal $safe$, the situation is monitored by the coordinator and the $UAVs$ on a regular basis with frequency *freq*. During *situation recognition*, in the risky cases monitoring is performed twice as frequently. Depending on the mixture and density of poisons in a location, some general cases followed by the relevant procedures are established. All remedial actions are to be performed relative to the contaminated area:

Case $\boldsymbol{safe}$**:** True $\longrightarrow$ *situation recognition*

Case *dangerous*$_{\mathbf{1}}$**:** *rain* $\longrightarrow$ *liquid L*$_1$ *to be poured on the soil*
normal or dry $\longrightarrow$ *liquid L*$_2$ *to be sprayed from the air*

Case *dangerous*$_{\mathbf{2}}$**:** *rain* $\longrightarrow$ *solid S*$_1$ *to be spread*, followed by *liquid catalyst K*$_1$ *to be poured*
normal or dry $\longrightarrow$ *solid S*$_1$ *to be spread*

Case *explosive*: *before explosion* $\longrightarrow$ *evacuation*
after explosion $\longrightarrow$ *rescue action*

3. Global plans

While team planning is a very interesting problem, the algorithms for solving it are exponential for most cases. For performance reasons it was decided to use a pre-compiled plan library (consisting of plans devised by us) in the case study. Please observe that while the existence of such a library simplifies planning, it doesn't eliminate the need for it. Some actions (e.g. $move$) presented in the plan library are complex, and can only be solved with specialized planning techniques.

The ad-hoc notation representing high-level plans can easily be translated into basic definitions introduced in [19, 15]. The only requirement in TEAMLOG is that the plan describes the order of actions and their performers. Furthermore, TEAMLOG doesn't enforce any representation of time in plans. A sensible implementation of a planner designed to work in a physical environment will have to synchronize agents' actions, but the specific representation of time is not an issue at the level of abstraction used here. Clearly, the planner will be able to obtain temporal dependencies between actions from the presented description.

In order to control the amount of interactions and decrease the time needed to establish beliefs, we introduce a hierarchical team model. The coordinator views a sub-team G_i as a single entity, even though the $UAVs$ manage the work of many autonomous neutralizing robots. All agents are initially located at a base B, which also houses chemicals required for decontamination. It may interact with agents via communication protocols that would guarantee that agents do not compete for resources such as decontaminants.

3.1. The social plan $\langle \boldsymbol{Cleanup} \rangle$

The global social plan for which the coordinator and $UAVs$ are responsible, is designed with respect to location A. It is a while-loop, in which observation is interleaved with treatment of current dangers by level of priority, from most to least dangerous. The goal (denoted as $Clean$) is to keep locations in a $safe$ state. All subplans mentioned in $\langle Cleanup \rangle$, namely $\langle$ Plan $SR\rangle$, $\langle$ Plan E $\rangle$,

$\langle$ Plan D_1R $\rangle$, $\langle$ Plan D_1N $\rangle$, $\langle$ Plan D_2R $\rangle$, and $\langle$ Plan D_2N $\rangle$ are described more precisely in the subsequent subsections.

```
begin
  freq := $a$;
  $\{freq$ - interval between two checks of the environment$\}$
  while true do
    $\langle$ Plan $SR\rangle$ $\{$Assess the situation at $A$, with frequency freq$\}$
    if explosive then do $\langle$ Plan $E$ $\rangle$ end;
    elif dangerous$_1$ and rain then do $\langle$ Plan $D_1R$ $\rangle$ end;
    elif dangerous$_1$ then do $\langle$ Plan $D_1N$ $\rangle$ end;
    elif dangerous$_2$ and rain then do $\langle$ Plan $D_2R$ $\rangle$ end;
    elif dangerous$_2$ then do $\langle$ Plan $D_2N$ $\rangle$ end;
    elif risky$_1$ $\vee$ risky$_2$ $\vee$ risky$_3$ then freq := $\frac{a}{2}$ end
    else $\{$safe situation$\}$ freq := $a$ end;
  end
end.
```

Here, a represents the default frequency of checking the environment. This interval is shortened when a risky situation is encountered, to handle such a case with more caution. Dividing the interval by 2 is done to make the example specific; it is easy to adapt the plan to frequency adjustments applicable in a real-world scenario.

3.2. The social plan $\langle SR\rangle$

This plan performs situation recognition at location A. One of the $UAVs$, for example UAV_1, is responsible for monitoring. Alternatively, this task could be assigned as a joint responsibility to $UAV_1, \ldots, UAV_k$. However, that solution would require information fusion which is in general a very complex process.

```
begin
  $C_1 := c_1(A)$ $\{C_1$ is the measured concentration of poison $X_1$ at $A\}$
  $C_2 := c_2(A)$ $\{C_2$ is the measured concentration of poison $X_2$ at $A\}$
  $T := t(A)$ $\{T$ is the measured temperature at $A\}$
  $P := p(A)$ $\{P$ is the measured air pressure at $A\}$
  $\{$Computation of the situation at $A\}$
  if $C_1 > \epsilon_1$ then compute $f_1(C_1, T, P)$ end;
  if $C_2 > \epsilon_2$ then compute $f_2(C_2, T, P)$ end;
  if $C_1 > \epsilon_1$ and $C_2 > \epsilon_2$ then compute $f_3(C_1, C_2, T, P)$ end;
end.
```

3.3. The social plan $\langle E \rangle$

After an explosion, evacuation and rescue of people should take place. This subject is discussed in many studies [6, 41, 30] and will not be elaborated here. Instead, other subplans included in $\langle Cleanup \rangle$ are presented.

3.4. The social plan $\langle D_1 R \rangle$

This plan is applicable when *dangerous*$_1$ occurs under weather condition *rain*. Each UAV_i may be allocated this social plan for a given location as decided by the *coordinator*.
Goal $\psi_1(L_1)$: to apply liquid L_1 on all areas contaminated with poison X_1.

{Assumption: One portion of L_1 neutralizes poison X_1 at a single location.}

```
while $contaminated\text{-}area \neq emptyset$ do
begin
  $A := calculate(UAV_i, \{rob_{i_j}\})$;
  {$UAV_i$ finds region $A$ for $rob_{i_j}$ to clean up}
  $get(rob_{i_j}, L_1, B)$; {$rob_{i_j}$ retrieves a tank with liquid $L_1$ from location $B$}
  $path := get\_path(UAV_i, rob_{i_j}, B, A)$; {$rob_{i_j}$ requests a path to follow}
  $move(rob_{i_j}, path)$; {$rob_{i_j}$ moves from location $B$ to location $A$}
  $pour(rob_{i_j}, L_1, A)$;
  $contaminated\text{-}area := contaminated\text{-}area \setminus A$;
  $return\_path := get\_path(UAV_i, rob_{i_j}, A, B)$;
  $move(rob_{i_j}, return\_path)$;
end.
```

Here and in subsequent social plans, the assumption about needing a single portion of a liquid (or of solid and catalyst, respectively) per location is meant as an illustrative example. Functions computing the needed amounts of these substances from the measured concentrations of poisons could be used in a real-world case. It is also assumed that UAV_i establishes priorities of locations to be cleaned when planning the robots' routes, taking into account that some robots should start clearing certain paths such that other robots can then safely traverse these.

3.5. The social plan $\langle D_1 N \rangle$

This plan is applicable when *dangerous*$_1$ occurs under weather condition *normal or dry*. The spraying is usually performed by the pilot on request from one of the $UAVs$. In the plan below, UAV stands for any of $UAV_1, \ldots, UAV_k$.
Goal $\psi_2(L_2)$: to spray liquid L_2 on areas contaminated with poison X_1.

{Assumption: One portion of L_2 neutralizes poison X_1 at a single location.}
{Assumption: The helicopter can transport k portions of liquid L_2.}

```
while $contaminated\text{-}area \neq emptyset$ do
begin
  $request(UAV, coordinator, pilot, \psi(L_2))$;
  $confirm(pilot, UAV, coordinator, \psi(L_2))$;
  $request(pilot, UAV, list_1, k)$;
  $send(UAV, pilot, list_1)$; {$list_1$ has at most $k$ contaminated areas}
  $upload(helicopter, L_2, |list_1|)$; {$pilot$ retrieves required amount of liquid $L_2$}
  $take\text{-}off(helicopter, B)$; {$pilot$ takes off from location $B$}
  do $\langle plan\text{-}for\text{-}spraying(helicopter, L_2, list_1)\rangle$;
  {$pilot$ sprays $L_2$ using his own invented plan}
  $confirm(pilot, UAV, done(plan\text{-}for\text{-}spraying(helicopter, L_2, list_1))$;
  $contaminated\text{-}area := contaminated\text{-}area \setminus list_1$;
  $landing(helicopter, B)$;
  $free(pilot, coordinator)$;
end.
```

3.6. The social plan $\langle D_2R\rangle$

This plan is applicable when *dangerous*$_2$ occurs under weather condition *rain*. Goal $\psi_3(S_1, K_1)$: to spread solid S_1 on all areas contaminated with poison X_2, followed by applying catalyst K_1 to all areas where S_1 is present.

{Assumption: One portion of S_1 and K_1 neutralize poison X_2 at a single location.}

```
while $contaminated\text{-}area \neq emptyset$ do
begin
  $A := calculate(UAV_i, \{rob_{i_j}, rob_{i_l}\})$;
  {$UAV_i$ finds region for $rob_{i_j}$ and $rob_{i_l}$ to spread solid and catalyst,
  respectively.}

  begin_parallel {two main operations are done in parallel:
  applying a solid to the area, and pouring a catalyst on it }
    {a plan similar to $\langle D_1R\rangle$, but using $S_1$:}
    $get(rob_{i_j}, S_1, B)$; {$rob_{i_j}$ retrieves a portion of solid $S_1$ from location $B$}
    $path := get\_path(UAV_i, rob_{i_j}, B, A)$; {$rob_{i_j}$ requests a path to follow}
    $move(rob_{i_j}, path)$; {$rob_{i_j}$ moves from location $B$ to location $A$}
    $pour(rob_{i_j}, S_1, A)$;
    $return\_path := get\_path(UAV_i, rob_{i_j}, A, B)$;
    $move(rob_{i_j}, return\_path)$;
  ||
    $get(rob_{i_l}, K_1, B)$;
    $path := get\_path(UAV_i, rob_{i_l}, B, A)$;
    $move(rob_{i_l}, path)$;
    $wait\_for(spread(S_1, A))$; {$rob_{i_l}$ waits for *someone* to spread $S_1$ in A}
    $pour(rob_{i_l}, K_1, A)$;
    $return\_path := get\_path(UAV, rob_{i_l}, A, B)$;
    $move(rob_{i_l}, return\_path)$;
  end_parallel
  $contaminated\text{-}area := contaminated\text{-}area \setminus A$;
end.
```

The planner executed by UAV_i computes the paths in such a way that the robots' movements are synchronized, so that they arrive in area A almost simultaneously.

3.7. Plan $\langle D_2N \rangle$

This plan, very similar to $\langle D_1R \rangle$, is applicable when *dangerous*$_2$ occurs under weather condition *normal or dry*. Each UAV_i may be allocated $\langle D_2N \rangle$ for a given location as decided by the *coordinator*. Goal $\psi_1(S_1)$ is to apply solid S_1 on all areas contaminated with poison X_2.

{Assumption: One portion of S_1 neutralizes poison X_2 at a single location.}

```
while $contaminated\text{-}area \neq emptyset$ do
begin
  $A := calculate(UAV_i, \{rob_{i_j}\})$; {$UAV_i$ finds region $A$ for $rob_{i_j}$ to clean up}
  $get(rob_{i_j}$, $S_1$, B); {$rob_{i_j}$ retrieves a portion of solid $S_1$ from location $B$}
  $path := get\_path(UAV_i, rob_{i_j}, B, A)$; {$rob_{i_j}$ requests a path to follow}
  $move(rob_{i_j}, path)$; {$rob_{i_j}$ moves from location $B$ to location $A$}
  $pour(rob_{i_j}, S_1, A)$;
  $contaminated\text{-}area := contaminated\text{-}area \setminus A$;
  $return\_path := get\_path(UAV_i, rob_{i_j}, A, B)$;
  $move(rob_{i_j}, return\_path)$;
end.
```

The decontamination action is usually successful. Unfortunately, its effects may not be immediately visible, therefore the plans don't check contamination levels right after applying chemicals.

Even though $\langle D_1R \rangle$, $\langle D_1N \rangle$, $\langle D_2R \rangle$, and $\langle D_2N \rangle$ do not contain explicit actions to check the success of decontamination, such checks are part and parcel of the main $\langle Cleanup \rangle$ plan, so the coordinator and $UAVs$ can re-plan accordingly if decontamination does not succeed.

4. Defining actions

The actions are defined using the standard STRIPS syntax and semantics:

1. For every $i \in G$, A_i is the set of actions that agent i is capable of performing;
2. Each action $a \in A = \bigcup_{i \in G} A_i$ is a tuple $\langle \alpha, \beta, \gamma, \delta \rangle$, where:

α is the set of all conditions that must be true for the action to be executable;
β is the set of all conditions that must be false for the action to be executable;
γ is the set of all conditions that become true as a result of this action;
δ is the set of all conditions that become false as a result of this action.

4.1. Actions in the case study

Some actions are complex and require further planning. Definitions of the robots' actions are as follows:

- $get(Ag, Ob, From) =$
  $\langle\{at(Ag, From), at(Ob, From), available(Ob), capable\_of\_lifting(Ag, Ob)\},$
  $\emptyset, \{has(Ag, Ob)\}, \{available(Ob), at(Ob, From)\}\rangle$
- $move(Ag, From, Direction) =$
  $\langle\{at(Ag, From), in\_direction(From, Direction, X), can\_enter(Ag, X)\},$
  $\emptyset, \{at(Ag, X)\}, \{at(Ag, From)\}\rangle$
- $wait\_for(Cond) = \langle\emptyset, \emptyset, \{Cond\}, \emptyset\rangle$
- $pour(Ag, Chem, Loc) =$
  $\langle\{at(Ag, Loc), contains(Ob, Chem), has(Ag, Ob)\}, \emptyset, \emptyset, \{has(Ag, Ob)\}\rangle$

The complex action $move(Ag, From, To)$ results from planning performed by agent Ag (it might request terrain information from its controlling UAV in the process). It will be implemented as a series of atomic actions of the form $move(Ag, From, Direction)$.

Definitions of the pilot's actions:

- $upload(helicopter, Chem, Num) =$
  $\langle\{at(helicopter, B)\},$
  $\{airborne(helicopter)\}, \{has(helicopter, Chem, Num)\}, \emptyset\rangle$
- $take\text{-}off(helicopter, Loc) =$
  $\langle\{at(helicopter, Loc)\},$
  $\{airborne(helicopter)\}, \{airborne(helicopter)\}, \emptyset\rangle$
- $landing(helicopter, Loc) =$
  $\langle\{at(helicopter, Loc), airborne(helicopter)\},$
  $\emptyset, \emptyset, \{airborne(helicopter)\}\rangle$
- $plan\text{-}for\text{-}spraying(helicopter, Chem, \textit{list}_1) =$
  $\langle\{has(helicopter, Chem, X), X \geq |\textit{list}_1|,$
  $capable\_of\_lifting(helicopter, Chem, |\textit{list}_1|), airborne(helicopter)\},$
  $\emptyset, \{has(helicopter, Chem, X)\}, \{has(helicopter, Chem, Y), Y \equiv X - |\textit{list}_1|\}\rangle$

Here $plan\text{-}for\text{-}spraying(helicopter, Chem, \textit{list}_1)$ is considered an atomic action by the planner, as the pilot is an independent agent planning for himself. Notation $|\textit{list}_1|$ stands for the length of $\textit{list}_1$.

4.2. Plans and plan projections

A *social plan* is defined as a composition of simple social plan expressions.

SP1 $\langle a, i\rangle$ is a (simple) social plan expression iff $i \in G \wedge a \in A_i$

SP2 if α and β are social plan expressions and φ is any true/false statement then:
- $\langle \alpha; \beta\rangle$ is a (compound) social plan expression (sequential execution)
- $\langle \alpha || \beta\rangle$ is a (compound) social plan expression (parallel execution)
- $\langle$ if φ then do α end $\rangle$ is a (compound) social plan expression (an if-then statement)

The notation ξ is used to refer to an empty plan:

- $\langle \alpha; \xi\rangle \equiv \alpha$
- $\langle \xi; \beta\rangle \equiv \beta$
- $\langle \alpha || \xi\rangle \equiv \alpha$
- $\langle \xi || \beta\rangle \equiv \beta$
- $\langle$ if φ then do ξ end $\rangle \equiv \xi$

A plan is written for roles to be adopted by agents. Each role has requirements assuring, for example, that a robot cannot assume the role of a pilot (since it is not capable of flying a helicopter). A plan projection for agent i can be understood as the individual view of the plan, where some of the roles have been adopted by agent i. An *agent i's projection of a social plan P* (denoted $P|_i$) is defined as:

- $\langle a, i\rangle|_j \equiv \langle a, i\rangle$ iff $i = j$
- $\langle a, i\rangle|_j \equiv \xi$ iff $i \neq j$
- $\langle \alpha; \beta\rangle|_i \equiv \langle \alpha|_i; \beta|_i\rangle$
- $\langle \alpha || \beta\rangle|_i \equiv \langle \alpha|_i || \beta|_i\rangle$
- $\langle$ if φ then do α end $\rangle|_i \equiv \langle$ if φ then do $\alpha|_i$ end $\rangle$

A projection of a social plan P for a group G of agents can be defined in a similar fashion, using $i \in G$ instead of $i = j$.

4.3. Plan projections in the case study

Agents naturally commit to their controlling UAV which acts as a 'middle manager' on behalf of the *coordinator*. Each UAV is committed to the *coordinator* with regard to the task of keeping all assigned regions in a $safe$ state.

Each agent has its own projection of the overall plan.

- The *coordinator* is aware of the $\langle Cleanup\rangle$ plan in the context of all regions;
- $UAVs$ need a projection of the $\langle Cleanup\rangle$ plan in all areas to which they are assigned;
- Robots need to have a projection of the $\langle Cleanup\rangle$ plan only regarding actions which they may take part in.

Projections for the pilot are not considered here, since he works independently.

5. TEAMLOG: a logical theory for teamwork

Due to the space limit, only the relevant fragment of TEAMLOG will be presented here. For extensive explanation and discussion see [14–16, 20].

5.1. Beliefs in TEAMLOG

For the individual part, a standard $KD45_n$ system for n agents has been adopted, governing the individual belief operator BEL, as explained in [23]. Additionally, for group beliefs, with a group $G \subseteq \{1, \ldots, n\}$, as in [23]:

C1 $\mathrm{E\text{-}BEL}_G(\varphi) \leftrightarrow \bigwedge_{i \in G} BEL(i, \varphi)$ *(general belief: "everyone believes")*

More general iterated form for $k \geq 2$:

$\mathrm{E\text{-}BEL}_G^k(\varphi) \leftrightarrow \mathrm{E\text{-}BEL}_G^{k-1}(\mathrm{E\text{-}BEL}_G(\varphi))$, where $\mathrm{E\text{-}BEL}_G^1(\varphi) \equiv \mathrm{E\text{-}BEL}_G(\varphi)$

A very strong and heavily used notion in TEAMLOG is that of *common belief*: $\mathrm{C\text{-}BEL}_G(\varphi)$ - "it is common belief in the group G that φ is true":

C2 $\mathrm{C\text{-}BEL}_G(\varphi) \leftrightarrow \mathrm{E\text{-}BEL}_G(\varphi \wedge \mathrm{C\text{-}BEL}_G(\varphi))$
RC1 From $\varphi \rightarrow \mathrm{E\text{-}BEL}_G(\psi \wedge \varphi)$ infer $\varphi \rightarrow \mathrm{C\text{-}BEL}_G(\psi)$ *(induction)*

Common belief is a very powerful notion. For example, if $\mathrm{C\text{-}BEL}_G(\varphi)$ holds then $\mathrm{C\text{-}BEL}_G(\psi)$ holds as well with ψ being any logical consequence of φ. This ensures that agents reach the same conclusions from φ and commonly believe in them.

5.2. Tuning collective attitudes

Collective notions from TEAMLOG ensure the calibration of agents' *awareness*. In this case study the strength of collective attitudes is adjusted to a specific domain, while group attitudes are set at the minimal level ensuring effective team operation. In general, the question regarding the level of awareness about each specific aspect of teamwork needs to be addressed.

Instances of $awareness_G$ in TEAMLOG definitions can be anything from $\emptyset$, through individual beliefs, different levels of $\mathrm{E\text{-}BEL}_G^k$, to common belief $\mathrm{C\text{-}BEL}_G$. Stronger levels of belief require increased communication to ensure their proper propagation. It has been argued that in ideal teamwork, $awareness_G$ is taken to be $\mathrm{C\text{-}BEL}_G$ [14]. Assuming that the communication medium is perfect, it is possible to attain this. In practical implementations in an asynchronous, uncertain medium, common knowledge ($\mathrm{C\text{-}KNOW}_G$) has been proven to be impossible to achieve [29], and common belief ($\mathrm{C\text{-}BEL}_G$) only under extremely restricted constraints [17]. Alternatively, in realistic circumstances, one can apply communication protocols that establish ever higher approximations of common belief within a group [4, 2]. These protocols are less efficient because the required number of messages passed back and forth between the initiator and other agents grows linearly with the desired level of iteration. However, the underlying assumptions are much easier to guarantee.

5.3. Intentions in TeamLog

For the individual intention INT, the TeamLog axioms comprise the system KD_n, including the intention consistency axiom D. In addition, the system developer may choose whether to add negative introspection of intentions (see [14]).

It is certainly not sufficient that all members of the team G have the associated individual intention $\mathrm{INT}(i, \varphi)$ to achieve φ, i.e. a *general intention*. To exclude competition, all agents should *intend* all members to have the associated individual intention, as well as the intention that all members have the individual intention, and so on; this is called a *mutual intention* ($\mathrm{M\text{-}INT}_G(\varphi)$). Furthermore, all team members are aware of this mutual intention: $awareness_G(\mathrm{M\text{-}INT}_G(\varphi))$. Please note that team members remain autonomous in maintaining their other motivational attitudes, and may compete about other issues.

M1 $\mathrm{E\text{-}INT}_G(\varphi) \leftrightarrow \bigwedge_{i \in G} \mathrm{INT}(i, \varphi)$ *(general intention: "everyone intends")*

$\mathrm{E\text{-}INT}^k_G$ is also defined iteratively, similarly to higher-order general beliefs, for $k \geq 2$:

M1′ $\mathrm{E\text{-}INT}^k_G(\varphi) \leftrightarrow \mathrm{E\text{-}INT}^{k-1}_G(\mathrm{E\text{-}INT}_G(\varphi))$, where $\mathrm{E\text{-}INT}^1_G(\varphi) \equiv \mathrm{E\text{-}INT}_G(\varphi)$

Mutual and collective intentions are governed by:

M2 $\mathrm{M\text{-}INT}_G(\varphi) \leftrightarrow \mathrm{E\text{-}INT}_G(\varphi \wedge \mathrm{M\text{-}INT}_G(\varphi))$ *(mutual intention)*
RM1 From $\varphi \to \mathrm{E\text{-}INT}_G(\psi \wedge \varphi)$ infer $\varphi \to \mathrm{M\text{-}INT}_G(\psi)$ *(induction)*
M3 $\mathrm{C\text{-}INT}_G(\varphi) \leftrightarrow \mathrm{M\text{-}INT}_G(\varphi) \wedge awareness_G(\mathrm{M\text{-}INT}_G(\varphi))$ *(collective intention)*

5.4. Collective commitment in TeamLog

After a group is constituted via collective intention, a *collective commitment* between the team members needs to be established. While a collective intention is an inspiration for team activity, the plan-based collective commitment expresses the case-specific details provided by planning and action allocation. It is reflected in bilateral commitments towards individual actions. A bilateral commitment from agent i towards agent j to perform action α is represented as $\mathrm{COMM}(i, j, \alpha)$. In this article, bilateral commitment is viewed as a primitive notion, but see [16] for its characterization and governing axiom.

5.5. Collective commitment schema

A flexible tuning schema for *collective commitments* is presented in [16]. In summary, group G has a *collective commitment* to achieve goal φ based on social plan P ($\mathrm{C\text{-}COMM}_{G,P}(\varphi)$) iff all of the following hold (in the corresponding definition below, parts between curly brackets may or may not be present): The group mutually intends φ (with or without being aware of this); moreover, successful execution of social plan P leads to φ ($cons(\varphi, P)$) (again, with or without the group being aware of this); and finally, for every action α from plan P, there is one agent in the group who is bilaterally committed to another agent in the group to fulfil that action ($\mathrm{COMM}(i, j, \alpha)$) (with or without the group being aware of this):

$$\begin{aligned}&\text{C-COMM}_{G,P}(\varphi) \leftrightarrow \text{M-INT}_G(\varphi) \wedge \{\mathit{awareness}_G(\text{M-INT}_G(\varphi))\} \wedge \\ &\quad \mathit{cons}(\varphi, P) \wedge \{\mathit{awareness}_G(\mathit{cons}(\varphi, P))\} \wedge \\ &\quad \bigwedge_{\alpha\in P} \bigvee_{i,j\in G} \text{COMM}(i,j,\alpha) \wedge \{\mathit{awareness}_G(\bigwedge_{\alpha\in P} \bigvee_{i,j\in G} \text{COMM}(i,j,\alpha))\}\end{aligned}$$

Strong collective commitment Different types of collective commitments related to different organizational structures and environments have been introduced in terms of a 'tuning machine' [16]. One instantiation of the above scheme is the *strong collective commitment* [16]. In this case, the team knows the overall goal, collectively believes that the social plan is correct ($\text{C-BEL}_G(\mathit{cons}(\varphi, P))$) and that things are under control. However, team members do not need to know exactly who is responsible for each task.

Strong collective commitment is applicable in teams with no collective responsibility:

$$\begin{aligned}&\text{S-COMM}_{G,P}(\varphi) \leftrightarrow \text{C-INT}_G(\varphi) \wedge \mathit{cons}(\varphi, P) \wedge \\ &\quad \mathit{awareness}_G(\mathit{cons}(\varphi, P)) \wedge \bigwedge_{\alpha\in P} \bigvee_{i,j\in G} \text{COMM}(i,j,\alpha) \wedge \\ &\quad \mathit{awareness}_G(\bigwedge_{\alpha\in P} \bigvee_{i,j\in G} \text{COMM}(i,j,\alpha))\end{aligned}$$

Weak collective commitment In *weak collective commitment* [16], the team knows the overall goal, but doesn't know the details of the plan: there is no collective awareness of the plan's correctness, even though actions have been appropriately allocated.

Weak collective commitment is applicable in teams with a dedicated planner, who takes care of the proper planning reflected in $\mathit{cons}(\varphi, P)$:

$$\begin{aligned}&\text{W-COMM}_{G,P}(\varphi) \leftrightarrow \text{C-INT}_G(\varphi) \wedge \mathit{cons}(\varphi, P) \wedge \\ &\quad \bigwedge_{\alpha\in P} \bigvee_{i,j\in G} \text{COMM}(i,j,\alpha) \wedge \mathit{awareness}_G(\bigwedge_{\alpha\in P} \bigvee_{i,j\in G} \text{COMM}(i,j,\alpha))\end{aligned}$$

Team commitment A further weakening of the group commitment results in the *team commitment* [16]. This case differs from *weak collective commitment* including solely obligatory elements of collective commitment, without the team being aware of them. Therefore, agents can't infer if any action they take will result in completion of plan P.

Team commitment is applicable in teams where agents only receive orders and don't help each other unless specifically told what to do.

$$\text{T-COMM}_{G,P}(\varphi) \leftrightarrow \text{C-INT}_G(\varphi) \wedge \mathit{cons}(\varphi, P) \wedge \bigwedge_{\alpha\in P} \bigvee_{i,j\in G} \text{COMM}(i,j,\alpha)$$

6. Adjusting the TEAMLOG notions to the case study

6.1. C-COMM_G on the sub-team level

Within a sub-team G_i, the UAV_i has the highest level of awareness as it knows the entire $\langle \mathit{Cleanup} \rangle$ plan for a particular region. There is no need for others to know the details of that plan.

The strength of the *collective commitment* depends on the assumptions regarding cooperation between cleanup robots. There are two possible scenarios:

- Robots help one another only when their UAV orders them to.
  In this case, robots from G_i need a quite limited awareness of the plan. For example, they need to know the partially instantiated subplans applicable in dangerous situations ($\langle D_1R\rangle$, $\langle D_2R\rangle$ or $\langle D_2N\rangle$). In the relevant weather conditions, they may need to carry out one of these subplans for a region assigned by the UAV_i. This UAV_i also assigns roles to robots, who do not know the shares of others. Thus, *team commitment* is sufficient on the sub-team level.
- Robots help each other voluntarily.
  In this case, they will also need to be aware about the partially instantiated plans of nearby robots from G_i in order to assist and assume a role that one of its colleagues fails to perform. With regard to the $\langle Cleanup\rangle$ plan, this corresponds to *weak collective commitment* on the sub-team level. In addition, it requires $\bigwedge_{\alpha\in P}\bigvee_{m,n\in G_i}\text{E-BEL}_{G_i}(\text{COMM}(m,n,\alpha))$. That specific belief is necessary if agents are to pitch in for one another.

6.2. C-COMM$_G$ on the team level

In team G the *coordinator* has the highest awareness; for example, it is in charge of the global planning, as explained in section 2.2. The $UAVs$ in its team only need to know their projection of the overall plan, and believe that the entire plan has been shared among them. The $coordinator$ knows both the plan and action allocation. The *coordinator*, the *pilot* and and all $UAVs$ are aware that $cons(safe, Cleanup)$ holds.

With regard to the $\langle Cleanup\rangle$ plan, this corresponds to *strong collective commitment* on the top team level, where all $UAVs$ and the pilot make social commitments to the coordinator. Taking $G' = \{coordinator, pilot, UAV_1, \ldots, UAV_k\}$ and putting the $awareness_{G'}$ dial at common belief, we have:

$$\begin{aligned}
&\text{S-COMM}_{G',Cleanup}(safe) \leftrightarrow \text{C-INT}_{G'}(safe) \wedge cons(safe, Cleanup) \wedge \\
&\quad \text{C-BEL}_{G'}(cons(safe, Cleanup)) \wedge \\
&\quad \textstyle\bigwedge_{\alpha\in Cleanup}\bigvee_{i\in G'}\text{COMM}(i, coordinator, \alpha) \wedge \\
&\quad \text{C-BEL}_{G'}(\textstyle\bigwedge_{\alpha\in Cleanup}\bigvee_{i\in G'}\text{COMM}(i, coordinator, \alpha))
\end{aligned}$$

The dialogues taking place between the coordinator and its direct subordinates during the creation of collective commitments ensure that the necessary awareness is established (see [20, Chapter 8]).

6.3. Organization structure: who is socially committed to whom?

Commitments in the team follow the organizational structure of Figure 2. The coordinator is socially committed to achieving the overall goal, with respect to the main social plan. Other agents are committed to their projection of that plan. The coordinator is committed towards itself and towards the relevant control authority, for example, the national environmental agency for which it works.

Each UAV_i for $i = 1, \ldots, k$ is committed towards the coordinator with respect to achieving its part of the plan, namely keeping specified regions safe.

The robots in G_i for $i = 1, \ldots, k$ commit to perform their share to their leading UAV_i, which has the power to uncommit them. There is a clear hierarchy where the coordinator is the leader of the team G as a whole, while the $UAVs$ are 'middle-managers' of sub-teams. The $UAVs$ also sometimes commit to a colleague UAV when some of their robots are temporarily delegated to the other's sub-team.

The human pilot has a somewhat special role in that he does not manage any sub-team. Instead, he directly commits to the coordinator, or sometimes to $UAVs$ if they request his assistance.

6.4. Minimal levels of group intention and awareness

What are the minimal levels of awareness and group intention needed for the agents on both sub-team and team levels?

The robots - two cases are applicable

1. They act only individually; this is the most limited (and economical) case;
2. They perform a limited form of cooperation, for example, they work together to clean up areas faster (e.g., by suitably combining the application of cleaning solids and catalysts), or pitch in for other robots when these turn out to be unable to perform their share of labor.

Both cases will be considered separately while investigating group attitudes of different types of agents involved in achieving a maintenance goal to keep the region safe.

The level of intention

1. In case 1, the robots need a general intention E-INT_{G_i} about the goals.
2. In case 2, $\text{E-INT}^2_{G_i}$ will be enough to allow forming two-robot teams that are not competitive internally. (But see [14] for a counter-example showing that a two-level intention is not sufficient to preclude competition among two-agent coalitions). If agents are supposed to be strictly cooperative, a two-level definition is also sufficient for larger teams: all agents intend to achieve the goal in cooperation with the others included in their team.

Although robots sometimes individually compete for resources, in our field of application where fast real-time team reaction to dangers is needed, we opt for strictly cooperative robots that use fixed protocols to load up on resources. The cleanup robots do not communicate with robots from other teams, and therefore do not need to have any beliefs, intentions and commitments regarding them.

The level of belief

1. In case 1, to act individually each robot i needs an individual belief about every group intention $(\mathrm{BEL}(i, \text{E-INT}_G(\varphi)))$. This way, a general belief $\text{E-BEL}_G(\text{E-INT}_G(\varphi))$ is in place and it suffices. Moreover, each robot in G_i should believe that the distribution of labor by means of bilateral commitments is done properly $(\text{E-BEL}_{G_i}(\bigwedge_{\alpha \in P} \bigvee_{n,m \in G_i} \mathrm{COMM}(n, m, \alpha)))$. This allows deliberation on actions of other robots from the same team. It may also prevent robots from doing all the work by themselves.
2. In case 2, $\text{E-BEL}^2_{G_i}$ will be enough to allow deliberation about other robots' intentions and beliefs (especially $\text{E-BEL}^2_{G_i}(\text{E-INT}^2_{G_i}(\varphi)))$. To see this, one may consider a pair of robots. With $\text{E-BEL}^2_{G_i}$, both robots have the same intention $(\text{E-INT}_{G_i}(\varphi))$, believe they have the same intention (the first-order belief $\text{E-BEL}_{G_i}(\text{E-INT}_{G_i}(\varphi)))$, and believe that the other believes this (the second-order belief $\text{E-BEL}_{G_i}(\text{E-BEL}_{G_i}(\text{E-INT}_{G_i}(\varphi)))$. Therefore, the robots can reason about the beliefs and intentions of their partner.

In both cases, it is assumed that the robots are incapable of forming coalitions of cardinality ≥ 2. This allows us to build a sufficiently strong collective intention based on a low degree of nesting of general intentions [13]. In case 2, the robots will also need to be aware of plan projections of their neighbours, in order to be able to notice when they can help.

The $\boldsymbol{UAVs}$ The $UAVs$ must sometimes work with one another. This requires at least E-BEL^2_G of other $UAVs$' intentions.

The level of intention - sub-team level Within each sub-team G_i consisting of an UAV and robots, UAV_i must make sure that all agents are motivated to do their tasks. Therefore:

- in case 1, $\mathrm{INT}(UAV_i, \text{E-INT}_{G_i}(\varphi))$ is required with respect to the sub-team intention $\text{E-INT}_{G_i}(\varphi)$,
- in case 2, $\mathrm{INT}(UAV_i, \text{E-INT}^2_{G_i}(\varphi))$ is required with respect to the sub-team intention $\text{E-INT}^2_{G_i}(\varphi)$. The UAV_i's intention is that all the robots in its team not only do their work, but also have the intention of helping each other in two-robot teams.

The level of belief - sub-team level Within each sub-team G_i consisting of UAV_i and robots $rob_{i_1}, \ldots, rob_{i_{n_i}}$, the UAV_i has the highest level of awareness, and acts as a coordinator. In order to facilitate this (make plans and reason correctly), it will require one level of belief more than its agents:

- in case 1, $\mathrm{BEL}(UAV_i, \text{E-BEL}_{G_i}(\text{E-INT}_{G_i}(\varphi)))$ is required with respect to the sub-team intention $\text{E-INT}_{G_i}(\varphi)$ as well as:
  $\mathrm{BEL}(UAV_i, \text{E-BEL}_{G_i}(\bigwedge_{\alpha \in Cleanup} \bigvee_{i,j \in G_i} \mathrm{COMM}(i, j, \alpha)))$,

– in case 2 $\mathrm{BEL}(UAV_i, \text{E-BEL}^2_{G_i}(\text{E-INT}^2_{G_i}(\varphi)))$ is required with respect to the sub-team intention $\text{E-INT}^2_{G_i}(\varphi)$ as well as:
$\mathrm{BEL}(UAV_i, \text{E-BEL}^2_{G_i}(\bigwedge_{\alpha \in Cleanup} \bigvee_{i,j \in G_i} \mathrm{COMM}(i, j, \alpha)))$.

The level of intention - team level There are situations when two $UAVs$ create coalitions of cardinality ≤ 2. In order not to be competitive internally, all $UAVs$ must have at least $\text{E-INT}^2_G(\varphi)$ with respect to every goal φ.

The level of belief - team level Cooperation between $UAVs$ requires at least E-BEL^2_G of other $UAVs$' intentions.

The coordinator The role of coordinator is to manage the team as a whole (see Figure 2), including all sub-teams and the pilot. Therefore he needs to know not only the global plan but also all the subplans.

The level of intention Similarly as in the relation of a UAV viz-a-viz its robots, the coordinator has one level of intention more than the $UAVs$ it manages, therefore $\mathrm{INT}(coordinator, \text{E-INT}^2_G(\varphi))$ is required.

The level of belief One extra level of belief allows the coordinator introspection and reasoning about the joint effort of all $UAVs$. Therefore, since teams are co-operative in a limited way, $\mathrm{BEL}(coordinator, \text{E-BEL}^2_G(\text{E-INT}^2_G(\varphi)))$ is required with respect to every group intention $\text{E-INT}^2_G(\varphi)$. Again, an analogical level of awareness is required with regard to distribution of bilateral commitments:
$\mathrm{BEL}(coordinator, \text{E-BEL}^2_G(\bigwedge_{\alpha \in Cleanup} \bigvee_{i,j \in G} \mathrm{COMM}(i, j, \alpha)))$.

Commands from the coordinator overrule temporary contracts between sub-teams. The coordinator does not only know the plan, but also keeps track of all relevant environmental conditions. It is assumed that even in the safe situation, the robots, the $UAVs$ and the pilot are prepared to take action at any moment.

The pilot The pilot's own awareness about the team level is similar to the $UAVs$', except that he usually does not need precise awareness about the robots' actions. Additionally, as a human team member he has a special position and we do not want to impose on him a straight-jacket of awareness from others. The only exception is that the $coordinator$ is aware of the pilot's share of labor and his commitments and if applicable, some $UAVs$ are aware of actions the pilot commits to perform for them.

6.5. Complexity of the language without collective attitudes

It seems that in the environmental case study, the language used is richer than propositional modal logic due to the use of continuous ranges, functions, and a theoretically unlimited number of agents and teams. Fortunately, most of the relevant part can be reduced to a fixed finite number of propositional atoms (that may be combined and be the subject of attitudes), based on finitely many predicates and constants, as follows:

– a fixed number of relevant environmental states;
– a fixed number of pre-named locations;
– a fixed finite number of agents and teams;
– a fixed finite number of other objects (liquids, solids, catalyst, helicopter);
– a fixed number of relevant thresholds $n_1, n_2, n_3, \epsilon_1, \epsilon_2$.

The only possible source of unbounded complexity is the use of continuous intervals and real-valued functions $f_1, f_2, f_3, \mathit{fit}$ appearing in Section 2. Recall that the architecture proposed in Section 1.2 allows us to query external entities. These concern data stored in databases and sensed from the environment, which are represented in the lower layer of the system. For example, the functions f_1, f_2, f_3 and *fit* are part of this lower layer. Therefore, even though the underlying structures are represented by first-order formulas, one extracts only propositional information from them to use in the upper layer of propositional TEAMLOG reasoning. In fact, one can obtain answers *true* or *false* about queries such as

$$f_3(p(A), t(A), c_1(A), c_2(A)) \in (v_3, n_3]?$$

from the lower layer, without needing to resort to a first-order language in the upper layer of the system.

7. Discussion of related approaches

One of the most influential theories of teamwork is the one of Wooldridge and Jennings [42]. The actual formal frameworks of their papers is quite different from ours. Wooldridge and Jennings define joint commitment towards φ in a more dynamic way than collective intentions defined in TEAMLOG: according to [42], initially the agents do not believe φ, and subsequently have φ as a goal until the termination condition is satisfied, including (as conventions) conditions on the agents to turn their eventual beliefs that termination is warranted into common beliefs. Subsequently, they define having a joint intention to do α as "having a joint commitment that α will happen next, and then α happens next". In contrast, agreeing with [5], we view collective commitments as stronger than collective intentions, and base the collective commitment on a specific social plan meant to realize the collective intention.

The emphasis on establishing appropriate collective attitudes for teamwork is shared with the SharedPlans approach of Grosz and Kraus [28, 27]. Nevertheless, the intentional component in their definition of collective plans is weaker than our collective intention: Grosz and Kraus' agents involved in a collective plan have individual intentions towards the overall goal and a common belief about these intentions; intentions with respect to the other agents play a part only at the level of individual sub-actions of the collective plan. We stress, however, that team members' intentions about their colleagues' motivation to achieve the overall goal play an important role in keeping the team on track even if their plan has to be changed radically due to a changing environment.

Similarly to [28, 27], Rao, Georgeff and Sonenberg [36] use a much weaker definition of joint intention than our collective intention: theirs is only one-level, being defined as "everyone has the individual intention, and there is a common belief about this". Thus, their definition does not preclude cases of coercion and competition. We have shown in the case study that, even though the full infinitary flavor of collective intention is not always needed, even among truly cooperative agents effective teamwork in disaster response demands at least a second level of general intentions (see section 6.4).

Another theory of teamwork, by Levesque, Cohen and Nunes [31], does incorporate higher-level intentions in its joint intention. Also, common belief is an integral part of a group's intention in both [31, 36], which may be problematic because creating common beliefs is costly in terms of communication and often impossible if the communication medium is untrustworthy [17, 4, 2]. Problematic communication was one of the reasons for restricting to minimal levels of belief in section 6.4. Additionally, in contrast to TEAMLOG, Levesque and colleagues assume a homogeneous group and the absence of social structure. Accounting for a group's social structure is crucial in a theory of teamwork, and in fact it is employed in our investigation of collective commitments in the case study.

Some approaches to collective commitments introduce other aspects, not treated here. For example, Aldewereld et al. add constraints about goal adoption and achievement to their definitions of joint motivational attitudes [1]. We have chosen to incorporate solely vital aspects of the defined attitudes, leaving room for any case-specific extensions. If needed, these extensions may be added by extra axioms. Note that in contrast to approaches such as [42, 31], our collective commitment is not iron-clad: it may vary in order to adapt to changing circumstances, in such a way that the collective intention on which it is based can still be reached.

In the logical MAS literature some phenomena such as the dynamics of attitude revision during reconfiguration have received scant attention (but see [38, 39]). Our notion of collective commitment ensures efficiency of reconfiguration in two ways. Unlike in [42], our approach to group commitments is formalized in a non-recursive way. This allows for a straightforward revision. Next, because only social commitments to individual actions appear, it often suffices to revise just some of them. That way it is possible to avoid involving the whole team in replanning. Thus, teamwork axioms may serve a system designer as a high-level specification at design-time. During run-time, formal verification methods may be applied to check the correctness of the system behavior.

In conclusion, our logical theory TEAMLOG has been developed against the background of Bratman's four criteria for shared cooperative activity [3]:

– mutual responsiveness;
– commitment to the joint activity;
– commitment to mutual support;
– formation of subplans that mesh with one another.

Kraus and colleagues [25] apply an illuminating comparison in the light of Bratman's criteria to six approaches to teamwork, including an early version of TEAMLOG [14] and some of those theories mentioned above [42, 28, 27, 31, 36] as well as [26]. They base their analysis on a very simple example of cooperation, where two agents move an object together. It would be interesting to compare all six theories on a complex time-critical example such as the present disaster management case study, but this would require a book-length study and is out of the scope of this paper.

8. Conclusion

In the case study we have shown how to implement teamwork within a strictly cooperative, but still heterogenous group of agents in TEAMLOG. The heterogeneity is taken seriously here, as advocated in [24]. Natural differences in agents' shares, opportunities and capabilities when acting together, have been reflected in different levels of agents' awareness about various aspects of their behaviour. The study dealt especially with cooperation and coordination. Having very generic definitions of common motivational and informational attitudes in TEAMLOG, it is challenging to choose the proper level of their complexity. We have shown that this is possible, by illustrating how to tailor complex definitions of intentions and commitments to a specific environment. Our focus was on building beliefs, intentions and, finally, commitments of all agents involved in teamwork on an adequate, but still minimal level. Therefore, even though not all aspects of teamwork have been shown, a bridge between theory and practice of teamwork has been constructed for this exemplary application, under the assumptions of fully cooperative agents and a hierarchical organization. It would be interesting to investigate the influence of relaxing both assumptions as well as to incorporate the dynamics of team dialogue and reconfiguration in the case study, according to the dynamic theory TEAMLOG^{dyn} [20, Chapters 5,6,8].

Future work will be to embed TEAMLOG into a form of approximate reasoning suitable for modeling perception. *Similarity-based approximate reasoning*, with its intuitive semantics compatible with that of TEAMLOG, is a promising candidate [9, 12, 10, 11].

References

1. Aldewereld, H., van der Hoek, W., Meyer, J.: Rational teams: Logical aspects of multi-agent systems. Fundamenta Informaticae 63 (2004)
2. van Baars, E., Verbrugge, R.: A communication algorithm for teamwork in multi-agent environments. Journal of Applied Non-Classical Logics 19, 431–461 (2009)
3. Bratman, M.: Shared cooperative activity. The Philosophical Review 101, 327–341 (1992)
4. Brzezinski, J., Dunin-Kęplicz, P., Dunin-Kęplicz, B.: Collectively cognitive agents in cooperative teams. In: Gleizes, M.P., Omicini, A., Zambonelli, F. (eds.) Engineering Societies in the Agents World V, (ESAW 2004): Revised Selected and Invited Papers. LNCS, vol. 3451, pp. 191–208. Springer, Berlin (2005)

5. Castelfranchi, C.: Commitments: From individual intentions to groups and organizations. In: Lesser, V. (ed.) Proc. First Int. Conference on Multi-Agent Systems. pp. 41–48. AAAI-Press and MIT Press, San Francisco (1995)
6. Cuny, F.: Disasters and Development. Oxford University Press, Oxford (1983)
7. Doherty, P., Granlund, G., Kuchcinski, K., Nordberg, K., Sandewall, E., Skarman, E., Wiklund, J.: The WITAS unmanned aerial vehicle project. In: Proc. of the 14th European Conference on Artificial Intelligence. pp. 747–755 (2000)
8. Doherty, P., Łukaszewicz, W., Skowron, A., Szałas, A.: Knowledge Representation Techniques. A Rough Set Approach, Studies in Fuzziness and Soft Computing, vol. 202. Springer Verlag (2006)
9. Doherty, P., Dunin-Kęplicz, B., Szałas, A.: Dynamics of approximate information fusion. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP. Lecture Notes in Computer Science, vol. 4585, pp. 668–677. Springer (2007)
10. Dunin-Kęplicz, B., Nguyen, L., Szałas, A.: Fusing approximate knowledge from distributed sources. In: Papadopoulos, G., Badica, C. (eds.) Proceedings of the Third International Symposium on Intelligent Distributed Computing. Studies in Computational Intelligence, vol. 237, pp. 75–86 (2009)
11. Dunin-Kęplicz, B., Szałas, A.: Agents in approximate environments. In: van Eijck, J., Verbrugge, R. (eds.) Games, Actions and Social Software. College Publications, London (2010), to appear
12. Dunin-Kęplicz, B., Szałas, A.: Towards approximate BGI systems. In: H.-D.Burkhard, Lindemann, G., Verbrugge, R., Z.Varga, L. (eds.) CEEMAS. Lecture Notes in Computer Science, vol. 4696, pp. 277–287. Springer (2007)
13. Dunin-Kęplicz, B., Verbrugge, R.: Collective commitments. In: Tokoro [40], pp. 56–63
14. Dunin-Kęplicz, B., Verbrugge, R.: Collective intentions. Fundamenta Informaticae 51(3), 271–295 (2002)
15. Dunin-Kęplicz, B., Verbrugge, R.: Evolution of collective commitments during teamwork. Fundamenta Informaticae 56, 329–371 (2003)
16. Dunin-Kęplicz, B., Verbrugge, R.: A tuning machine for cooperative problem solving. Fundamenta Informaticae 63, 283–307 (2004)
17. Dunin-Kęplicz, B., Verbrugge, R.: Creating common beliefs in rescue situations. In: Dunin-Kęplicz, B., Jankowski, A., Skowron, A., Szczuka, M. (eds.) Proc. of Monitoring, Security and Rescue Techniques in Multiagent Systems (MSRAS). pp. 69–84. Advances in Soft Computing, Springer, Berlin (2005)
18. Dunin-Kęplicz, B., Verbrugge, R.: Awareness as a vital ingredient of teamwork. In: Stone, P., Weiss, G. (eds.) Proc. of the Fifth Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06). pp. 1017–1024. IEEE/ACM Press, New York (NY) (2006)
19. Dunin-Keplicz, B., Verbrugge, R.: Dynamics of collective attitudes during teamwork. In: ESAW. pp. 107–122 (2003)
20. Dunin-Keplicz, B., Verbrugge, R.: Teamwork in Multi-Agent Systems: A Formal Approach. John Wiley & Sons, Ltd. (2010)
21. Dziubiński, M.: Complexity of the logic for multiagent systems with restricted modal context. In: Dunin-Kęplicz, B., Verbrugge, R. (eds.) Proc. of the Third Int. Workshop on Formal Approaches to Multi-Agent Systems, FAMAS'007. pp. 1–18. Durham University (2007)
22. Dziubiński, M., Verbrugge, R., Dunin-Kęplicz, B.: Complexity issues in multiagent logics. Fundamenta Informaticae 75(1-4), 239–262 (2007)
23. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press, Cambridge, MA (1995)

24. Gold, N. (ed.): Teamwork. Palgrave McMillan, Basingstoke and New York (2005)
25. Grant, J., Kraus, S., Perlis, D.: Formal approaches to teamwork. In: Artemov, S., others (eds.) We Will Show Them: Essays in Honour of Dov Gabbay, vol. 1, pp. 39–68. College Publications, London (2005)
26. Grant, J., Kraus, S., Perlis, D.: A logic-based model of intention formation and action for multi-agent subcontracting. Artificial Intelligence 163(2), 163–201 (2005)
27. Grosz, B., Kraus, S.: The evolution of SharedPlans. In: Rao, A., Wooldridge, M. (eds.) Foundations of Rational Agency, pp. 227–262. Kluwer, Dordrecht (1999)
28. Grosz, B., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence 86(2), 269–357 (1996)
29. Halpern, J., Moses, Y.: Knowledge and common knowledge in a distributed environment. Journal of the ACM 37, 549–587 (1990)
30. Kleiner, A., Prediger, J., Nebel, B.: RFID technology-based exploration and SLAM for search and rescue. In: Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2006). pp. 4054–4059. Bejing (2006)
31. Levesque, H., Cohen, P., Nunes, J.: On acting together. In: Proc. Eighth National Conference on AI (AAAI90). pp. 94–99. MIT Press, Cambridge (MA) (1990)
32. Meyer, J., van der Hoek, W.: Epistemic Logic for AI and Theoretical Computer Science. Cambridge University Press, Cambridge (1995)
33. Parikh, R., Krasucki, P.: Levels of knowledge in distributed computing. Sadhana: Proc. of the Indian Academy of Sciences 17, 167–191 (1992)
34. Pynadath, D.V., Tambe, M.: The communicative multiagent team decision problem: Analyzing teamwork theories and models. Journal of Artificial Intelligence Research 16, 389–423 (2002)
35. Rao, A., Georgeff, M.: Modeling rational agents within a BDI-architecture. In: Fikes, R., Sandewall, E. (eds.) Proc. of the Second Conference on Knowledge Representation and Reasoning. pp. 473–484. Morgan Kaufman (1991)
36. Rao, A., Georgeff, M., Sonenberg, E.: Social plans: A preliminary report. In: Werner, E., Demazeau, Y. (eds.) Decentralized A.I.-3. pp. 57–76. Elsevier, Amsterdam (1992)
37. Sycara, K., Lewis, M.: Integrating intelligent agents into human teams. In: Salas, E., Fiore, S. (eds.) Team Cognition: Understanding the Factors that Drive Process and Performance. pp. 203–232. American Psychological Association, Washington (DC) (2004)
38. Tambe, M.: Teamwork in real-world, dynamic environments. In: Tokoro [40], pp. 361–368
39. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research 7, 83–124 (1997)
40. Tokoro, M. (ed.): Proc. Second Int. Conference on Multi-Agent Systems. AAAI-Press, Menlo Park (CA) (1996)
41. Wisner, B., Blaikie, P., Cannon, T., Davis, I.: At Risk - Natural Hazards, People's Vulnerability and Disasters. Routledge, Wiltshire (2004)
42. Wooldridge, M., Jennings, N.: Cooperative problem solving. Journal of Logic and Computation 9, 563–592 (1999)

Barbara Dunin-Kęplicz s a Professor of computer science at the Institute of Informatics of Warsaw University and at the Institute of Computer Science of the Polish Academy of Sciences. She obtained her PhD in 1990 on computational linguistics from the Jagiellonian University, and in 2004 she was awarded her habilitation on formal methods in multi-agent systems from the Polish Academy of Sciences.

She is a recognized expert in multi-agent systems. She was one of the pioneers of modeling BDI systems, recently introducing approximate reasoning to the agent-based approach.

Rineke Verbrugge is a Professor of logic and cognition at the Institute of Artificial Intelligence of the University of Groningen. She obtained her PhD in 1993 on the logical foundations of arithmetic from the University of Amsterdam, but shortly thereafter moved to the research area of multi-agent systems.

She is a recognized expert in multi-agent systems and one of the leading bridge-builders between logic and cognitive science.

Michał Ślizak is a PhD student at the Institute of Informatics of Warsaw University, studying the field of multi-agent systems and approximate reasoning under supervision of Barbara Dunin-Keplicz.

Received: February 9, 2010; Accepted: May 3, 2010.

UDC 004.75, DOI:10.2298/CSIS100118011J

A Case Study on Availability of Sensor Data in Agent Cooperation

Christian Johansson, Fredrik Wernstedt, Paul Davidsson

School of Computing, Blekinge Institute of Technology
PO Box 520, SE-372 25, Ronneby, Sweden
christian.johansson@bth.se
fredrik.wernstedt@bth.se
paul.davidsson@bth.se

Abstract. Multi-agent cooperation can in several cases be used in order to mitigate problems relating to task sharing within physical processes. In this paper we apply agent based solutions to a class of problems defined by their property of being predictable from a macroscopic perspective while being highly stochastic when viewed at a microscopic level. These characteristic properties can be found in several industrial processes and applications, e.g. within the energy market where the production and distribution of electricity follow this pattern. Another defining problem characteristic is that the supply is usually limited as well as consisting of several layers of differentiating production costs. We evaluate and compare the performance of the agent system in three different scenarios, and for each such scenario it is shown to what degree the optimization system is dependent on the level of availability of sensor data.

Keywords: agent co-operation, team work

1. Introduction

Schemes for sustaining cooperative behavior among agents are often dependent on a certain level of communication in order to establish and maintain a reciprocal sense of trust. However, in real-life applications it is not always possible to uphold the desired level of availability and quality of data being communicated among the agents, thus causing suboptimal cooperative behavior. In this paper we focus on a problem domain where multi-agent task sharing cooperative behavior is applied. However, as practical implementations within this domain often are spread geographically over a wide area and lack dedicated network communication infrastructure, there are often practical limitations on the availability and quality of sensor data which in turn limits the effectiveness of the multi-agent system cooperative behavior.

For agents to effectively coordinate their actions, the agents normally need to share information. Information sharing, i.e. communication and its effect on overall performance is a well established area and has been studied by several researchers [6], [7] and [13]. Also,the area of multi-sensor networks and sensor data quality and fusion has received a fair amount of interest [4], [10] and [8]. However, the quality of information in combination with information sharing has so far, to our knowledge, only received little attention.

1.1. Problem Domain

The problem domain is characterised by being predictable from a macroscopic perspective while being stochastic when viewed at a microscopic level. As the macroscopic behaviour is a reflection of a collection of highly stochastic microscopic events which in themselves cannot be predicted, it follows that although a process control system is able to foresee general trends and tendencies within the process, it must be able to handle the stochastic behaviour in order to actually manipulate the process. For example, although it is possible to foresee the overall heating demand within a building being higher tomorrow as the weather prognosis shows a drop in outdoor temperature, it is not possible to predict when individual residents will take a shower, thus creating peaks in the total energy demand when combining usage of hot tap water and space heating. Basically these processes are driven by one or more producers supplying some kind of utility and one or more consumers acting to satisfy their own demand of the utility.

When optimizing the operational production one tries to determine the financially and operationally most efficient way to combine the production resources, while satisfying the consumer needs. This problem is often formalized by using the Economic Dispatch Problem (EDP) and the Unit Commitment Problem (UCP) [5]. By solving the EDP we find out how much load to generate by each of the different available production units at a given time. Since most production units in real life settings cannot be turned on and off at the blink of an eye, it is important to plan ahead of time and determine which units need to be started, when they need to be started and how long they should be committed to being in use, i.e. solving the UCP. These problems are related to each other and are solved using similar optimization techniques. A complicating factor when optimizing production is that the production costs usually display non-linear patterns, due to physical processes like valve effects and the usage of differently priced fuels in production. This leads to objective functions with discontinuous and non-differentiable points, which means that it is generally more appropriate to treat the cost function as a set of piecewise quadratic functions [9], [3]. As demand rises the producing entity is forced to engage increasingly costly production units, and eventually the production costs exceed the possible sale price of the utility. The only way for the producer to mitigate such a situation is to manipulate consumption in order to lower the demand.

By solving the UCP and EDP the producer finds an optimal production strategy for a given time frame, e.g. the next twenty-four hours. This means that the producer wants the consumption to be as close to this strategy as possible; if the consumption falls to low it will result in unnecessarily low income, while a too high consumption will necessitate starting costly peak-load production units. In order to achieve and uphold the desired production strategy multi agent systems and other distributed systems can be used to manage the consumption. We evaluate the success of such systems by measuring their ability to stick to the production strategy in question, while at the same time satisfying consumer demand.

Typically the consumer entity has a wanted state which it tries to uphold at all times. This wanted state is dependent on the physical environment in which the system is functioning, e.g. maintaining comfortable indoor climate in a district heating system. Often, however, a consumer agent can accept smaller deviations from this wanted state during shorter periods of time. This is what makes it possible for a control system to manage the society of consumers in order to achieve some local or global goals. By measuring the deviation from the wanted state it is possible to evaluate the impact of change in the overall system caused by individual consumers.

1.2. Problem Description

The consumption, and thus the production, follows certain patterns which are predictable to some extent from a system wide perspective. These patterns are generated by a composition of highly stochastic microscopic behaviour among consumer entities, which, as long as their demand is fulfilled, are oblivious to their surroundings or any other part of the larger system. By reacting on these individual microscopic events and controlling and limiting the effect of them, the overall system can achieve several benefits for both the consumers and the suppliers of the utility. Trying to control the consumption in such a way is generally called Demand Side Management (DSM), and can in many cases be achieved by using agent technology or other distributed control schemes [14], [2] and [11]. The reason agent technology is useful in DSM, is that there is no need for any centralized entity supervising the Quality of Service (QoS) among the consumers as each consumer is assigned an agent responsible for this task. Each agent will participate in achieving the overall goal set by the DSM strategy, only as long as sufficient QoS can be upheld. This makes the system highly scalable and easy to maintain.

In theory this a school book example for an agent system to solve. The problem is that the agent based solutions proposed for solving DSM in such environments are dependent on the availability of high-quality sensor data, which in practice can be hard to achieve due to limitations in underlying hardware and communication solutions. That an agent system will perform at its best in a domain were high quality sensor data and communication solutions are readily available is not in question, and it is not the intent of this paper to compare different agent based resource allocation schemes within such a high quality domain. The point of this paper is instead to develop an understanding of how different levels of availability of sensor data influence the behaviour of the agent system in a practical setting. Normally there are practical limitations on the sensor data infrastructure and communication set-up which leads to situations far from any high quality scenario. Investing in modern sensor data and communication solutions can be expensive and there is an apparent need to quantify the performance benefits within different scenarios. In Figure 1 this is visualized.

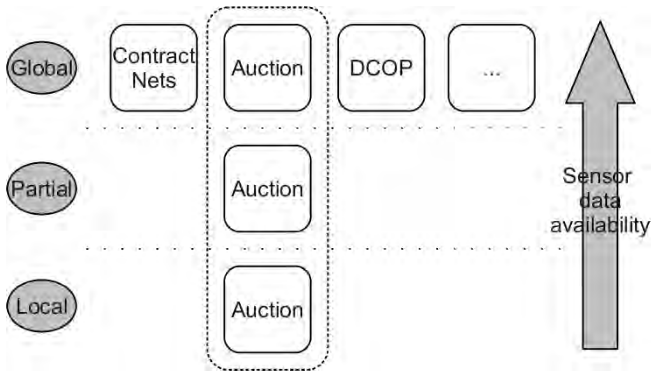


Fig. 1. Comparing availability of sensor data using three different scenarios

Within this study three different scenarios are used to represent different levels of availability of sensor data, i.e global, partial and local. The global level corresponds to a scenario with full access to high quality sensor data while the partial and local scenarios display various levels of deteriorating access to sensor data. There are several ways to coordinate resource allocation within a multi agent system, e.g. contract nets, different auction processes or distributed optimization models. In this study we have used an auction process in order to compare the different scenarios according to Figure 1.

2. The Agent System

The agent system we study in this paper is used to implement DSM strategies within district heating systems and its function has been described in previous work [14]. In district heating systems one or several production plants heat water which is then pumped through a pipe network throughout a city in order to heat buildings and tap water. Every building has a substation with heat exchangers which transfer the heat energy from the primary pipe network into the buildings secondary piping system. The agent system is based on distributed cooperative entities with an overall goal of combining the production and consumption in an optimal manner.

2.1. Agents

Every producer and consumer entity in the system is represented by an agent. A producer agent will try to minimize its own supply cost function while supplying enough utility to satisfy consumer demand. When a producer agent deems it necessary to implement an DSM action it will try to do so by sharing the task among the consumer agents in order to minimize the side effects of DSM on any individual consumer agent. A consumer agent will seek to implement these requests as long as its internal comfort constraints allow for this.

Producer Agent The producer agent is responsible for supervising the continuous utility consumption and also for instigating and distributing DSM tasks when the measured consumption deviates from the desired DSM level. The task sharing is done by first decomposing the initial task into smaller tasks. This is done since the optimization action as a whole is usually too large for one single consumer agent to handle. The tasks are then allocated through a series of auctions. The DSM level is found beforehand by solving the optimization problem relating to the production units, and this is then used as input to the production agent. In larger agent systems it is possible to use cluster agents which act as mediators between a producer agent and a group of consumer agents. This eases the computational load in the producer agent when handling large scale auctions.

The producer agent needs to know the wanted consumption level in order to implement DSM. This is found by solving the EDP and the UCP. These solutions are then used as decision basis for the DSM strategy for the following time frame, normally the next twenty-four hour period. In order to solve the EDP the agent uses an objective function which is found in the smooth function described in Equations 1 and 2.

$$\text{Minimize} \sum_{i \in I} F_i(P_i) \tag{1}$$

$$F_i(P_i) = \alpha_i + \beta_i P_i + \gamma P_i \tag{2}$$

This is simply a summation of the utility cost in all supply units [1]. The value of α describes a fixed cost for starting and running the production unit, while the values of β and γ describe costs dependant on the level of production. The accompanying equality constraint is the utility balance which should be satisfied accordingly:

$$\sum_{i \in I} P_i = D + P_{loss} \tag{3}$$

where D represent the utility demand and P_{loss} indicates any production and distribution losses. The inequality constraints describes the production units working within their respective limits:

$$P_i, min \leq P_i \leq, max \qquad \forall i \in I \tag{4}$$

In practical settings these functions are normally not sufficient to describe many situations in utility production. Normally the production entity will have to treat the cost function as a set of piecewise quadratic functions which are defined as [9], [3]:

$$F_i(P_i) = \begin{cases} \alpha_{i1} + \beta_{i1}P_i + \gamma_{i1}P_i & if P_i^{min} \leq P_i \leq P_{i1} \\ \alpha_{i2} + \beta_{i2}P_i + \gamma_{i2}P_i & if P_{i2} \leq P_i \leq P_{i2} \\ . & . \\ . & . \\ . & . \\ \alpha_{im} + \beta_{im}P_i + \gamma_{im}P_i & if P_{im} - 1 \leq P_i \leq P_i^{max} \end{cases} \tag{5}$$

This behaviour is due to the fact that a utility provider usually has a range of different production units, using differently priced fuels. From a economical point of view there is no smooth transition when switching between the different fuels, which makes the resulting function non-differentiable.

The producer also has to solve the UCP. The UCP is interconnected with the EDP and uses similar optimization methods. The UCP is used to determine which production units to commit to usage and which ones not to use at any given time. In a real world scenario a production unit cannot be turned on and off with a simple switch. It takes a substantial amount of time to start and stop such units, and the cost related to these processes should be kept at a minimum.

By solving the above systems for each relevant point in time it is possible to identify a wanted system wide consumption level within the studied time frame.

Consumer Agent Each consumer unit is controlled by a consumer agent which is responsible for contributing to achieving the overall DSM strategies while maintaining a sufficient level of local comfort. The consumer agents act locally in order to monitor any deviations from the wanted comfort state. The amount of deviation from the optimal comfort state is used as currency when a consumer agent participates in an auction process, i.e the more the consumer agent is straying from its desired comfort state, the less likely it will be to win any auction. The consumer agents are cooperative in the sense that they do not lie about their cost for participating in a DSM task, since this could possibly jeopardize their internal comfort levels. A positive side effect from using the comfort state as currency, is that these calculations are made by the consumer agent in any case and thus the computational effort for valuation and information gathering in regards to the auctioning is kept at a minimum.

2.2. Agent Goal

For every consumer agent there is at any time a wanted comfort level which is dependent on the level of local consumption. Since the physical nature of the process introduces a delay in the dependency between the comfort level and the local consumption level a time frame is created within which it is possible to manipulate the local consumption while still keeping the local comfort level within its constraints. An example of this phenomena is that it will take some time before people notice if you shut off the radiators in a building, i.e. there is a delay before the people start to freeze even though the energy consumption is reduced directly. Combining the local consumption from each consumer agent will yield the total actual consumption. The goal for the agent system is then; for each point in time to achieve a total actual consumption as close as possible to the total wanted consumption while keeping all local comfort levels within their individual constraints.

In a steady state system this could be seen as a traditional optimization problem, i.e. to find a optimum between two conflicting objective functions. However, since we are dealing with a dynamic system the aspects of adaptation and re-planning becomes important, which requires a more sophisticated solution.

2.3. Auction Process

Whenever a producer agent needs to implement a DSM action it will distribute this by using a private value first priced, sealed bid auction process. For the consumer agent the value is to implement as much DSM tasks as possible, and the currency used is the amount of deviation from the optimal comfort state possible without affecting the local QoS. This type of auction based multi agent system has previously been successfully implemented in district heating networks in order to achieve DSM [15]. Strategic decisions are made based on global or local views within the environment, and the specific optimization actions rely on continuous sensor data. Global knowledge is needed in order to identify individual consumer agents able and willing to participate in local task accomplishment. Without sufficient communication abilities the auction process is not able to function, thus making it more difficult to distribute DSM tasks while taking into account the local consumer agent comfort state. By using an action process it is possible to distribute the complexity and computational effort, since all reasoning and planning about the delivered QoS is done by the consumer agents. This leads to a more scalable and extendable system.

2.4. Levels of Agent Knowledge

In the described DSM system the agents are able to communicate freely among their peers, in order to continuously propagate system status based on available sensor data and perform coordinated task sharing when needed. In this study we compare the performance of such a fully functional system with two other systems displaying increasingly worse availability of sensor data. These three different scenarios are based on the level of system wide knowledge available to the participating agents; global, partial and local. We choose to compare these specific three levels of system wide knowledge because they correspond to infrastructural prerequisites which can normally be found in actual physical systems, and because they display a broad and clear view of the problem discussed.

Global Knowledge This is the normal operational view for the MAS used to operate the DSM strategies. The producer agents are able to continuously supervise the use of production utility and are able to instigate DSM actions as need arises. Each DSM action is divided into control tasks which can be distributed throughout the network of consumer agents by system wide auctions. The consumer agents are able to uphold their individual QoS by deciding when and how to participate in these auctions, i.e. a DSM task is never forced upon a consumer agent against its will.

Partial Knowledge The producer agents are able to supervise the consumption of production utility, but they are not able to communicate local sensory data with consumer agents. This means that cooperative behaviour through auctioning is not possible. A producer agent is, however, still able to instigate uninformed DSM actions. This is normally done by using predefined activation lists, which force consumer agents to implement DSM tasks in a round-robin fashion. Since no communication of consumer sensor data is available it is not possible for the producer agents to gain any feedback about the impact of these DSM tasks on the QoS delivered to the local consumer. The local consumer agent is still working to uphold its own QoS, and it might decide not to implement the DSM task appointed to it. Either way, as the consumer sensor data communication is impaired, the producer agent will never gain any knowledge about what decision the consumer agent takes.

Local Knowledge In this scenario the producer agents have little or no knowledge about the continuous consumption of production utility, and they do not have any possibility at all to implement any DSM actions, either by cooperation or force. The consumer agents still have access to their own local sensor data but they cannot successfully communicate this to other agents within the MAS. This basically means that they act oblivious to the state of any other agent. In such a system the consumer agents are often assigned the task of keeping the local utility use to a minimum while upholding the desired QoS. Depending on the situation such behaviour might or might not be for the good of the global system state, but the consumer agent will never know anything about this.

3. The Experiment

In this study we investigate the effects of different levels of availability of sensor data within an operational agent based control system. Under normal circumstances the agent system is based on cooperative behaviour which is in turn heavily dependent on functioning and reliable communication of high quality sensor data. Performance of the multi-agent system will deteriorate if the availability or quality of sensor data declines. We have studied how extensive this deterioration will be in a practical setting, i.e. how will the quality of the communication underlying the decision-making affect the overall performance from a system wide perspective. The case study is based on operational data from an agent based control system operational in a district heating network in the town of Karlshamn in the south of Sweden [14], [15]. This data is used as input when simulating the various scenarios described in the previous sections.

3.1. Reference Data

District heating networks are good examples of the described problem domain as they display most, if not all, of the mentioned characteristics. The reference data in question is collected during a twenty-four hour period with no DSM strategy active, i.e. no external control is applied to the consumers. The data is representative of normal usage during wintertime when the heating demand is substantial. The energy consumption in a district heating system is measured by combining the flow of the water with the primary supply temperature of the water. During the course of a single day the primary supply temperature in this district heating network is rather constant so the flow is a good estimate of the total energy use.

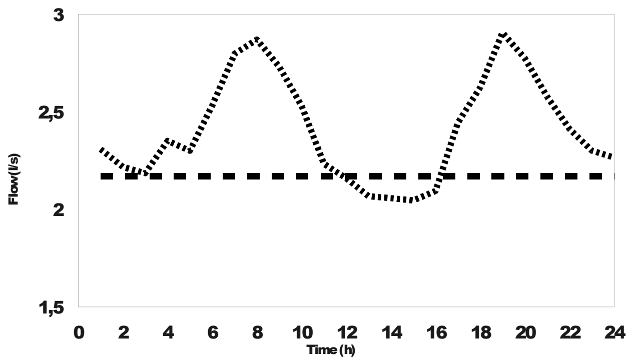


Fig. 2. Reference data (dotted) and wanted DSM level (dashed)

Figure 2 shows the flow data from the Karlshamn district heating network during a full twenty-four hour time period. The straight dashed line shows the calculated wanted level of energy consumption which the producer agent uses as a benchmark during this specific day. This level of consumption is based on a solution of the Economic Dispatch Problem and the Unit Commitment Problem. The peaks above the dashed line represents peak loads which would need to be satisfied by using financially and environmentally unsound fossil fuel. In other words, the global goal of the agent system is to keep the consumption as close to the straight dashed line as possible.

3.2. Utility Evalution

The consumer agents all have different comfort constraints based on a function of size, shape and material of the individual building, i.e. the amount of thermal buffer available [12]. In the operational system each consumer agent has access to sensor and actuator data through an I/O hardware platform, which enables the agent to measure the physical behaviour of the heating system within the building as well as the outdoor temperature. Based on this data the agent can calculate the indoor temperature which is then used as the basis for the agents own comfort evaluation. Each agent has a value of wanted indoor climate, and constantly tries to minimize all deviation from this value. However, in order to participate in achieving the global system goal an individual consumer agent can accept smaller deviations from this value under shorter periods of time, as this will not affect the indoor climate to a degree where the inhabitants will notice it. The consumer agent has two basic values to consider, namely the comfort level and the buffer level. These are connected with a delay, so that it is possible to adjust the level of the energy buffer during shorter periods of time without the comfort level having the time to react. It is possible for the consumer agent to use more than the available buffer, but then the comfort level will start to fall. Under normal circumstances a consumer agent will be very unwilling to use more the available buffer, although it might do so during shorter periods of time in order to achieve some global goal. When a consumer agent responds to an auction it will use its currently available buffer level as the price it is willing pay for implementing a single DSM task. This process will ensure that only the consumer agent which is best suited at any given time, i.e will be least in risk of compromising its comfort level, will be appointed the DSM task in question. We evaluate the performance of the consumer agents by measuring how they choose to use their individual buffers.

The producer agent in the system use the energy delivered to the area as input for its calculations concerning the need for DSM actions. The optimization strategy used in this experiment is that of peak shedding, i.e. at any given moment when the total energy use exceeds a certain threshold the producer agent will try to convince the consumer agents to lower their local energy usage in a coordinated fashion. When implementing this strategy the producer wants to limit the utility consumption down to the wanted threshold, as forcing the consumption down even further than the threshold will reduce sale of utility produced by economically viable production supplies. Therefore we measure the success of these system wide optimization actions by measuring any deviations between the wanted fluid flow value and the resulting actual flow level. By analyzing these values it is possible to evaluate to what extent the overall agent system accomplishes its objective, i.e. to uphold the wanted DSM strategy level without jeopardizing the comfort among the consumer agents. If the agent system is to be considered successful in its endeavors it will have to fulfill both these requirements.

3.3. Availability of Sensor Data

The agents within the Karlshamn district heating area communicate through a LAN network and have direct access to high quality sensor data. In this sense they are extremely spoiled, as this kind of communication solution is rarely part of the hardware set-up in similar real-world environments. As building a physical network and sensory infrastructure can be costly, similar agent systems are usually limited to using communication techniques such as GSM-modems, radio link or standard limited master/slave networks to evaluate operational data. With such solutions there is often limitations in regards to communication bandwidth and temporary sensor breakdowns are not uncommon. In this experiment we evaluate the impact of such system deterioration by simulating different levels of availability of the sensor data. In order to do this we model the three previously described scenarios, i.e. global knowledge, partial knowledge and local knowledge.

In the global scenario the producer agent and the consumer agents are allowed to communicate freely throughout every time step in the simulation. The producer agent can instigate auction processes according to its own desires, and the consumer agents are able to respond to this.

During the partial scenario there is only one-way communication possible from the producer agent to the consumer agents. The producer agent can distribute DSM tasks, and does so according to a previously defined static list. The producer agent can distribute several DSM individual tasks during each time step. A consumer agent might implement such a DSM task or it might not, depending on the current level of its internal buffer. Any which way, the producer agent will not receive any response about this.

In the local scenario there is no communication what so ever between the agents. The consumer agents can perform local load control, but this is done purely based on local knowledge. The local load is made up of a combination of energy used for space heating and tap water heating. During local load control, the consumer agent will try to limit local space heating when there is local tap water usage. The tap water usage is randomized within the simulation.

3.4. Simulation

We use real operational data from the Karlshamn district heating network as input into the simulation model, where actual flow data is used as initial values for the calculations. The implemented agent system is functioning according to the same principles as previously described. In the simulation there are fourteen active agents; one producer agent and thirteen consumer agents. By simulating the described levels of agent knowledge we can evaluate the performance of the agent system during different scenarios.

A simulation run begins by calculating specific solutions to the Economic Dispatch Problem and the Unit Commitment Problem. These solutions yield a wanted system wide consumption level for each time step throughout the day. This wanted consumption level is then used by the producer agent as a decision basis, when deciding when and how to instigate DSM actions throughout each time step. The consumer agents starts the simulation with full available buffer levels. This buffer level is then adjusted through each time step as they perform DSM tasks, which in turn makes it possible to calculate the comfort levels for each time step.

For each time step the producer agent then decides if there is any need for DSM actions based on the current flow level in relation to the wanted flow level. If it deems this necessary it will try to distribute individual DSM tasks. Depending on the specific scenario this will be achieved differently. The system wide consumption is then calculated and used as input into the next time step.

4. Results

We evaluate the different scenarios according to how well they manage to achieve the DSM strategy in question while staying within the comfort constraints set by the consumer agents. We present how well the agent system upholds the DSM strategy within the three different scenarios, and then we show how well the system manages to keep itself within the available energy buffer, and thus the consumer comfort constraints during these same scenarios.

4.1. Control Strategy

The control strategy is evaluated by measuring the flow of hot water into the area. Energy usage in a district heating network is measured by combining the temperature of the water with the flow. Since the supply water temperature in the primary network is more or less stable throughout a single day the flow in itself gives a good estimation of the energy usage within all the buildings. Figure 2 in the previous section shows the reference data without any DSM strategy active, i.e. this is what the overall consumption pattern looks like in a district heating network without any agent system installed. In Figure 3, Figure 4 and Figure 5 we show the flow data achieved during the three different scenarios in relation to the wanted DSM strategy.

It is clearly visible that the flow value in the global scenario (Figure 3) most closely resembles the desired DSM strategy, with the partial scenario (Figure 4) being somewhat worse, and finally the local scenario (Figure 5) showing a distinct lack in ability to achieve the desired level of consumption. To make these results clearer we also summarize the deviation of the scenarios for every time frame throughout the simulation. This value has a theoretical optimum at zero, i.e no deviation from the desired DSM level what so ever. The values are then normalized around the value achieved by the global scenario. These results are shown graphically in Figure 6 and the actual numbers in Table 1.

4.2. Agent Buffer Usage

The level of comfort is dependent on the buffer used by each individual consumer agent. Every agent has an maximum allowed buffer usage of 1, with a

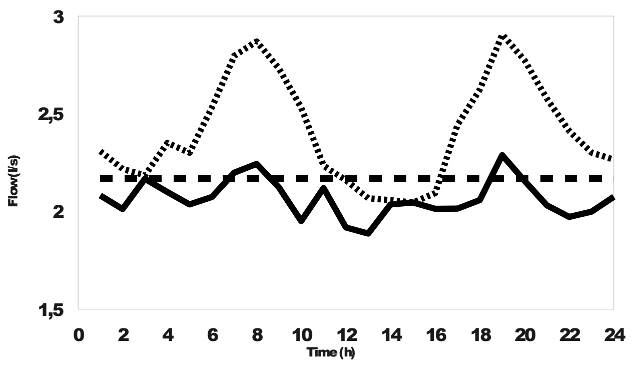


Fig. 3. Global scenario. Agent performance (continuous), reference data (dotted) and wanted DSM level (dashed)

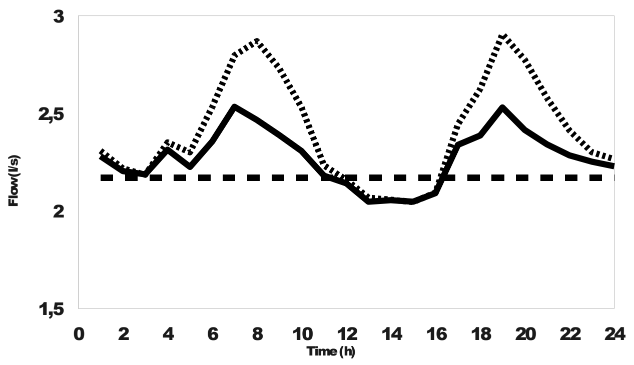


Fig. 4. Partial scenario. Agent performance (continuous), reference data (dotted) and wanted DSM level (dashed)

Table 1. DSM strategy deviation values

Scenario	Value
Global	1
Partial	6.75
Local	11.07

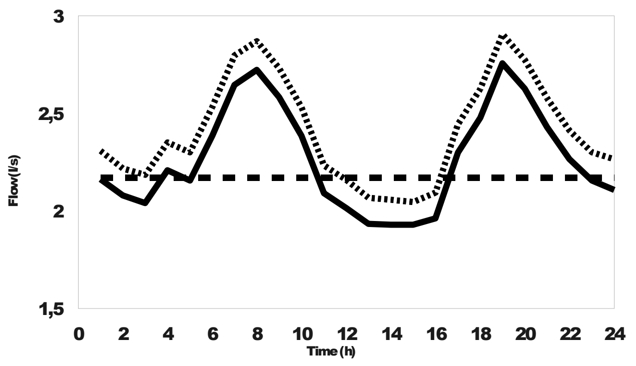


Fig. 5. Local scenario. Agent performance (continuous), reference data (dotted) and wanted DSM level (dashed)

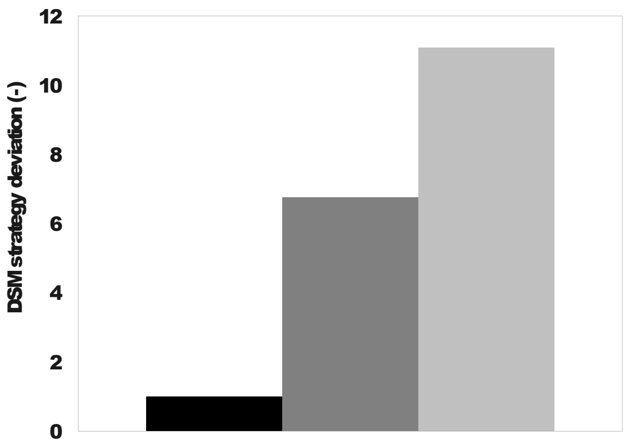


Fig. 6. DSM strategy deviation. Global scenario (black), partial scenario (dark grey) and local scenario (light grey)

minimum of 0. The level of comfort will not be negatively effected by a usage between 1 and 0. If the buffer usage is above 1 the consumer agent has used more than the allowed buffer and the comfort can be in jeopardy if such a status is allowed to continue for a longer period of time. In other words a consumer agent has an optimal buffer usage of 1, i.e. the agent participates in achieving the global goal as much as possible but does this without sacrificing its desired comfort level. The values for the individual consumer agents are shown in Figure 7, together with a theoretical optimum of 1. The numerical values are then showed in Table 2.

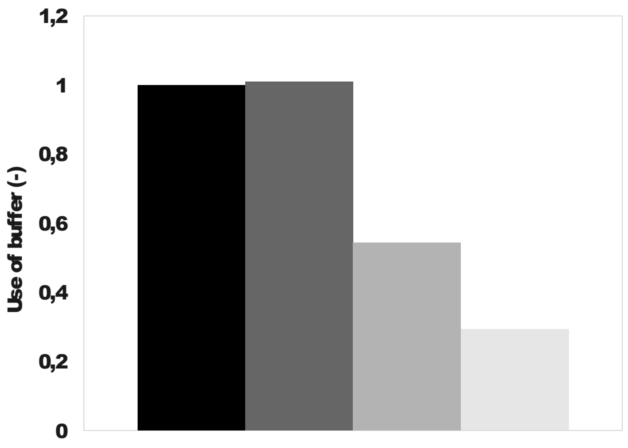


Fig. 7. Individual buffer usage. Theoretical optimum (black), global scenario (dark grey), partial scenario (grey) and local scenario (light grey)

Table 2. Agent comfort and buffer usage for each individual consumer agent

Scenario	Value
Optimum	1
Global	1.01
Partial	0.54
Local	0.29

Figure 8 shows the dynamic system wide buffer usage during the whole time period. The range on the y axis is dependent on the amount of consumer agents, since every such agent has a optimal buffer usage of one. In this case study we have thirteen agents, so an optimal usage of the system wide buffer would obviously be 13. In the global and partial scenarios the buffer usage clearly follows the reference data as the agents continuously try to counter the varying consumption.

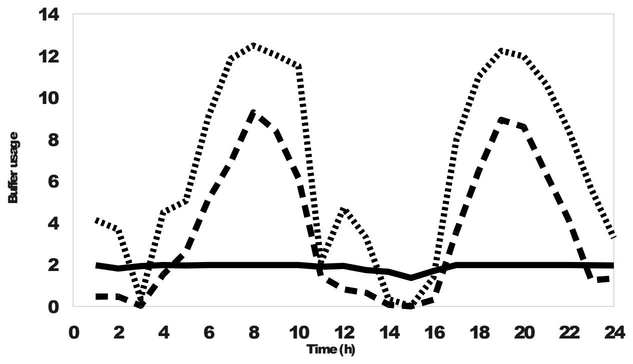


Fig. 8. Total buffer usage. Global scenario (dotted), partial scenario (dashed) and local scenario (continuous)

5. Discussion

Multi-agent system solutions being applied to the physical processes described in this paper are heavily dependent on the availability of high-quality sensor data to function properly. This study quantifies the way system performance rapidly deteriorates as the availability of high-quality sensor data is reduced. The evaluation is based on the systems ability to adhere to the wanted control level while maintaining an acceptable level of agent comfort. By combining the control strategy and agent comfort values it is then possible to evaluate the performance.

It is important to factor in both the DSM strategy and the consumer agent comfort value when evaluating an implementation for handling DSM within the problem domain. If a system is only evaluated on the basis of its ability to adhere to the DSM strategy it might give rise to problems on the consumer side as no consideration is given to upholding a sufficient level of QoS.

The notion of what is an acceptable level of the control strategy value is dependant on the process in question. In a district heating network there are several benefits of having the ability to perform load control, the most apparent being the ability to shed peak loads in order to avoid using expensive and environmentally unsound peak load fuels. In our case the global scenario would be considered acceptable since it manages to shed the peaks.

6. Conclusions

The local scenario is similar to a type of control system that is often implemented in both electrical grids and district heating networks, as a local uninformed optimization technique. This study indicates that such systems have little global effect in regards to overall production optimization strategies. As Figure 5 and Figure 8 clearly shows the local scenario is inadequate in order to handle any system wide DSM strategy. The reason that the local scenario never goes beyond a certain level in Figure 8 is that the consumer agents are only reacting to their own local peak loads, which are well beyond their own capacity to handle. This is due to the fact that individual peaks are much larger than any individual buffer, so in the local scenario some agents are always maximizing their use of their individual buffer, but without the ability to somehow distribute the load through the producer agents their efforts will always fall short on a system wide scale. This shows a clear advantage of the two distributed DMS solutions in relation to any local efforts, which can never hope to counter system wide peaks.

Figure 8 also shows that producer agent knowledge is needed in order to dynamically counter the user demand in regards to the DSM strategy. This is also the buffer usage, which shows that the partial scenario is not able to fully use the available buffer. This is due to the fact that the agents cannot perform cooperative work. The difference between the global and partial scenarios in Figure 8 basically shows the superiority of agent cooperation versus centralized enforcement. The lower use of available buffer of the partial scenario is caused by the fact that although the consumer agent is handed a DSM task, it can choose not to implement the task if the agent considers it to jeopardize its internal QoS level. Since the producer agent never receives any feedback about this, it will not be able to distribute the task to another consumer better suited for the task, and hence the system will on average not utilize the maximum of the available buffer.

Figure 8 shows that the global scenario is close to using the maximum available buffer on several occasions, while neither the partial or the local scenarios are close to utilizing their full DSM potential. The global scenario is rather close to achieving the DSM strategy, but it does not manage to fully adhere to the wanted level. To achieve this would require the system to continuously foresee and counter highly stochastic microscopic behaviour within the process, which is not feasible in a practical setting.

In this paper we have shown that distributed multi agent systems based on cooperative auctioning are able to achieve the studied DSM strategy, while maintaining an acceptable level of QoS. As the availability and quality of the sensor data diminishes the system performance deteriorates, first into the equivalence of static distributed models and then into the equivalence of simple local optimization models. This shows that real-time cooperative behaviour among communicating agent nodes is needed in order to successfully implement DMS in real world applications, and that indirect methods, like differentiable taxation, or uninformed local optimization is not able to produce the coordinated system-wide behaviour required.

7. Future Work

This paper is the result of an case study in regards to sensor data utilization within industrial multi-agent system applications. In the future we will use this as groundwork while incorporating the financial factors underlying the discussion, in order to further study the economical effects found within such systems.

According to Figure 1 we have only used an action process in order to evaluate the the different scenarios. In the future we intend to expand this study by using other collaboration techniques within the different scenarios.

Another issue is the formalization of a model for the follow-up of DSM actions. Sometimes actions are taken when there is no need for them, and other times actions are needed without them being implemented. By improving our understanding and control of this process it should be possible to better utilize the individual consumer agent buffer. This could be used when combining the multi agent system with a continuous optimization model in order to dynamically follow the process.

8. Acknowledgements

The operational data used in this study was supplied by NODA Intelligent Systems. The project has also been supported by Karlshamns Energi and Karlshamnsbostader.

References

1. Arvastson, L.: Stochastic Modeling and Operational Optimization in District Heating Systems. Lund Institute of Technology (2001)
2. Aune, M.: Energy technology and everyday life the domestication of ebox in norwegian households. In: Proceedings of ECEEE Summer Study (2001)
3. C.E., L., G.L., V.: Hierarchical economic dispatch for piecewise quadratic cost functions. In: Trans Power Apparatus Systems, vol. 103-6, pp. 1170–1175. The Institute of Electrical and Electronics Engineers, Inc (1984)
4. Dash, R., Rogers, A., S., R., Roberts, S., Jennings, N.R.: Constrained bandwidth allocation in multi-sensor information fusion: A mechanism design approach. In: Proceedings of The Eight International Conference on Information Fusion. Philadelphia, PA, USA (2005)
5. Dotzauer, E.: Simple model for prediction of loads in district-heating systems. Applied Energy 73(3–4), 277–284 (2002)
6. Dutta, P.S., Goldman, C., Jennings, N.R.: Communicating effectively in resource-constrained multi-agent systems. In: Proceedings of the 20th International Joint Conference on AI (IJCAI). Hyderabad, India (2007)

7. Goldman, C.V., Zilberstein, S.: Optimizing information exchange in cooperative multi-agent systems. In: Proceedings of Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 137–144 (2003)
8. Jayasima, D.N.: Fault tolerance in multisensor networks. IEEE Transactions on Reliability 45(2), 308–315 (1996)
9. Koay, C.A., Lai, L.L., Lee, K., Lu, H., Park, J.B., Song, Y.H., Srinivasan, D., Vlachogiannis, J.G., Yu, I.K.: Applications to power system scheduling. In: Lee, K.Y., El-Sharkawi, M.A. (eds.) Modern Heuristic Optimization Techniques. The Institute of Electrical and Electronics Engineers, Inc (2008)
10. Lesser, V., Ortiz, C., Tambe, M.: Distributed Sensor Networks: a multiagent perspective. Kluwer Publishing (2003)
11. Nordvik, H., Lund, P.E.: How to achieve energy efficient actions as an alternative to grid reinforcement. In: Proceedings of ECEEE Summer Study (2003)
12. Olsson Ingvarsson, L., Werner, S.: Building mass used as short term heat storage. In: Proceedings of The 11th International Symposium on District Heating and Cooling. Reykjavik, Iceland (2008)
13. Shen, J., Lesser, V., Carver, N.: Minimizing communication cost in a distributed bayesian network using a decentralised mdp. In: Proceedings of Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 678–685 (2003)
14. Wernstedt, F., Davidsson, P., Johansson, C.: Demand side management in district heating systems. In: Proceedings of Sixth International Conference on Autonomous Agents and Multiagent Systems (AAMAS). Honolulu, Hawaii, USA (2007)
15. Wernstedt, F., Johansson, C.: Intelligent distributed load control. In: Proceedings of The 11th International Symposium on District Heating and Cooling. Reykjavik, Iceland (2008)

Christian Johansson is a Ph.D. student in Computer Science at Blekinge Institute of Technology (BTH). His main focus is optimization and system-wide coordination in distributed systems with applications in energy systems.

Fredrik Wernstedt has a BSc degree in Computer Science from Blekinge Institute of Technology and a BSc in Nautical Science from University of Kalmar. He received his Licentiate degree in Computer Science from BTH in 2003 and his Ph.D. in Computer Science from BTH in 2005. His main interest is autonomous agents and multi-agent systems, particularly for optimization of operational systems.

Paul Davidsson is a Professor in Computer Science at Blekinge Institute of Technology. Paul is the manager of the Distributed and Intelligent Systems Laboratory at BTH and received his Ph.D. in Computer Science from Lund University 1996. His research interests includes the theory and application of multi-agent systems and machine learning.

Received: January 18, 2010; Accepted: May 3, 2010.

UDC 004.896, DOI:10.2298/CSIS100209015D

A Layered Rule-Based Architecture for Approximate Knowledge Fusion*

Barbara Dunin-Kęplicz[1,2], Linh Anh Nguyen[1], and Andrzej Szałas[1,3]

[1] Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
{keplicz,nguyen,andsz}@mimuw.edu.pl
[2] Institute of Computer Science, Polish Academy of Sciences
Ordona 21, 01-237 Warsaw, Poland
[3] Dept. of Computer and Information Science, Linköping University
SE-581 83 Linköping, Sweden

Abstract. In this paper we present a framework for fusing approximate knowledge obtained from various distributed, heterogenous knowledge sources. This issue is substantial in modeling multi-agent systems, where a group of loosely coupled heterogeneous agents cooperate in achieving a common goal. In paper [5] we have focused on defining general mechanism for knowledge fusion. Next, the techniques ensuring tractability of fusing knowledge expressed as a Horn subset of propositional dynamic logic were developed in [13,16].

Propositional logics may seem too weak to be useful in real-world applications. On the other hand, propositional languages may be viewed as sublanguages of first-order logics which serve as a natural tool to define concepts in the spirit of description logics [2]. These notions may be further used to define various ontologies, like e.g. those applicable in the Semantic Web. Taking this step, we propose a framework, in which our Horn subset of dynamic logic is combined with deductive database technology. This synthesis is formally implemented in the framework of HSPDL architecture. The resulting knowledge fusion rules are naturally applicable to real-world data.

Keywords: knowledge fusion, multi-agent systems, approximate reasoning, rule-based systems.

1. Introduction

In this paper we investigate a framework for fusing approximate knowledge obtained from various distributed, heterogenous knowledge sources. This issue is substantial in modeling multiagent systems, where a group of loosely coupled heterogeneous and autonomous agents cooperate in achieving a common

* Supported by the Polish MNiSW grant N N206 399334 and by grants from the Swedish Foundation for Strategic Research (SSF), Strategic Research Center MOVIII and the Swedish Research Council (VR) Linnaeus Center CADICS.

goal. Information exchange, leading ultimately to knowledge fusion, is a natural and vital ingredient of cooperation, coordination and negotiations, which constitute paradigmatic activities of advanced multiagent systems. This is particularly visible, when environment model an agent has access to and from which it can reason is assumed to be limited by inherent perceptual limitations. There are many reasons why approximate approaches are needed in this context, including the following:

- sensor measurements, video streams, etc. are always approximate in their very nature – in fact one can never expect precise, accurate data from such sources
- even in the case of idealized perfect perception agents may draw substantially different conclusions, based on their circumstances. For example, due to different camera angles and light reflections one agent may draw a conclusion that a given object is red while another agent may classify the object to be brown. As this is highly contextual, probabilistic sensor models may be of little help and qualitative approximate reasoning may be needed.

As discussed in [7], in the past several years attempts have been made to broaden the traditional definition of data fusion as state estimation via aggregation of multiple sensor streams. There is still a need to broaden the definition to include the many additional processes used in all aspects of data and information fusion identified in large scale distributed systems. One of the more successful proposals for providing a framework and model for this broadened notion of data fusion is the data fusion model [33] and its revisions [29,21]. In [29] for example, data fusion is defined as “the process of combining data or information to estimate or predict entity states” and the data fusion problem “becomes that of achieving a consistent, comprehensive estimate and prediction of some relevant portion of the world state”.

There is a variety of possibilities to model approximate knowledge [6,11,10,9,20,23,28,34,35,36]. In this presentation we have chosen a generalization of rough sets and relations [27]. In contrast to [27] where only equivalence relations are considered, our approach depends on allowing arbitrary similarity relations. In order to construct approximations, a covering of the underlying domain by similarity-based neighborhoods is used here. Resulting approximations have been shown to be useful in applications requiring approximate knowledge structures [6].

There are many choices as to possible constraints to be placed on the similarity relation used to define approximations. The basic requirement is that the lower approximation is included in the upper one of any set/relation. This is equivalent to the seriality of similarity relations (see [8]), which we set as the only requirement. On the other hand, one might not want the relation be transitive since similar objects do not naturally chain in a transitive manner (see, e.g., [4,14,6,22,31]). Similarity measures on sets that could be adapted to the context of approximate reasoning we deal with have been intensively studied in the area of computer vision and fuzzy sets (see, e.g., [17,32]).

The focus of this paper is approximate knowledge fusion based on the idea of approximations. Our starting point is [5], where a framework for knowledge fusion in multi-agent systems is introduced. Agent's individual perceptual capabilities are represented by similarity relations, further aggregated to express joint capabilities of teams. The aggregation expressing a shift from individual to social level of agents' activity has been formalized by means of dynamic logic. The approach of [5], as using the full propositional dynamic logic, does not guarantee tractability of reasoning [18]. To overcome this constraint we adapt the techniques of [24,25,26] to provide an engine for tractable approximate database querying restricted to a Horn fragment of serial propositional dynamic logic, denoted by HSPDL.

1.1. Contributions of the Paper

In this paper we substantially extend our work presented in [13], where we have concentrated on techniques allowing one to query HSPDL databases in a tractable manner. Propositional logics have a very limited expressivity and may seem too weak to be useful in real-world applications. For example, one cannot express rules using even very basic arithmetics, like in rules (12) and (13) of Section 5. On the other hand, propositional languages may be viewed as sublanguages of first-order logics which serve as a natural tool to define concepts in the spirit of description logics [2]. Additionally, allowing one to query other modules of the system, not necessarily propositional (but returning Boolean values), provides a powerful tool. Taking this step, we propose a framework, in which our Horn subset of dynamic logic is combined with deductive database technology [1], allowing one to express an advanced knowledge fusion applicable in real-world data.

The synthesis of the two formalisms naturally leads to a layered architecture, with the lowest layer containing raw data and basic knowledge structures, the middle one allowing to express rules specifying knowledge fusion, new concepts and their approximations, and the upper level providing the resulting knowledge database. We provide this architecture with both the formal semantics and tractable querying machinery. This makes the framework a pragmatic, rich formalism to be directly used in the chosen application domain.

The framework we propose can also be adapted to other propositional logics designed as a specification and computation tool, e.g., for multiagent systems as well as other robotics and software systems. This bridges the gap between propositional languages and real-world, usually non-propositional data.

1.2. The HSPDL Architecture

There are three main layers of HSPDL architecture (see Figure 1):

– the lower layer consisting of perception data, knowledge databases, results of classifiers, etc.

– the middle layer containing HSPDL rules to define new concepts and their approximations
– the upper layer using the data resulting from the lower layers, i.e., fused concepts and approximations, to define new advanced rules and obtain new facts.

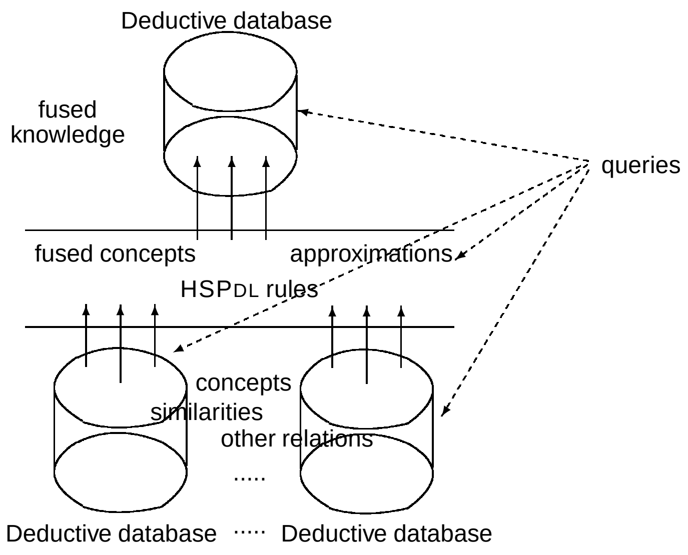


Fig. 1. The HSPDL architecture.

The architecture is highly independent of a particular technology. They can be founded, e.g., on SQL databases or any other software systems.[4] We only make the following assumptions:

– the lower layer is conceptually a database storing relations (but, as indicated, not necessarily a relational database)
– the lower layer provides a programming interface allowing:
  • the middle layer to ask queries about concepts (unary relations) and similarity relations (binary relations)
  • the upper layer to ask queries about any relations represented in the lower layer.
– the lower (respectively, upper) layer computes answers to queries in time polynomial in the size of its domain.

The context in which the middle layer rules appear may be very expressive. It might be the case that all tractable knowledge fusion procedures become

[4] In this paper we shall mainly focus on deductive databases technology using Datalog as its query language (see, e.g., [1]).

expressible without using HSPDL. However, we insist that the introduction of HSPDL rules in the middle layer is both well motivated and intuitively appealing. In the first place, some middle layer constructs are not expressible in many database technologies, including standard SQL and Datalog. Otherwise, the resulting rules happen to be indirect and lead to programs difficult to understand and analyze, while HSPDL rules are fully declarative,

Layered architectures in similar but substantially different contexts have been considered, e.g., in [15,31].

1.3. The Paper Structure

The paper is structured as follows. In Section 2 we recall the serial propositional dynamic logic. Computational aspects of its Horn fragment HSPDL are discussed in Section 3. Section 4 is devoted to combining HSPDL with Datalog. Section 5 illustrates possible applications of the introduced framework on an example. Finally, Section 6 concludes the paper.

2. Serial Propositional Dynamic Logic

2.1. Language and Semantics of SPDL

Let us define *serial propositional dynamic logic* (SPDL). The key idea is to provide calculus on similarity relations rather than on programs. This somehow unusual move allows us to reason about similarities using the whole apparatus of dynamic logic, where "programs" are replaced by similarity relations.

Let SNames denote the set of *similarity relation symbols*, CNames denote the set of *concept names* (i.e., propositions), and INames denote the set of *individuals*. We assume that INames is finite and non-empty. We use letters like σ to indicate elements of SNames, use letters like p, q to indicate elements of CNames, and use letters like a, b, c to indicate elements of INames.

Definition 1. *Formulas* and *similarity expressions* (of SPDL) are respectively defined by the two following BNF grammar rules:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \langle\alpha\rangle\varphi \mid [\alpha]\varphi$$
$$\alpha ::= \sigma \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?$$

Operator $;$ is called the *composition*, $\cup$ the *union*, * the *iteration* and $\varphi?$ the *test operator*. $\triangleleft$

We use letters like α, β to denote similarity expressions, and use letters like φ, ψ to denote formulas.

Intuitively,

- $\alpha_1;\alpha_2$ stands for a set-theoretical composition of relations α_1 and α_2
- $\alpha_1 \cup \alpha_2$ stands for set-theoretical union of relations α_1 and α_2
- α^* stands for the reflexive and transitive closure of α

– $\varphi?$ stands for the test operator.

Operators $\langle\alpha\rangle$ and $[\alpha]$ are modal operators of the dynamic logic with the following intended meaning:

– $\langle\alpha\rangle\varphi$: "there is an object similar w.r.t. α to a given object and satisfying formula φ"
– $[\alpha]\varphi$: "all objects similar w.r.t. α to a given object satisfy φ".

The following definitions naturally capture these intuitions. Observe, however, that rather than possible worlds or states, objects are used as elements of domains of Kripke structures.

Definition 2. A *Kripke structure* is a pair $\mathcal{I} = \langle\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set of *objects*, and $\cdot^{\mathcal{I}}$ is an interpretation function that maps each individual a to an element $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each concept name p to a subset $p^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and each similarity relation symbol σ to a binary relation $\sigma^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$. $\lhd$

The interpretation function is extended for all formulas and similarity expressions as follows:

$$\begin{array}{lll}\top^{\mathcal{I}} = \Delta^{\mathcal{I}} & (\neg\varphi)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus \varphi^{\mathcal{I}} & \\ (\varphi \wedge \psi)^{\mathcal{I}} = \varphi^{\mathcal{I}} \cap \psi^{\mathcal{I}} & (\varphi \vee \psi)^{\mathcal{I}} = \varphi^{\mathcal{I}} \cup \psi^{\mathcal{I}} & (\varphi \to \psi)^{\mathcal{I}} = (\neg\varphi \vee \psi)^{\mathcal{I}}\end{array}$$

$$\begin{array}{l}(\langle\alpha\rangle\varphi)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y\, [\alpha^{\mathcal{I}}(x,y) \wedge \varphi^{\mathcal{I}}(y)]\} \\ ([\alpha]\varphi)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y\, [\alpha^{\mathcal{I}}(x,y) \to \varphi^{\mathcal{I}}(y)]\}\end{array}$$

$$\begin{array}{lll}\multicolumn{3}{l}{(\alpha;\beta)^{\mathcal{I}} = \alpha^{\mathcal{I}} \circ \beta^{\mathcal{I}} = \{(x,y) \mid \exists z\, [\alpha^{\mathcal{I}}(x,z) \wedge \beta^{\mathcal{I}}(z,y)]\}} \\ (\alpha \cup \beta)^{\mathcal{I}} = \alpha^{\mathcal{I}} \cup \beta^{\mathcal{I}} & (\alpha^*)^{\mathcal{I}} = (\alpha^{\mathcal{I}})^* & (\varphi?)^{\mathcal{I}} = \{(x,x) \mid \varphi^{\mathcal{I}}(x)\}.\end{array}$$

We sometimes write $\mathcal{I}, x \models \varphi$ to denote $x \in \varphi^{\mathcal{I}}$. For a set Γ of formulas, we write $\mathcal{I}, x \models \Gamma$ to denote that $\mathcal{I}, x \models \varphi$ for all $\varphi \in \Gamma$. If $\mathcal{I}, x \models \Gamma$ for all $x \in \Delta^{\mathcal{I}}$ then we call $\mathcal{I}$ a *model of* Γ. If $\varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}$ then we say that φ is valid in $\mathcal{I}$.

When dealing with the data complexity of the instance checking problem, without loss of generality we can assume that both the sets SNames and CNames are finite and fixed.

Definition 3. The *size of a Kripke structure* $\mathcal{I}$ is defined to be

$$|\Delta^{\mathcal{I}}| + \Sigma_{p \in \mathrm{CNames}} |p^{\mathcal{I}}| + \Sigma_{\sigma \in \mathrm{SNames}} |\sigma^{\mathcal{I}}|.$$

The *length of a formula* is the number of symbols occurring in it. The *size of a set of formulas* is defined to be the sum of the lengths of its formulas. $\lhd$

Lemma 1. *Given a Kripke structure $\mathcal{I}$ with domain of size n and a formula φ with length m, the set $\varphi^{\mathcal{I}}$ can be computed in $O(m \times n^3)$ steps.*

Proof. Just notice that the complexity of computing the transitive closure of a binary relation is $O(n^3)$ (see, e.g., [3]). $\lhd$

For every $\sigma \in \mathrm{SNames}$, we adopt the axioms

$$[\sigma]\varphi \to \langle\sigma\rangle\varphi \tag{1}$$

(or $\langle\sigma\rangle\top$, equivalently). It is well known (see, e.g., [8,30]) that (1) corresponds to the *seriality property*:

$$\forall x \exists y\ \sigma^{\mathcal{I}}(x, y). \tag{2}$$

Therefore we have the following definition.

Definition 4. By an *admissible interpretation for* SPDL we understand any Kripke structure $\mathcal{I}$ with all similarities $\sigma \in \mathrm{SNames}$ satisfying (2). We call such Kripke structures *serial*. $\triangleleft$

Note that we do not require a serial Kripke structure to satisfy the seriality condition $\forall x \exists y\, \alpha^{\mathcal{I}}(x, y)$ for every similarity expression α. This condition holds when α does not contain the test operator, but does not hold, e.g., for $\alpha = ((\neg\top)?)$.

2.2. Expressing Approximations in SPDL

Let us now explain how SPDL is used as a query language involving approximate concepts. First, observe that interpretations assign sets of objects to formulas. Therefore, it is natural to identify any formula with a query selecting all objects satisfying this formula.

In order to explain the role of similarities and modal operators, let us first recall the notion of approximations.

Definition 5. Let Δ be a set of objects and α be a similarity expression representing a serial binary relation on Δ. For $a \in \Delta$, by the *neighborhood of a w.r.t. α*, we understand the set of elements similar to a: $n^{\alpha} \stackrel{\text{def}}{=} \{b \in \Delta \mid \alpha(a, b)\}$.

For $A \subseteq \Delta$, the *lower and upper approximations of A w.r.t. α*, denoted respectively by A^{+}_{α} and $A^{\oplus}_{\alpha}$, are defined by

$$A^{+}_{\alpha} = \{a \in \Delta \mid n^{\alpha}(a) \subseteq A\}$$
$$A^{\oplus}_{\alpha} = \{a \in \Delta \mid n^{\alpha}(a) \cap A \neq \emptyset\}.$$ $\triangleleft$

The meaning of those approximations is illustrated in Figure 2. Intuitively, assuming that the perception of an agent is modeled by similarity expression α,

- $a \in A^{+}_{\alpha}$ means that all objects indiscernible from a are in A
- $a \in A^{\oplus}_{\alpha}$ means that there are objects indiscernible from a which are in A.

Note that seriality guarantees that the lower approximation of a set is included in its upper approximation. In fact, this is the weakest requirement regarding approximations. The following is often desirable in many applications

$$A^{+}_{\alpha} \subseteq A \subseteq A^{\oplus}_{\alpha}, \tag{3}$$

as, in fact, shown in Figure 2. This property corresponds to the reflexivity of the similarity relation expressed by α (see, e.g., [8,37,30]) and guarantees the following:

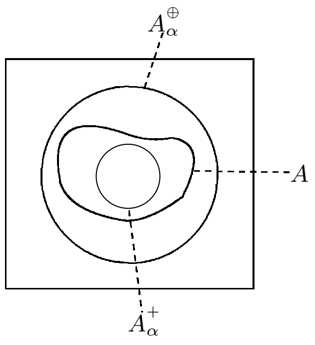


Fig. 2. Lower approximation A^{+}_{α} and upper approximation $A^{\oplus}_{\alpha}$ of a set A.

- $a \in A^{+}_{\alpha}$ means that, from the point of view of the agent, a surely is in A, since all objects indiscernible from a are in A
- $a \in A^{\oplus}_{\alpha}$ means that, from the point of view of the agent, a possibly is in A, since there are objects indiscernible from a which are in A.

Unfortunately, in some applications the set A is given solely via its approximations, so constraints (3) cannot be checked automatically. This is often the case of vague concepts lack precise definitions or lead to definitions unacceptable in applications due to its complexity or other issues. For example one could define a concept of a “dog” via genetic code what is not that much of help when classifying dogs in everyday life. Also, machine learned concepts are often approximated, as e.g., in version spaces (see [12]).

As an immediate consequence of Definitions 5 and 2 we have that:

$[\alpha]A$ expresses the lower approximation of A w.r.t. α, i.e., A^{+}_{α}, (4)

$\langle\alpha\rangle A$ expresses the upper approximation of A w.r.t. α, i.e., $A^{\oplus}_{\alpha}$. (5)

Remark 1. In the view of (4) and (5), axiom (1) expresses the property that the lower approximation of a set A w.r.t. any similarity expression α is included in its upper approximation. As indicated before, axiom (1) is equivalent to seriality expressed by (2). This justifies seriality to be the key requirement based on approximations. $\lhd$

2.3. The Horn Fragment HSPDL

In order to express tractable queries we restrict the query language to the Horn fragment HSPDL, defined below.

Definition 6. *Positive formulas* (of PDL), φ_{pos}, are defined by the following BNF grammar:

$$\begin{aligned}
\varphi_{pos} &::= \top \mid p \mid \varphi_{pos} \land \varphi_{pos} \mid \varphi_{pos} \lor \varphi_{pos} \mid \langle \alpha_{pos\Diamond} \rangle \varphi_{pos} \mid [\alpha_{pos\Box}]\varphi_{pos} \\
\alpha_{pos\Diamond} &::= \sigma \mid \alpha_{pos\Diamond} ; \alpha_{pos\Diamond} \mid \alpha_{pos\Diamond} \cup \alpha_{pos\Diamond} \mid \alpha^{*}_{pos\Diamond} \mid \varphi_{pos}? \\
\alpha_{pos\Box} &::= \sigma \mid \alpha_{pos\Box} ; \alpha_{pos\Box} \mid \alpha_{pos\Box} \cup \alpha_{pos\Box} \mid \alpha^{*}_{pos\Box} \mid (\neg\, \varphi_{pos})?
\end{aligned}$$

HSPDL *program clauses*, φ_{prog}, are defined by the following BNF grammar:[5]

$$\begin{aligned}
\varphi_{prog} &::= \top \mid p \mid \varphi_{pos} \to \varphi_{prog} \mid \varphi_{prog} \land \varphi_{prog} \mid \langle \alpha_{prog_\Diamond} \rangle \varphi_{prog} \mid [\alpha_{prog_\Box}]\varphi_{prog} \\
\alpha_{prog_\Diamond} &::= \sigma \mid \alpha_{prog_\Diamond} ; \alpha_{prog_\Diamond} \mid \varphi_{prog}? \\
\alpha_{prog_\Box} &::= \sigma \mid \alpha_{prog_\Box} ; \alpha_{prog_\Box} \mid \alpha_{prog_\Box} \cup \alpha_{prog_\Box} \mid \alpha^*_{prog_\Box} \mid \varphi_{pos}?
\end{aligned}$$

An HSPDL *logic program* is a finite set of HSPDL program clauses. The *Horn fragment* HSPDL for the problem of checking whether $\langle \mathcal{P}, \mathcal{A} \rangle \models_s \varphi(a)$ consists of HSPDL logic programs for $\mathcal{P}$ and positive formulas for φ. ◁

Example 1. The following formulas are HSPDL program clauses:

$$\begin{aligned}
&p \land q \land r \to s \\
&[\sigma_1]p \land \langle \sigma_2 \rangle q \to \langle \sigma_3 \rangle (r \land [\sigma_4]s) \\
&[(\sigma_1 \cup \sigma_2)^*]\big(\langle (\sigma_3 \cup \sigma_4)^* \rangle (p \lor q) \to [\sigma_3]\langle \sigma_4 \rangle r\big),
\end{aligned}$$

while the following formulas are not:

$$\begin{aligned}
&p \land q \to r \lor s \\
&p \to \langle \sigma_1 \lor \sigma_2 \rangle q \\
&p \to \langle \sigma^* \rangle q.
\end{aligned}$$ ◁

Let us now formally link SPDL with databases.

Definition 7.

- A *concept assertion* is an expression of the form $p(a)$, where p is a concept name and a is an individual. A *similarity assertion* is an expression of the form $\sigma(a, b)$, where σ is a similarity relation symbol and a, b are individuals.[6] An *ABox* is a finite set of concept assertions and similarity assertions.[7] The *size of an ABox* is the number of its assertions.
- Given a Kripke structure $\mathcal{I}$ and an ABox $\mathcal{A}$, we say that $\mathcal{I}$ is a *model of* $\mathcal{A}$, denoted by $\mathcal{I} \models \mathcal{A}$, if $a^{\mathcal{I}} \in p^{\mathcal{I}}$ for every concept assertion $p(a) \in \mathcal{A}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \sigma^{\mathcal{I}}$ for every similarity assertion $\sigma(a, b) \in \mathcal{A}$.
- Given an HSPDL logic program $\mathcal{P}$ and an ABox $\mathcal{A}$, we call the pair $\langle \mathcal{P}, \mathcal{A} \rangle$ an HSPDL *database*, with $\mathcal{A}$ as the extensional database and $\mathcal{P}$ as the intensional part. An SPDL *model of* $\langle \mathcal{P}, \mathcal{A} \rangle$ is a serial Kripke structure that is a model of both $\mathcal{P}$ and $\mathcal{A}$.
- Let $\langle \mathcal{P}, \mathcal{A} \rangle$ be an HSPDL database, φ be a positive formula, and a be an individual. We say that a *has the property* φ *w.r.t.* $\langle \mathcal{P}, \mathcal{A} \rangle$ *in* SPDL (or $\varphi(a)$ is a logical consequence of $\langle \mathcal{P}, \mathcal{A} \rangle$ in SPDL), denoted by $\langle \mathcal{P}, \mathcal{A} \rangle \models_s \varphi(a)$, if $a^{\mathcal{I}} \in \varphi^{\mathcal{I}}$ for every SPDL model $\mathcal{I}$ of $\langle \mathcal{P}, \mathcal{A} \rangle$. ◁

By the *instance checking* problem for HSPDL we mean the problem of checking whether $\langle \mathcal{P}, \mathcal{A} \rangle \models_s \varphi(a)$. The *data complexity* of this problem is measured when $\mathcal{P}$, φ and a are fixed (and compose a query), while $\mathcal{A}$ varies as input data.

[5] Notice the two occurrences of φ_{pos} in the grammar. We do not allow formulas of the form $\langle \alpha \cup \beta \rangle \varphi$ or $\langle \alpha^* \rangle \varphi$ to be HSPDL program clauses because they cause non-determinism.

[6] Similarity assertions correspond to role assertions of description logic.

[7] In [19], such an ABox is said to be *extensionally reduced*.

3. Computational Aspects of HSPDL

3.1. Ordering Kripke Structures

To construct least models for HSPDL we need the following definitions.

Definition 8. A Kripke structure $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ is said to be *less than or equal to* $\mathcal{I}' = \langle \Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'} \rangle$, denoted by $\mathcal{I} \leq \mathcal{I}'$, if for every positive formula φ and every individual a, $a^{\mathcal{I}} \in \varphi^{\mathcal{I}}$ implies $a^{\mathcal{I}'} \in \varphi^{\mathcal{I}'}$. $\lhd$

Definition 9. Given Kripke structures $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{I}' = \langle \Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'} \rangle$ and a binary relation $r \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}'}$, we say that $\mathcal{I}$ *is less than or equal to* $\mathcal{I}'$ *w.r.t.* r, denoted by $\mathcal{I} \leq_r \mathcal{I}'$, if the following conditions hold for every individual a, every similarity relation symbol σ, and every concept name p :

1. $r(a^{\mathcal{I}}, a^{\mathcal{I}'})$
2. $\forall x, x', y \ \big[[\sigma^{\mathcal{I}}(x,y) \land r(x,x')] \to \exists y' \, [\sigma^{\mathcal{I}'}(x',y') \land r(y,y')]\big]$
3. $\forall x, x', y' \ \big[[\sigma^{\mathcal{I}'}(x',y') \land r(x,x')] \to \exists y \, [\sigma^{\mathcal{I}}(x,y) \land r(y,y')]\big]$
4. $\forall x, x' \ [r(x,x') \to (x \in p^{\mathcal{I}} \to x' \in p^{\mathcal{I}'})]$. $\lhd$

In Definition 9, the first three conditions state that r is a kind of bisimulation between the *frames* of $\mathcal{I}$ and $\mathcal{I}'$. Intuitively, $r(x,x')$ states that x has fewer positive properties than x'.

The following lemma is proved in [16].

Lemma 2. *Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{I}' = \langle \Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'} \rangle$ be Kripke structures, and $r \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}'}$ be a relation that satisfies conditions 2, 3, 4 of Definition 9. If $r(x,x')$ holds then, for every positive formula φ, $x \in \varphi^{\mathcal{I}}$ implies $x' \in \varphi^{\mathcal{I}'}$.* $\lhd$

Corollary 1. *Let $\mathcal{I}$ and $\mathcal{I}'$ be Kripke structures such that $\mathcal{I} \leq_r \mathcal{I}'$ for some r. Then $\mathcal{I} \leq \mathcal{I}'$.* $\lhd$

We are now ready to define the least SPDL model of a HSPDL database.

Definition 10. Let $\langle \mathcal{P}, \mathcal{A} \rangle$ be an HSPDL database. We say that a Kripke structure $\mathcal{I}$ is a *least* SPDL *model of* $\langle \mathcal{P}, \mathcal{A} \rangle$ if $\mathcal{I}$ is an SPDL model of $\langle \mathcal{P}, \mathcal{A} \rangle$ and for any other SPDL model $\mathcal{I}'$ of $\langle \mathcal{P}, \mathcal{A} \rangle$ we have that $\mathcal{I} \leq \mathcal{I}'$. $\lhd$

3.2. Constructing Least SPDL Models for HSPDL Databases

Now, we are ready to present an algorithm that constructs a finite least SPDL model for a given HSPDL database $\langle \mathcal{P}, \mathcal{A} \rangle$. During execution, the algorithm constructs the following data structures:

- Δ is a set of objects. We distinguish the subset Δ_0 of Δ that consists of all individuals (from INames).
- H is a mapping that maps every $x \in \Delta$ to a set of formulas, which are the properties that should hold for x. When the elements of Δ are treated as states, $H(x)$ denotes the contents of the state x.

- $Next$ is a mapping such that, for $x \in \Delta$ and $\langle\sigma\rangle\varphi \in H(x)$, we have $Next(x, \langle\sigma\rangle\varphi) \in \Delta$. The meaning of $Next(x, \langle\sigma\rangle\varphi) = y$ is that:
  - $\langle\sigma\rangle\varphi \in H(x)$ and $\varphi \in H(y)$,
  - the "requirement" $\langle\sigma\rangle\varphi$ is realized for x by going to y via a σ-transition.

We call the tuple $\langle\Delta, H, Next\rangle$ a *model graph*.
Using the above data structures, we define a Kripke structure $\mathcal{I}$ such that:

- $\Delta^{\mathcal{I}} = \Delta$,
- $a^{\mathcal{I}} = a$ for every $a \in \mathrm{INames}$,
- $p^{\mathcal{I}} = \{x \in \Delta \mid p \in H(x)\}$ for every $p \in \mathrm{CNames}$,
- $\sigma^{\mathcal{I}} = \{(a, b) \mid \sigma(a, b) \in \mathcal{A}\} \cup \{(x, y) \mid Next(x, \langle\sigma\rangle\varphi) = y$ for some $\varphi\}$ for every $\sigma \in \mathrm{SNames}$.

Definition 11. For $x, y \in \Delta$, we say that y is *reachable from* x if there exists a word $\sigma_1 \dots \sigma_k$ such that $(\sigma_1 \dots \sigma_k)^{\mathcal{I}}(x, y)$ holds. We say that y is *reachable from* Δ_0 if it is reachable from some $x \in \Delta_0$. $\triangleleft$

Definition 12. The *saturation* of a set Γ of formulas, denoted by $\mathsf{Sat}(\Gamma)$, is defined to be the smallest superset of Γ such that:

- $\top \in \mathsf{Sat}(\Gamma)$ and $\langle\sigma\rangle\top \in \mathsf{Sat}(\Gamma)$ for all $\sigma \in \mathrm{SNames}$,
- if $\varphi \wedge \psi \in \mathsf{Sat}(\Gamma)$ or $\langle\varphi?\rangle\psi \in \mathsf{Sat}(\Gamma)$ then $\varphi \in \mathsf{Sat}(\Gamma)$ and $\psi \in \mathsf{Sat}(\Gamma)$,
- if $\langle\alpha;\beta\rangle\varphi \in \mathsf{Sat}(\Gamma)$ then $\langle\alpha\rangle\langle\beta\rangle\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha;\beta]\varphi \in \mathsf{Sat}(\Gamma)$ then $[\alpha][\beta]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha \cup \beta]\varphi \in \mathsf{Sat}(\Gamma)$ then $[\alpha]\varphi \in \mathsf{Sat}(\Gamma)$ and $[\beta]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\alpha^*]\varphi \in \mathsf{Sat}(\Gamma)$ then $\varphi \in \mathsf{Sat}(\Gamma)$ and $[\alpha][\alpha^*]\varphi \in \mathsf{Sat}(\Gamma)$,
- if $[\varphi?]\psi \in \mathsf{Sat}(\Gamma)$ then $(\varphi \to \psi) \in \mathsf{Sat}(\Gamma)$. $\triangleleft$

Observe that $\mathsf{Sat}(\Gamma)$ is finite when Γ is finite. It can be shown that the size of $\mathsf{Sat}(\Gamma)$ is quadratic in the size of Γ (cf. Lemma 6.3 in [18]).

Definition 13. The *transfer of Γ through σ* is defined by:

$$\mathsf{Trans}(\Gamma, \sigma) \overset{\mathrm{def}}{=} \mathsf{Sat}(\{\varphi \mid [\sigma]\varphi \in \Gamma\}). \qquad \triangleleft$$

We use procedure $\mathsf{Find}(\Gamma)$ defined as:

if there exists $x \in \Delta \setminus \Delta_0$ with $H(x) = \Gamma$ then return x,
else add a new object x to Δ with $H(x) = \Gamma$ and return x.

Algorithm 1 shown in Figure 3 constructs a least SPDL model for an HSPDL database $\langle\mathcal{P}, \mathcal{A}\rangle$ as follows. At the beginning, Δ starts from $\Delta_0 = \mathrm{INames}$ with $H(x)$, for $x \in \Delta_0$, being the saturation of $\mathcal{P} \cup \{p \mid p(x) \in \mathcal{A}\}$. Then for each $x \in \Delta$ reachable from Δ_0 and for each formula $\varphi \in H(x)$ that does not hold for x, the algorithm makes a change to satisfy φ for x.

There are three relevant forms of φ:[8]

[8] The other possible forms of φ are dealt with by the saturation operator Sat.

Algorithm 1

Input: An HSPDL database $\langle \mathcal{P}, \mathcal{A} \rangle$.
Output: A least SPDL model $\mathcal{I}$ of $\langle \mathcal{P}, \mathcal{A} \rangle$.

```
1. set $\Delta_0 := \mathrm{INames}$, $\Delta := \Delta_0$, $\mathcal{P}' := \mathsf{Sat}(\mathcal{P})$
   for $x \in \Delta$, set $H(x) := \mathcal{P}' \cup \{p \mid p(x) \in \mathcal{A}\}$
2. for every $x \in \Delta$ reachable from $\Delta_0$ and for every formula $\varphi \in H(x)$
   (a) case $\varphi = \langle\sigma\rangle\psi$ : if $Next(x, \langle\sigma\rangle\psi)$ is not defined then
           $Next(x, \langle\sigma\rangle\psi) := \mathsf{Find}(\mathsf{Sat}(\{\psi\}) \cup \mathsf{Trans}(H(x), \sigma) \cup \mathcal{P}')$
   (b) case $\varphi = [\sigma]\psi$ :
          i. for every $y \in \Delta_0$ such that $\sigma^{\mathcal{I}}(x, y)$ holds and $\psi \notin H(y)$
                 $H(y) := H(y) \cup \mathsf{Sat}(\{\psi\})$
         ii. for every $y \in \Delta \setminus \Delta_0$ such that $\sigma^{\mathcal{I}}(x, y)$ holds and $\psi \notin H(y)$
               A. $y_* := \mathsf{Find}(H(y) \cup \mathsf{Sat}(\{\psi\}))$
               B. for every $\xi$ such that $Next(x, \langle\sigma\rangle\xi) = y$
                      $Next(x, \langle\sigma\rangle\xi) := y_*$
   (c) case $\varphi = (\psi \to \xi)$ : if $x \in \psi^{\mathcal{I}}$ and $Next(y, \langle\sigma\rangle\top)$ is defined for every $y$
       reachable from $x$ and every $\sigma \in \mathrm{SNames}$ then
          i. if $x \in \Delta_0$ then $H(x) := H(x) \cup \mathsf{Sat}(\{\xi\})$
         ii. else
               A. $x_* := \mathsf{Find}(H(x) \cup \mathsf{Sat}(\{\xi\}))$
               B. for every $y$, $\sigma$, $\zeta$ such that $Next(y, \langle\sigma\rangle\zeta) = x$
                      $Next(y, \langle\sigma\rangle\zeta) := x_*$
3. if some change occurred, go to Step 2
4. delete from $\Delta$ every $x$ unreachable from $\Delta_0$ and delete from $H$ and $Next$
   all elements related to such an $x$.
```

Fig. 3. Constructing a least SPDL model for an HSPDL database.

1. φ is of the form $\langle\sigma\rangle\psi$:
   To satisfy φ for x, we connect x via a σ-transition to an object $y \in \Delta \setminus \Delta_0$ with

   $$H(y) = \mathsf{Sat}(\{\psi\} \cup \{\xi \mid [\sigma]\xi \in H(x)\} \cup \mathcal{P})$$

   by setting $Next(x, \langle\sigma\rangle\psi) := y$.
2. φ is of the form $[\sigma]\psi$:
   We intend to add ψ to $H(y)$ for every y such that $\sigma^{\mathcal{I}}(x, y)$. We do this for the case when $y \in \Delta_0$. However, for $y \in \Delta \setminus \Delta_0$ modifying $H(y)$ has two drawbacks:
   - first, other objects connected to y will be affected (e.g., if p is added to $H(y)$ and $\sigma_2^{\mathcal{I}}(z, y)$ holds, then $\langle\sigma_2\rangle p$ becomes satisfied for z, while x and z may be independent)
   - second, modifying $H(y)$ may cause $H(y) = H(y')$ for some $y' \in \Delta \setminus \Delta_0$ different from y, which we try to avoid.

As a solution, instead of modifying $H(y)$ we replace σ-transitions (x, y) by σ-transitions (x, y_*), where y_* is the object such that

$$H(y_*) = H(y) \cup \mathsf{Sat}(\{\psi\}).$$

3. φ is of the form $\psi \to \xi$ (where ψ is a positive formula):
If ψ "must hold"[9] for x then we intend to add ξ to $H(x)$. We do this for the case $x \in \Delta_0$. However, when $x \in \Delta \setminus \Delta_0$, analogously to the case when φ is of the form $[\sigma]\zeta$, we do not modify $H(x)$, but replace transitions (y, x) by transitions (y, x_*), where x_* is the object such that

$$H(x_*) = H(x) \cup \mathsf{Sat}(\{\xi\}).$$

Example 2. Let $\mathcal{P} = \{p \to [\sigma^*]q,\ [\sigma^*]q \to p\}$ and $\mathcal{A} = \{p(a), s(a), \sigma(a, b)\}$. In Figure 4 we illustrate the construction of a least SPDL model of $\langle \mathcal{P}, \mathcal{A} \rangle$. $\lhd$

The proofs of the following lemma and theorem can be found in [16]. For the theorem we assume that the set of individuals (of INames) that do not occur in the ABox $\mathcal{A}$ is fixed.

Lemma 3. *Let $\mathcal{I}$ be the model constructed by Algorithm 1 for $\langle \mathcal{P}, \mathcal{A} \rangle$, and $\mathcal{I}'$ be an arbitrary* SPDL *model of $\langle \mathcal{P}, \mathcal{A} \rangle$. Let*

$$\begin{aligned} r = \{&(a, a^{\mathcal{I}'}) \mid a \text{ is an individual occurring in } \mathcal{A}\} \cup \\ &\{(x, x') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}'} \mid x \text{ is not an individual and } \mathcal{I}', x' \models H(x)\}. \end{aligned}$$

Then $\mathcal{I} \leq_r \mathcal{I}'$. $\lhd$

Theorem 1. *For an input* HSPDL *database $\langle \mathcal{P}, \mathcal{A} \rangle$ Algorithm 1 runs in polynomial time in the size of $\mathcal{A}$ and returns a least* SPDL *model $\mathcal{I}$ of $\langle \mathcal{P}, \mathcal{A} \rangle$ of a size polynomial in the size of $\mathcal{A}$.* $\lhd$

Remark 2. The above theorem is central for the querying machinery developed in this paper. According to Definitions 8 and 10, the least model $\mathcal{I}$ has the property that for every positive formula φ and for every individual a, we have that $\langle \mathcal{P}, \mathcal{A} \rangle \models_s \varphi(a)$ iff $a^{\mathcal{I}} \in \varphi^{\mathcal{I}}$. The model is then used to compute answers to queries. $\lhd$

The following corollary follows from the above theorem and Lemma 1.

Corollary 2. *The data complexity of* HSPDL *is in* PTIME. $\lhd$

[9] The statement "ψ must hold for x" intuitively means that "ψ follows from $H(x)$". As it can be seen later, a sufficient condition for the truth of this statement is that $x \in \psi^{\mathcal{I}}$ and $Next(y, \langle\sigma\rangle\top)$ is defined for every y reachable from x and every $\sigma \in \mathrm{SNames}$.

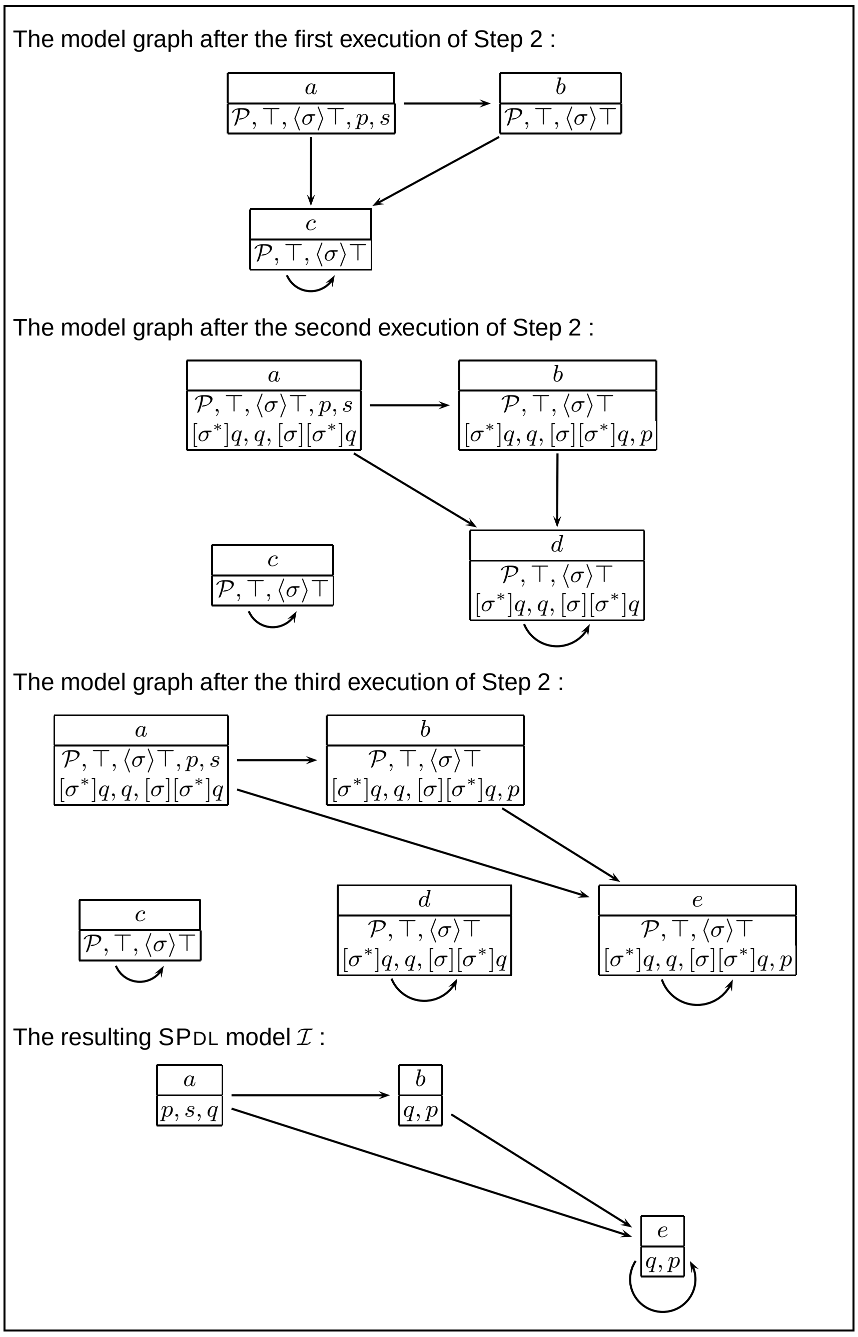

The model graph after the first execution of Step 2 :
a
$\mathcal{P}, \top, \langle\sigma\rangle\top, p, s$
b
$\mathcal{P}, \top, \langle\sigma\rangle\top$
c
$\mathcal{P}, \top, \langle\sigma\rangle\top$
The model graph after the second execution of Step 2 :
a
$\mathcal{P}, \top, \langle\sigma\rangle\top, p, s$
$[\sigma^*]q, q, [\sigma][\sigma^*]q$
b
$\mathcal{P}, \top, \langle\sigma\rangle\top$
$[\sigma^*]q, q, [\sigma][\sigma^*]q, p$
c
$\mathcal{P}, \top, \langle\sigma\rangle\top$
d
$\mathcal{P}, \top, \langle\sigma\rangle\top$
$[\sigma^*]q, q, [\sigma][\sigma^*]q$
The model graph after the third execution of Step 2 :
a
$\mathcal{P}, \top, \langle\sigma\rangle\top, p, s$
$[\sigma^*]q, q, [\sigma][\sigma^*]q$
b
$\mathcal{P}, \top, \langle\sigma\rangle\top$
$[\sigma^*]q, q, [\sigma][\sigma^*]q, p$
c
$\mathcal{P}, \top, \langle\sigma\rangle\top$
d
$\mathcal{P}, \top, \langle\sigma\rangle\top$
$[\sigma^*]q, q, [\sigma][\sigma^*]q$
e
$\mathcal{P}, \top, \langle\sigma\rangle\top$
$[\sigma^*]q, q, [\sigma][\sigma^*]q, p$
The resulting SPDL model $\mathcal{I}$:
a
p, s, q
b
q, p
e
q, p

Fig. 4. An illustration of the run of Algorithm 1 for $\mathcal{P} = \{p \to [\sigma^*]q,\ [\sigma^*]q \to p\}$ and $\mathcal{A} = \{p(a), s(a), \sigma(a, b)\}$. We have that $\Delta_0 = \{a, b\}$. In the shown model graphs, an edge from a node x to a node y means $Next(x, \langle\sigma\rangle\top) = y$. The edges in the resulting model $\mathcal{I}$ represent the similarity relation $\sigma^{\mathcal{I}}$.

3.3. Important Consequences of the Construction of Least Models for HSPDL

Some steps of Algorithm 1 add new objects to satisfy certain formulas. This is a new phenomenon, comparing to more traditional rule languages, where, e.g., existential quantification in heads of program clauses is forbidden. Such an addition of new objects sometimes occurs as a result of application of procedure $Find$, e.g., in Step 2a of Algorithm 1. On the other hand, this phenomenon seriously affects similarity relations. The following example illustrates the problem.

Example 3. Consider an ABox $\{p(a), \sigma(a,a)\}$ and a rule $p \rightarrow \langle\sigma\rangle q$. In such a case, during construction of a least model, Algorithm 1 adds to its universe two new objects, say b and c, for which $\sigma(a,b)$, $\sigma(a,c)$, $\sigma(b,c)$, $\sigma(c,c)$ and $q(b)$ additionally hold. Extending the domain by artificially created objects might be seen as a rather unexpected side-effect of the construction of a least model. The explanation is that new objects are sometimes added to satisfy certain rules. $\lhd$

The above example shows that rules do add new objects which are not grounded in the ABox of the database. Methodologically, such a situation is doubtful, as such artificially added objects are not as strongly justified as objects “observed” and directly described in terms of facts. In fact, these artificial objects are only possible explanations of rules, so have a rather weak status. We address this point in our layered architecture.

Another important issue is that the construction of a least model provided by Algorithm 1 may result in unexpected consequences due to identifying certain new objects. The following example illustrates this problem.

Example 4. Consider the rule $\sigma(a,x) \wedge \sigma(b,x) \rightarrow p(a,b)$, where σ is a similarity relation. In general $p(a,b)$ may not be a consequence of the rule. However, Algorithm 1 might have identified two objects, say c, d such that $\sigma(a,c)$ and $\sigma(b,d)$ hold. In such a case $p(a,b)$ would become deducible from the considered rule. $\lhd$

Observe that in the light of Theorem 1, Algorithm 1 constructs a least model for the input program. This means (see Remark 2) that the method we propose is correct. The above discussion applies to the case when queries of the highest level directly refer to similarity relations constructed in the middle layer.

Summing up, it is not safe to use similarity relations that have been changed by the HSPDL layer in higher-level queries. In the following sections we define a query language which allows one to ask only “safe” queries, free of the unwanted side-effects discussed above and guaranteeing the correctness of reasoning.

4. Combining HSPDL With Datalog

In the presence of ABoxes, a concept name can be viewed as a unary predicate, and a similarity relation symbol can be viewed as a binary predicate. In this section we extend our language HSPDL with external capabilities offered by database technologies and/or other software systems. The idea is quite general. However, in what follows we focus on Datalog as a possible instantiation of the idea. Therefore, in what follows we consider combination of HSPDL and Datalog and external data types.[10]

To solve problems discussed in Section 3.3, we introduce a special unary predicate $\overline{\mathrm{IName}}$ (treated as a concept name) with the semantics that, in every interpretation $\mathcal{I}$, $\overline{\mathrm{IName}}^{\mathcal{I}}$ consists of all objects which are not assigned to any individual from INames.

We use two basic types $\mathcal{O}$ and $\mathcal{D}$, where $\mathcal{O}$ is called the *individual type* (or *object type*) and $\mathcal{D}$ is called the *data type*. We assume that $\mathcal{D}$ is a fixed non-empty set, which may be the set of real numbers, the set of natural numbers, the set of strings, or a mixture of them. For simplicity we do not divide $\mathcal{D}$ into components.

An individual $a \in \mathrm{INames}$ has type $\mathcal{O}$, a concept name $p \in \mathrm{CNames} \cup \{\overline{\mathrm{IName}}\}$ has type $P(\mathcal{O})$ (the powerset type of $\mathcal{O}$), and a similarity relation symbol $\sigma \in \mathrm{SNames}$ has type $P(\mathcal{O} \times \mathcal{O})$. Apart from CNames and SNames, we use also a set OPreds of *ordinary predicates* and a set $\mathrm{ECPreds}$ of *external checkable predicates*. A k-ary predicate of OPreds has type $P(T_1 \times \ldots \times T_k)$, where each T_i is either $\mathcal{O}$ or $\mathcal{D}$. A k-ary predicate of $\mathrm{ECPreds}$ has type $P(\mathcal{D}^k)$. We assume that each predicate of $\mathrm{ECPreds}$ has a fixed interpretation which is checkable in the sense that, if p is a k-ary predicate of $\mathrm{ECPreds}$ and $d_1, \ldots, d_k$ are elements of $\mathcal{D}$, then the truth value of $p(d_1, \ldots, d_k)$ is fixed and computable. For example, when $\mathcal{D}$ is the type of real numbers, we may want to use the binary predicates $>$, $\geq$, $<$, $\leq$ on $\mathcal{D}$ with the usual semantics.

We assume that the sets INames, CNames, SNames, OPreds, $\mathrm{ECPreds}$ and $\mathcal{D}$ are pairwise disjoint and do not contain $\overline{\mathrm{IName}}$. Let $\mathrm{Preds} = \mathrm{CNames} \cup \mathrm{SNames} \cup \mathrm{OPreds} \cup \mathrm{ECPreds} \cup \{\overline{\mathrm{IName}}\}$. It is the set of all predicates of our language.

Definition 14. An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set of objects and $\cdot^{\mathcal{I}}$ is an interpretation function that maps each individual a to an element $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, maps $\overline{\mathrm{IName}}$ to $\overline{\mathrm{IName}}^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus \{a^{\mathcal{I}} \mid a \in \mathrm{INames}\}$, and maps each predicate $p \in \mathrm{CNames} \cup \mathrm{SNames} \cup \mathrm{OPreds}$ of type $P(T_1 \times \ldots \times T_k)$ to a subset $p^{\mathcal{I}}$ of $D_1 \times \ldots \times D_k$, where $D_i = \Delta^{\mathcal{I}}$ if $T_i = \mathcal{O}$, and $D_i = \mathcal{D}$ if $T_i = \mathcal{D}$, for $1 \leq i \leq k$. $\lhd$

An interpretation $\mathcal{I}$ can be treated as a Kripke structure by restricting $\cdot^{\mathcal{I}}$ to $\mathrm{INames} \cup \mathrm{CNames} \cup \mathrm{SNames} \cup \{\overline{\mathrm{IName}}\}$, especially when interpreting formulas of PDL. Given a formula φ of PDL built from concept names of $\mathrm{CNames} \cup \{\overline{\mathrm{IName}}\}$ and similarity relation symbols of SNames, the set $\varphi^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ is defined as usual. For $a \in \mathrm{INames}$, we write $\mathcal{I} \models \varphi(a)$ to denote that $a^{\mathcal{I}} \in \varphi^{\mathcal{I}}$.

[10] In the literature (e.g., [1]), Datalog programs consist of an extensional database (facts) and an intensional database (rules). In this paper we refer to the extensional part of the database as to its ABox.

Definition 15. A *term* is either an individual or an element of $\mathcal{D}$ (called a *data constant*) or a *variable* (of type $\mathcal{O}$ or $\mathcal{D}$). If p is a predicate of type $P(T_1 \times \ldots \times T_k)$, and for $1 \leq i \leq k$, t_i is a term of type T_i, then $p(t_1, \ldots, t_k)$ is an *atomic formula* (also called an *atom*). $\triangleleft$

From now on we use letters like x, y, z to denote variables, and letters like t to denote terms. We assume that the types of used predicates are given, and each used variable has a unique type $\mathcal{O}$ or $\mathcal{D}$, known from the context.

Definition 16. A *variable assignment* w.r.t. an interpretation $\mathcal{I}$ is a function that maps each variable of type $\mathcal{O}$ to an element of $\Delta^{\mathcal{I}}$ and maps each variable of type $\mathcal{D}$ to an element of $\mathcal{D}$.

The *value of a term* t w.r.t. a variable assignment ν is denoted by t^ν and defined as follows: if t is a variable then $t^\nu = \nu(t)$; if t is an individual then $t^\nu = t^{\mathcal{I}}$; if t is a data constant (i.e. $t \in \mathcal{D}$) then $t^\nu = t$.

Let $\mathcal{I}$ be an interpretation and ν be a variable assignment w.r.t. $\mathcal{I}$. We say that an atom $p(t_1, \ldots, t_k)$ is *satisfied* in $\mathcal{I}$ using ν, write $\mathcal{I}, \nu \models p(t_1, \ldots, t_k)$, if $(t_1^\nu, \ldots, t_k^\nu) \in p^{\mathcal{I}}$ for the case $p \notin \mathrm{ECPreds}$, and $p(t_1^\nu, \ldots, t_k^\nu)$ holds for the case $p \in \mathrm{ECPreds}$. A ground atom A (i.e. an atom without variables) is *satisfied* in $\mathcal{I}$, write $\mathcal{I} \models A$, if $\mathcal{I}, \nu \models A$ for any ν. $\triangleleft$

Definition 17. An *ABox* (in the extended language) is a finite set of ground atoms of predicates of $\mathrm{CNames} \cup \mathrm{SNames} \cup \mathrm{OPreds}$. An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ if all atoms of $\mathcal{A}$ are satisfied in $\mathcal{I}$. $\triangleleft$

We now define Datalog extended with external checkable predicates.

Definition 18.

– A *Datalog program clause* is a formula of the form

$$A_1 \wedge \ldots \wedge A_n \rightarrow B$$

where $n \geq 0$ and $A_1, \ldots, A_n, B$ are atomic formulas with the restriction that:
  - B is an atom of a predicate of $\mathrm{CNames} \cup \mathrm{SNames} \cup \mathrm{OPreds}$
  - every variable occurring in B occurs also in $A_1 \wedge \ldots \wedge A_n$
  - every variable occurring in the clause occurs, amongst others, in an atom of a predicate not belonging to $\mathrm{ECPreds}$.

  The last two restrictions are called the *range-restrictedness* condition. We call B the *head*, and $A_1 \wedge \ldots \wedge A_n$ the *body* of the clause. We omit the implication sign $\rightarrow$ when $n = 0$.

– A *Datalog program* is a finite set of Datalog program clauses.

– An interpretation $\mathcal{I}$ *validates* a Datalog program clause $A_1 \wedge \ldots \wedge A_n \rightarrow B$ if for every variable assignment ν, if $\mathcal{I}, \nu \models A_i$ for all $1 \leq i \leq n$ then $\mathcal{I}, \nu \models B$. An interpretation $\mathcal{I}$ is a *model* of a Datalog program $\mathcal{P}$ if it validates all the clauses of $\mathcal{P}$. $\triangleleft$

We now consider combination of HSPDL and Datalog. In the combined language, a database consists of an ABox $\mathcal{A}$ as the extensional part, and a mixed logic program $\mathcal{P}$ of HSPDL and Datalog as the intensional part. We will study the case when $\mathcal{P}$ consists of three layers, as discussed in Section 1.2 and illustrated in Figure 1. To be now more concrete, consider $\mathcal{P} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ with the following meaning:

- the lower layer $\mathcal{P}_1$ is a Datalog program intended for specifying the most basic predicates, basic concepts and similarity relations by using the perceptual data of agents and assertions stored in $\mathcal{A}$
- the middle layer $\mathcal{P}_2$ is an HSPDL logic program built on top of $\mathcal{P}_1$ and $\mathcal{A}$ for specifying advanced concepts by using the concepts and similarity relations specified in $\mathcal{P}_1$ and $\mathcal{A}$
- the upper layer $\mathcal{P}_3$ is a Datalog program built on top of $\mathcal{P}_2$, $\mathcal{P}_1$ and $\mathcal{A}$ for defining additional ordinary predicates and for completing definition of concepts and ordinary predicates.

The set $\mathit{InPreds}(\mathcal{P})$ (resp. $\mathit{OutPreds}(\mathcal{P})$) of *input predicates* (resp. *output predicates*) a Datalog program $\mathcal{P}$ is the set of all predicates occurring in the heads (resp. bodies) of program clauses of $\mathcal{P}$.

We define the set $\mathit{OutPreds}(\varphi)$ of *output predicates* of an HSPDL program clause φ recursively as follows:

$$\begin{aligned}
\mathit{OutPreds}(\top) &= \emptyset \\
\mathit{OutPreds}(p) &= \{p\} \\
\mathit{OutPreds}(\psi \to \xi) &= \mathit{OutPreds}(\xi) \\
\mathit{OutPreds}(\psi \land \xi) &= \mathit{OutPreds}(\psi) \cup \mathit{OutPreds}(\xi) \\
\mathit{OutPreds}(\langle\alpha\rangle\psi) &= \mathit{OutPreds}(\alpha) \cup \mathit{OutPreds}(\psi) \\
\mathit{OutPreds}([\alpha]\psi) &= \mathit{OutPreds}(\psi) \\
\mathit{OutPreds}(\sigma) &= \{\sigma\} \\
\mathit{OutPreds}(\alpha;\beta) &= \mathit{OutPreds}(\alpha) \cup \mathit{OutPreds}(\beta) \\
\mathit{OutPreds}(\psi?) &= \mathit{OutPreds}(\psi)
\end{aligned}$$

For example, $\mathit{OutPreds}([\sigma_1]p \to \langle\sigma_3\rangle(\langle\sigma_2\rangle q \to (r \land [\sigma_4]s))) = \{\sigma_3, r, s\}$.

The set $\mathit{OutPreds}(\mathcal{P})$ of *output predicates* of an HSPDL logic program $\mathcal{P}$ is defined to be $\bigcup_{\varphi\in\mathcal{P}} \mathit{OutPreds}(\varphi)$.

Definition 19. A *three-layered* HSPDL-*Datalog program* is a tuple $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$, where $\mathcal{P}_1$, $\mathcal{P}_3$ are Datalog programs and $\mathcal{P}_2$ is an HSPDL logic program using $\mathrm{CNames} \cup \{\overline{\mathrm{IName}}\}$ as the set of concept names, with the property that:

$$\mathit{OutPreds}(\mathcal{P}_2) \cap \mathit{InPreds}(\mathcal{P}_1) = \emptyset \tag{6}$$

$$\mathit{OutPreds}(\mathcal{P}_3) \subseteq \mathrm{CNames} \cup \mathrm{OPreds} \tag{7}$$

$$\mathit{OutPreds}(\mathcal{P}_3) \cap \mathit{InPreds}(\mathcal{P}_1) = \emptyset \tag{8}$$

$$\mathit{OutPreds}(\mathcal{P}_3) \cap \mathit{InPreds}(\mathcal{P}_2) = \emptyset \tag{9}$$

$$\mathit{OutPreds}(\mathcal{P}_2) \cap \mathrm{SNames} \cap \mathit{InPreds}(\mathcal{P}_3) = \emptyset \tag{10}$$
$$\overline{\mathrm{IName}} \notin \mathit{OutPreds}(\mathcal{P}_2) \tag{11}$$

Condition (6) states that the output predicates of $\mathcal{P}_2$ are not used as input predicates of $\mathcal{P}_1$. Conditions (7), (8) and (9) state that the output predicates of $\mathcal{P}_3$ can be only concept names or ordinary predicates which are not used as input predicates of $\mathcal{P}_1$ and $\mathcal{P}_2$. Roughly speaking, these conditions state that $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ is well-layered. Additionally, Conditions (10) and (6) guarantee that the similarity relations specified by $\mathcal{P}_2$ are not used as input predicates of $\mathcal{P}_1$ and $\mathcal{P}_3$. The reason is that an HSPDL logic program is intended to specify and minimize only (complex) concepts, but not to minimize similarity relations that are specified as side effects of existential modal operators. By (11) and the definition of Datalog program clauses, $\overline{\mathrm{IName}} \notin \mathit{OutPreds}(\mathcal{P}_1) \cup \mathit{OutPreds}(\mathcal{P}_2) \cup \mathit{OutPreds}(\mathcal{P}_3)$.

Definition 20. If $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ is a three-layered HSPDL-Datalog program and $\mathcal{A}$ is an ABox then the tuple $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$ is called a *three-layered* HSPDL-*Datalog database* (with $\mathcal{A}$ as a part of the bottom layer). $\triangleleft$

Observe that $\mathcal{A}$ is separated as data complexity takes its size as input. Also, facts from $\mathcal{A}$ are accessible to all layers.

In the following definition we accept the well-known Unique Names Assumption (see, e.g., [1]) for individuals from INames. Furthermore, the interpretation of similarity relations restricted to objects interpreting individuals from INames is computed and fixed by the first layer $\mathcal{P}_1$ using the minimal Herbrand model semantics.

Definition 21. Let $\mathcal{I}$ be an interpretation, $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ be a three-layered HSPDL-Datalog program, and $\mathcal{A}$ be an ABox. We say that $\mathcal{I}$ is a *model* of the three-layered HSPDL-Datalog database $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$ if:

- it is a model of $\mathcal{P}_1, \mathcal{P}_3, \mathcal{A}$ and is an SPDL model of $\mathcal{P}_2$ (when restricting $\cdot^{\mathcal{I}}$ to $\mathrm{INames} \cup \mathrm{CNames} \cup \mathrm{SNames} \cup \{\overline{\mathrm{IName}}\}$)
- for every $a \neq b \in \mathrm{INames}$, we have that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$
- for every interpretation $\mathcal{I}'$ satisfying the previous two conditions, for every $\sigma \in \mathrm{SNames}$ and $a, b \in \mathrm{INames}$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \sigma^{\mathcal{I}}$ then $(a^{\mathcal{I}'}, b^{\mathcal{I}'}) \in \sigma^{\mathcal{I}'}$. $\triangleleft$

Definition 22. A *query* to a three-layered HSPDL-Datalog database $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$ is an atomic formula A of a predicate of $\mathrm{CNames} \cup \mathrm{OPreds}$. A (correct) *answer* to such a query is a substitution $\theta = \{x_1/t_1, \ldots, x_k/t_k\}$ such that:

- $x_1, \ldots, x_k$ are all the different variables occurring in the query
- for $1 \leq i \leq k$, if x_i is a variable of type $\mathcal{O}$ then t_i is an individual, else (x_i is a variable of type $\mathcal{D}$ and) t_i is a data constant of $\mathcal{D}$
- the ground atom $A\theta$ is satisfied in every model of $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$. $\triangleleft$

Note that more complex queries can be expressed by adding a clause to the intensional part of the database. For example, if φ is a positive formula of PDL without predicates of $\mathit{OutPreds}(\mathcal{P}_3)$ and a is an individual, then to check whether $a^{\mathcal{I}} \in \varphi^{\mathcal{I}}$ in every model $\mathcal{I}$ of a database $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$ we can check whether $\{x/a\}$ is a correct answer to the query $p(x)$ w.r.t. the extended database $\langle \mathcal{P}_1, \mathcal{P}_2 \cup \{\varphi \to p\}, \mathcal{P}_3, \mathcal{A} \rangle$, where p is a new concept name. Similarly, if $\varphi = A_1 \land \ldots \land A_n$ is a formula such that $x_1, \ldots, x_k$ are all the variables occurring in φ and no predicate of φ belongs to $\mathit{OutPreds}(\mathcal{P}_2) \cap \mathrm{SNames}$, then answers to the query $p(x_1, \ldots, x_k)$ w.r.t. $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \cup \{\varphi \to p(x_1, \ldots, x_k)\}, \mathcal{A} \rangle$, where p is a new predicate of OPreds, are exactly the ground substitutions θ such that $A_1\theta$, ..., $A_n\theta$ are satisfied in every model of $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$.

Definition 23. By the *three-layered* HSPDL*-Datalog query language* we refer to the language of three-layered HSPDL-Datalog databases and their queries. The *data complexity* of this language is the complexity of the problem of finding all answers to a given query A w.r.t. a given three-layered HSPDL-Datalog database $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$, which is measured in the size of $\mathcal{A}$, when A, $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ are fixed. $\triangleleft$

The following theorem is the main result of this section, for which we assume that, given a k-ary predicate p of $\mathrm{ECPreds}$ and elements $d_1, \ldots, d_k$ of $\mathcal{D}$, checking whether $p(d_1, \ldots, d_k)$ holds can be done in polynomial time in the number of bits needed to represent d_1, ..., d_k.

Theorem 2. *The three-layered* HSPDL*-Datalog query language has* PTIME *data complexity.*

Proof. (sketch) Let a three-layered HSPDL-Datalog database $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$ be given. First, we compute the minimal Herbrand model $\mathcal{I}_1$ of $\mathcal{P}_1$ and $\mathcal{A}$. Treating it as an ABox, we next compute a least model $\mathcal{I}_2$ for $\langle \mathcal{P}_2, \mathcal{I}_1 \rangle$ using Algorithm 1 with the following modification of procedure $\mathsf{Find}(\Gamma)$:

if there exists $x \in \Delta \setminus \Delta_0$ with $H(x) = \Gamma \cup \{\overline{\mathrm{IName}}\}$ then return x,
else add a new object x to Δ with $H(x) = \Gamma \cup \{\overline{\mathrm{IName}}\}$ and return x.

Treating $\mathcal{I}_2$ as an ABox, we now compute the minimal Herbrand model $\mathcal{I}_3$ of $\mathcal{P}_3$ and $\mathcal{I}_2$. It can be shown that, for any query A, a ground substitution θ is an answer to A w.r.t. the database $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$ iff $A\theta$ is satisfied in the interpretation corresponding to $\mathcal{I}_3$. By Theorem 1 and the fact that the data complexity of Datalog is in PTIME, the above computation runs in time polynomial w.r.t. the size of $\mathcal{A}$. $\triangleleft$

5. Example

5.1. The Scenario

Consider safety of UGVs' movement on a specific surface (UGV is an acronym for Unmanned Ground Vehicle). For simplicity, we consider two UGVs, denoted by UGV$_1$ and UGV$_2$, operating on the same road segment and exchanging their knowledge, as well as assume that slipperiness of the road and speed of the UGV are the only factors that affect their safety.

Assume that the following equipment and data is available:

– there is an external sensor measuring the slipperiness of the road, evaluated by a real number in the range $[1, 10]$
– each UGV is equipped with sensors detecting its speed, measured in $\frac{km}{h}$, being a real number in $[0, 20]$
– there is a database of facts about situations that led or did not lead to accidents caused by the UGVs.

The main objects are situations, which form the type $\mathcal{O}$. In this example, the type $\mathcal{D}$ consists of real numbers and we use the following predicates, where $i \in \{1, 2\}$:[11]

– ordinary predicates spd_i (speed), slp (slipperiness), $dec\text{-}spd_i$ (decrease speed); the intuitive meaning of these predicates is:
  • $spd_i(x, y)$ holds when the speed of UGV$_i$ in situation x is y
  • $slp(x, y)$ holds when the slipperiness of the considered road segment in situation x is y
  • $dec\text{-}spd_i(x, y)$ holds when the speed of UGV$_i$ in situation x should be decreased by y $\frac{km}{h}$.
– external checkable predicates $>$, $<$, $\leq$ and sim, sim_i, where $>$, $<$, $\leq$ have the standard meaning, as in the arithmetics of reals and

$$sim_i(x_1, x_2) \overset{\text{def}}{\equiv} \frac{abs(x_1 - x_2)}{max(x_1, x_2)} \leq \epsilon_i,$$

  where ϵ_i reflects the accuracy of speed measurements of UGV$_i$; we assume here that $\epsilon_1 \overset{\text{def}}{=} 0.12$ and $\epsilon_2 \overset{\text{def}}{=} 0.09$; for sim (non-indexed) we use $\epsilon \overset{\text{def}}{=} 0.18$
– concept names $h\text{-}spd_i$ (high speed), $h\text{-}slp$ (highly slippery), $unsafe$, $unsafe_i$, $h\text{-}unsafe_i$ (highly unsafe), $accident$
– similarity relation symbols σ_i, where the intended meaning of $\sigma_i(x_1, x_2)$ is that situations x_1 and x_2 are similar w.r.t. UGV$_i$
– auxiliary similarity relation symbol ϱ, used for expressing that one situation is “safer” then the other w.r.t. both speed and slipperiness.

5.2. Exemplary Rules

Consider the following layers of the system, where we assume that $i \in \{1, 2\}$.

[11] The index i indicates subjective knowledge of UGV$_i$, while the lack of index indicates that the respective concepts are related to the external or fused knowledge.

Rules of the Lower Layer

$$spd_i(x,y) \land y > 15 \to h\text{-}spd_i(x) \tag{12}$$

$$slp(x,y) \land y > 7 \to h\text{-}slp(x) \tag{13}$$

$$\big(spd_i(x_1,y_1) \land slp(x_1,z_1) \land spd_i(x_2,y_2) \land slp(x_2,z_2) \land sim_i(y_1,y_2) \land sim(z_1,z_2)\big) \to \sigma_i(x_1,x_2) \tag{14}$$

$$\big(spd_i(x_1,y_1) \land slp(x_1,z_1) \land spd_i(x_2,y_2) \land slp(x_2,z_2) \land y_1 < y_2 \land z_1 \leq z_2\big) \to \varrho_i(x_1,x_2) \tag{15}$$

$$\big(spd_i(x_1,y_1) \land slp(x_1,z_1) \land spd_i(x_2,y_2) \land slp(x_2,z_2) \land y_1 \leq y_2 \land z_1 < z_2\big) \to \varrho_i(x_1,x_2) \tag{16}$$

The meaning of these rules is:

(12): UGV's speed greater than $15\frac{km}{h}$ is considered high
(13): road slipperiness greater than 7 is considered high
(14): two situations are similar w.r.t. σ_i when speeds and slipperiness in these situations are similar (w.r.t. sim_i)
(15) and (16): $\varrho_i(x_1,x_2)$ holds when situation x_1 is "safer" than x_2 from the point of view of UGV$_i$.

Rules of the Middle Layer

$$h\text{-}spd_i \land h\text{-}slp \to unsafe_i \tag{17}$$

$$unsafe_i \to [\varrho_i]unsafe_i \tag{18}$$

$$[\varrho_i](\overline{\mathrm{IName}} \lor \langle\sigma_i\rangle accident) \to unsafe_i \tag{19}$$

$$[\varrho_i](\overline{\mathrm{IName}} \lor accident) \to h\text{-}unsafe_i \tag{20}$$

$$[\varrho_1 \cup \varrho_2](\overline{\mathrm{IName}} \lor (unsafe_1 \land h\text{-}unsafe_2)) \to unsafe \tag{21}$$

$$[\varrho_1 \cup \varrho_2](\overline{\mathrm{IName}} \lor (h\text{-}unsafe_1 \land unsafe_2)) \to unsafe \tag{22}$$

Note that a formula of the form $(\overline{\mathrm{IName}} \lor \varphi)$ represents the set of objects x such that if x is assigned to some individual of INames then x satisfies the property φ. Intuitively, this means that either the situation x is not explicitly presented in the database[12] or it satisfies the property φ.

The meaning of the above rules is:

(17): the situation is unsafe whenever both the speed and the slipperiness are high
(18): if situation s is unsafe then also situations with the speed and slipperiness greater than or equal to those of s, are unsafe
(19): if, for a given situation s, every situation that is explicitly presented in the database and more dangerous than s (w.r.t. ϱ_i) is similar (w.r.t. σ_i) to a situation in which there was an accident, then conclude that s is also unsafe for UGV$_i$

[12] That is, x is added by Algorithm 1.

(20): if, for a given situation s, in every situation that is explicitly presented in the database and more dangerous than s (w.r.t. ϱ_i) there was an accident, then conclude that s is a highly unsafe situation for UGV$_i$

(21) and (22): specify unsafeness by fusing similarity relations of both UGVs.

Rules of the Upper Layer

$$unsafe_i(x) \land h\text{-}slp(x) \rightarrow dec\text{-}spd_i(x, 2) \tag{23}$$

$$h\text{-}unsafe_i(x) \land h\text{-}slp(x) \rightarrow dec\text{-}spd_i(x, 5) \tag{24}$$

$$unsafe(x) \rightarrow dec\text{-}spd_i(x, 3) \tag{25}$$

The meaning of these rules is:

(23): if x is an unsafe situation for UGV$_i$ in which the road is highly slippery, then UGV$_i$ should decrease its speed by $2\frac{km}{h}$

(24): if x is a highly unsafe situation for UGV$_i$, in which the road is highly slippery, then UGV$_i$ should decrease its speed by $5\frac{km}{h}$

(25): if x is an unsafe situation, then each UGV should decrease its speed by $3\frac{km}{h}$.

Let

- $\mathcal{P}_1$ be the Datalog program consisting of the clauses (12) – (16)
- $\mathcal{P}_2$ be the HSPDL program consisting of the clauses (17) – (22)
- $\mathcal{P}_3$ be the Datalog program consisting of the clauses (23) – (25).

Thus, $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ is a three-layered HSPDL-Datalog program.

Exemplary Facts of the Lower Layer

Let $\mathcal{A}$ be the exemplary ABox consisting of facts presented below:

situation	slp	spd_1	spd_2	$accident$
s_0	10	20	20	yes
s_1	9	19	17	yes
s_2	8	17	18	yes
s_3	7	16	17	yes
s_4	7	16	16	(no)
s_5	6	12	17	(no)
s_6	4	18	15	(no)
s_7	6	16	16	?

This ABox contains only four facts of predicate $accident$, with argument s_0, s_1, s_2 or s_3. In the situations s_4, s_5, s_6, there were no accidents. The situation s_7 is the current situation, for which we want to consider safeness of the UGVs. With respect to the three-layered HSPDL-Datalog database $\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{A} \rangle$, the query $dec\text{-}spd_1(s_7, x)$ returns answer $x = 2$.

6. Conclusions

In the paper we have addressed the problem of fusing possibly approximate knowledge from distributed sources. To express fusion rules we have used the Horn fragment of serial propositional dynamic logic combined with Datalog-based deductive databases machinery. As a framework for such a combination we have proposed a three-layered architecture. This allowed us to provide a pragmatic framework for knowledge fusion that can be used in many application areas. We have demonstrated the use of our approach on an example.

The paper can also be considered as an advanced case-study of embedding expressive propositional logics into database environments. Similar methodology can be used for many other logics designed as specification and computational tools for advanced software and robotics systems.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Pub. Co., 1996.
2. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *Description Logic Handbook*. Cambridge University Press, 2002.
3. T.H. Cormen, E.H. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.
4. S.P. Demri and E.S. Orłowska. *Incomplete Information: Structure, Inference, Complexity*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Heidelberg, 2002.
5. P. Doherty, B. Dunin-Kęplicz, and A. Szałas. Dynamics of approximate information fusion. In M. Kryszkiewicz, J. Peters, H. Rybinski, and A. Skowron, editors, *Proc. RSEISP 2007, Rough Sets and Emerging Intelligent Systems Paradigms*, number 4585 in LNAI, pages 668–677. Springer-Verlag, 2007.
6. P. Doherty, W. Łukaszewicz, A. Skowron, and A. Szałas. *Knowledge Representation Techniques. A Rough Set Approach*, volume 202 of *Studies in Fuziness and Soft Computing*. Springer-Verlag, 2006.
7. P. Doherty, W. Łukaszewicz, and A. Szałas. Communication between agents with heterogeneous perceptual capabilities. *Journal of Information Fusion*, 8(1):56–69, 2007.
8. P. Doherty and A. Szałas. On the correspondence between approximations and similarity. In S. Tsumoto, R. Slowinski, J. Komorowski, and J.W. Grzymala-Busse, editors, *Proc. of 4th International Conf. on Rough Sets and Current Trends in Computing, RSCTC'2004*, volume 3066 of *LNAI*, pages 143–152. Springer-Verlag, 2004.
9. D. Dubois, J. Lang, and H. Prade. Fuzzy sets in approximate reasoning, part 2: logical approaches. *Fuzzy Sets and Systems*, 40(1):203–244, 1991.
10. D. Dubois and H. Prade. Fuzzy sets in approximate reasoning, part 1: inference with possibility distributions. *Fuzzy Sets and Systems*, 40(1):143–202, 1991.
11. D. Dubois and H. Prade. *Fuzzy Sets and Systems*. Fuzzy Logic CDROM Library. Academic Press, 1996.
12. V. Dubois and M. Quafafou. Concept learning with approximation: Rough version spaces. In J.J Alpigini, J.F. Peters, A. Skowron, and N. Zhong, editors, *Rough Sets and Current Trends in Computing: Proc. of the Third International Conf., RSCTC 2002*, number 2475 in LNAI, pages 239–246. Springer-Verlag, 2002.

13. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. Fusing approximate knowledge from distributed sources. In G.A. Papadopoulos and C. Badica, editors, *Proc. IDC'09, 3rd International Symposium on Intelligent Distributed Computing*, volume 237 of *Studies in Computational Intelligence*, pages 75–86. Springer, 2009.
14. B. Dunin-Kęplicz and A. Szałas. Towards approximate BGI systems. In H-D. Burkhard, G. Lindeman, L. Varga, and R. Verbrugge, editors, *Proc. CEEMAS 2007, 5th International Central and Eastern European Conf. on Multi-Agent Systems*, number 4696 in LNAI, pages 277–287. Springer-Verlag, 2007.
15. B. Dunin-Kęplicz. An architecture with multiple meta-levels for the development of correct programs. In *Proc. of Fourth International Workshop on Meta Programming in Logic*, number 883 in LNCS, pages 293–310, 1995.
16. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. Tractable approximate knowledge fusion using the Horn fragment of serial propositional dynamic logic. *Int. J. Approx. Reasoning*, 51(3):346–362, 2010.
17. Y. Gdalyahu and D. Weinshall. Measures for silhouettes resemblance and representative silhouettes of curved objects. In Bernard F. Buxton and Roberto Cipolla, editors, *Proceedings of the 4th European Conference on Computer Vision ECCV'96, Volume II*, volume 1065 of *Lecture Notes in Computer Science*, pages 365–375. Springer, 1996.
18. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
19. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In L.P. Kaelbling and A. Saffiotti, editors, *Proc. of IJCAI-05*, pages 466–471. Professional Book Center, 2005.
20. R. Kruse, E. Schwecke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge Based Systems. Numerical Methods*. Springer-Verlag, 1991.
21. J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White. Revisions and extensions to the JDL data fusion model II. In P. Svensson and J. Schubert, editors, *Proc. of the 7th Int. Conf. on Information Fusion*, 2004.
22. J. Maluszyński, A. Szałas, and A. Vitória. Paraconsistent logic programs with four-valued rough sets. In C.-C. Chan, J. Grzymala-Busse, and W. Ziarko, editors, *Proc. of 6th International Conf. on Rough Sets and Current Trends in Computing (RSCTC 2008)*, volume 5306 of *LNAI*, pages 41–51, 2008.
23. J. McCarthy. Approximate objects and approximate theories. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proc. 7th International Conf. on Principles of Knowledge Representation and Reasoning, KR2000*, pages 519–526, 2000.
24. L.A. Nguyen. On the deterministic Horn fragment of test-free PDL. In I. Hodkinson and Y. Venema, editors, *Advances in Modal Logic - Volume 6*, pages 373–392. King's College Publications, 2006.
25. L.A. Nguyen. Weakening Horn knowledge bases in regular description logics to have PTIME data complexity. In S. Ghilardi, U. Sattler, V. Sofronie-Stokkermans, and A. Tiwari, editors, *Proc. of Automated Deduction: Decidability, Complexity, Tractability ADDCT'07*, pages 32–47, 2007.
26. L.A. Nguyen. Constructing finite least Kripke models for positive logic programs in serial regular grammar logics. *Logic Journal of the IGPL*, 16(2):175–193, 2008.
27. Z. Pawlak. *Rough Sets. Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, 1991.
28. H. Prade. A quantitative approach to approximate reasoning in rule-based expert systems. In L. Bolc and M.J. Coombs, editors, *Expert System Applications*, pages 199–256. Springer-Verlag, 1988.
29. A. Steinberg and C. Bowman. Revisions to the JDL data fusion model. In D. Hall and J. Llinas, editors, *Handbook of Multisensor Data Fusion*. CRC Press LLC, 2001.

30. J. van Benthem. Correspondence theory. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2, pages 167–247. D. Reidel Pub. Co., 1984.
31. A. Vitória, J. Maluszyński, and A. Szałas. Modeling and reasoning in paraconsistent rough sets. *Fundamenta Informaticae*, 97(4):405–438, 2009.
32. X. Wang, B. De Baets, and E. Kerre. A comparative study of similarity measures. *Fuzzy Sets and Systems*, 73(2):259–268, 1995.
33. F. White. A model for data fusion. In *Proc. of 1st National Symposium for Sensor Fusion*, volume 2, 1988.
34. Yang Xiang. Distributed multi-agent probabilistic reasoning with bayesian networks. In Zbigniew W. Ras and Maria Zemankova, editors, *ISMIS*, volume 869 of *Lecture Notes in Computer Science*, pages 285–294. Springer, 1994.
35. Yang Xiang. Semantics of multiply selectioned bayesian networks for cooperative multi-agent distributed interpretation. In Gordon I. McCalla, editor, *Canadian Conference on AI*, volume 1081 of *Lecture Notes in Computer Science*, pages 213–226. Springer, 1996.
36. Yang Xiang and Victor R. Lesser. On the role of multiply sectioned bayesian networks to cooperative multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 33(4):489–501, 2003.
37. Y.Y. Yao and T.Y. Lin. Generalization of rough sets using modal logic. *Intelligent Automation and Soft Computing, An International Journal*, 2(2):103–120, 1996.

Barbara Dunin-Kęplicz is a Professor of computer science at the Institute of Informatics of Warsaw University and at the Institute of Computer Science of the Polish Academy of Sciences. She obtained her PhD in 1990 on computational linguistics from the Jagiellonian University, and in 2004 she was awarded her habilitation on formal methods in multi-agent systems from the Polish Academy of Sciences.

She is a recognized expert in multi-agent systems and one of the pioneers in the area of modeling BDI systems.

Linh Anh Nguyen is an assistant professor of computer science at the Institute of Informatics of Warsaw University. He obtained a PhD degree in 2000 and a hablitation degree in 2009 on modal logics from Warsaw University. He has published a considerable number of papers on modal logic programming, Horn fragments of modal logics, and tableau-based automated reasoning for modal and description logics.

Andrzej Szałas is a Professor of computer science at the Institute of Informatics of Warsaw University and at the Department of Computer and Information Science of the Linköping University. He obtained his degrees in computer science from Warsaw University: PhD in 1984 and habilitation in 1991. In 1999, he obtained the scientific title of professor from the President of Poland.

He is an expert in logics in computer science and artificial intelligence.

Received: February 9, 2010; Accepted: May 3, 2010.

UDC 004.5:62, DOI:10.2298/CSIS100119014O

Emergent Properties for Data Distribution in a Cognitive MAS

Andrei Olaru, Cristian Gratie, and Adina Magda Florea

University Politehnica of Bucharest,
Splaiul Independentei 313,
060042 Bucharest, Romania
cs@andreiolaru.ro, cgratie@yahoo.com, adina@cs.pub.ro

Abstract. Emergence is a key element in the research of multi-agent systems. Emergent properties provide higher level features to a system formed of simpler individuals. So far, emergence has been studied mostly through systems formed of reactive agents – that are easier to design and implement. As computational power increases and even small devices become more capable, cognitive agents become a promising solution for more complex problems. This paper presents a multi-agent system that uses cognitive agents and that is designed to manifest emergent properties: a data storage system that assures distribution and replication of data as emergent properties, without the need for these features to be controlled in a centralized way.

Keywords: multi-agent system, cognitive agent, emergence, self-organization.

1. Introduction

Although emergence and self-organization have been the subject of research for a long time, it is only relatively recently that they have come to attention in the field of multi-agent systems (MAS). Its role in MAS is crucial, as it is a concept that fits perfectly with the agent-oriented paradigm and leads to important advantages.

Several definitions of emergence exist [4] but none is yet generally accepted. Many times emergence is defined by its effect – the spontaneous formation of patterns in the structure or behaviour of certain systems.

In the context of a large number of agents, forming a complex system, emergence offers the possibility of obtaining a higher level function or property by using agents with lower level implementations. The use of emergence and self-organization allows for a considerable reduction in the complexity needed for the individual agents, therefore they can run on simpler and smaller devices. Moreover, self-organization leads to properties like robustness, flexibility, adaptivity and scalability.

Many recent papers on emergence and multi-agent systems deal with emergent properties and behaviour in systems formed of a large number of reactive agents. Reactive agents are used because they are simple and need only very limited computational capacity, but emergent functions may be more complex and the structure is robust [3, 20, 38].

Nowadays, the capabilities of even very basic computing devices have considerably increased, allowing for a much more complex internal structure of agents – the possibility to hold reasonable amounts of data and to have a more nuanced behaviour. Cognitive agents have knowledge about the surrounding environment, have goals they desire to fulfill, make plans and take action in order to fulfill them. There is important research on emergence in cognitive agent systems, but there is still much room for improvement.

One of the domains of application for agent-based self-organizing systems is Ambient Intelligence (or AmI) [11]. Software agents (particularly cognitive agents) are considered as a very adequate paradigm for AmI [28], due to the properties of agents: autonomy, proactivity, reactivity, adaptivity and reasoning. A system for AmI must deal with a great number of interconnected devices (sensors, actuators, mobile phones, laptops, desktop computers, "smart" appliances), and needs to assist the user with daily activities. Most times assistance is performed by providing the user with interesting information. In the same time, the system must be robust and resilient to failure, while being decentralized, so an approach based on self-organization fits the requirements [6, 16, 17].

The purpose of this paper is to study emergent properties in a multi-agent system formed of cognitive agents, that is, study how such a system should be designed in order to obtain global, emergent, properties. The long-term goal of this research is to apply the results to the design of a middleware for AmI that facilitates the distribution of information through a network of devices, in a self-organized manner.

A system of cognitive agents used for the storage of data has been designed and implemented. While treating, at the time, a very specific case, this prototype makes a good framework for the study of emergent properties. It was designed with the purpose of manifesting emergent properties like uniform distribution and availability of data. Individual agents were given local goals that do not directly imply that the data is uniformly distributed or that data is not completely lost. No agent has the capability to have a global image on the system or even on a significant part of it. Knowledge of the agents is limited to beliefs about their immediate or very close neighbours. Yet the system as a whole keeps data well replicated and distributed.

This work presents the design of a system that satisfies the requirements above, using techniques of self-organization and emergent properties. The presented experiments were performed not necessarily to show how the system performs under heavy loads or if it is resilient to failure (although we expect it to), but to show a kind of global behaviour that can be obtained by local interaction, in the context of using cognitive agents.

A practical application to the studied system would be, for instance, a network of sensors in a building. The sensors have low computational capabilities so their behavior must be simple and use little memory. They sense events in the environment, like temperature or the presence of people. The system is completely decentralized, but information must be available for retrieval from any of the sensing devices. Therefore, sensors aggregate their sensor information into larger chunks of data and ”insert” it into the system, which, by using only local information and communication between neighbour sensors, distribute the data and make it uniformly available throughout the building. This is only an example. The system is generic and was built to fit a greater number of possible applications.

The paper is organized as follows. Section 2 is dedicated to related work in the field of emergence and multi-agent systems. Section 3 presents the main topic of this paper: a cognitive multi-agent system designed to manifest emergent properties. Section 4 describes the results obtained from experiments. The last section is dedicated to conclusions.

2. Related Work

What we know is that emergence appears in the context of complex systems [1] – systems composed of a large number of interacting individual entities. Emergence needs two levels of perspective: the inferior, or micro level of the individual entities and the superior, or macro level of the whole system. A simple definition is that ”emergence is the concept of some new phenomenon arising in a system that wasn’t in the system’s specification to start with” [36]. A more elaborated definition is that ”a system exhibits emergence when there are coherent emergents at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergents are novel with respect to the individual parts of the system” [8]. An ”emergent” is a notion that can represent a property, a structure or a behaviour that results from emergence.

The essence of emergence is not actually the novelty or the unexpectedness of the emergent – as these will fade at later experiments although the emergents will stay the same – but the difference between the description of the individual and the description of the emergent, from the point of view of an observer [36, 29]. If the minimal description of the individual is taken, it cannot be used to describe the emergents resulting from the system, therefore the emergent is considered as novel and, potentially, unexpected.

There are a few important features of emergence [8, 13, 18]:

– emergence occurs out of the interactions between the parts of a complex system;
– emergence is defined in relation with two levels – it is manifested at the higher level, arising from interactions at the lower level;
– the representation of the emergents cannot be reduced to the specification of the individuals;
– emergents only arise after a certain time in which the system has evolved;
– once they occurred, emergents will maintain identity over time;
– emergents arise without any centralized or exterior control;

– the emergent phenomenon is robust and flexible, i.e. it is not influenced by damage on the system (even if the emergent is temporarily not observable, it will arise again as the system evolves).

In the field of multi-agents systems, most examples that demonstrate emergent properties use reactive agents [35]. This is because they are easier to implement and control, and they are suitable for small devices with low computational power. But, more than anything, systems of reactive agents are easier to predict and to design so that they will manifest self-organization or other emergent properties. Notable examples of emergents in reactive agent systems relate to the formation of a certain geometrical or geometry-related structure or behaviour. Self-organization is reached by mechanisms that are usually inspired by nature and the behaviour of animal societies (generally insects) [22].

For example, ”smart dust” uses attraction and repulsion forces and simple leader election algorithms to dispose individuals in a circular shape, or in a lobed shape, that is resilient to structural damage [3, 21, 38]; ”spider” agents roam through the pixels of an image and use stigmergy to mark the different areas of the image [5]; local behaviour can be used for the gathering of resources in a single area [30], for the foraging of food or the transportation of loads [37]; in a very interesting example of dynamic self-organized behaviour, agents establish, by means of emergent coordination, traffic directions through narrow corridors for the transportation of resources between two areas [25].

Although reactive agent systems may be very useful, there are many advantages that a cognitive agent has over a reactive agent. First, it is proactive. Even if there are no signals, perceptions or stimuli from the environment, a cognitive agent may act by itself, taking action according to its objectives. Second, a cognitive agent is aware of its situation and may reason about it. It is aware of what it is supposed to fulfill as final goal and is capable of making plans and taking action towards the realization of its goal. The cognitive agent can use its experience and information about the environment and the consequences of past actions to develop a better plan every time a similar situation occurs.

Ricci et al. support the concept of *cognitive stigmergy* [32], that is inspired from natural, reactive, self-organization (stigmergy, introduced by Grassé [14]) but is using cognitive agents and a more complex environment. The environment is composed of *artifacts* that agents are using and in relation to which they make annotations. The usage of shared resources in the environment by one agent driven by local goals influences the other agents that have access to the resource, leading to feedback and eventually to organization. The research shows encouraging results in applying the concepts from self-organizing reactive architectures to cognitive agent systems.

Other studies of emergent behaviour in cognitive agents exist. Emergence of norms in a social game is possible through *social learning* [15, 34]. Gleizes et al show that local cooperative behaviour of cognitive agents can lead, by means of emergence, to better system-wide results [12]. The study is continued, using the AMAS (adaptive multi-agent system) technology as a framework that facilitates emergent functions based on cooperation [7].

However, the use of cognitive agents in self-organizing systems is still very reduced. While, in the present, numerous real-life applications of self-organization exist [27], they are based on reactive behaviour of the individual entities. It is true that these systems are easier to verify and control, and it is also true that even in reactive agent systems it is hard to find a good balance between explicitly designed behaviour and implicit, emergent behaviour [26], but the features of reactive systems are limited by the absence of reasoning in agents.

Besides taking inspiration from biological systems, there is also another approach to the design of self-organization, that is used mostly in wireless sensor networks. WSNs use simple individual units and, more importantly, need to consume little power. This approach is also many times validated formally, not only by simulation. It consists of different algorithms used for routing, resource allocation, structure formation, synchronization, power conservation and resilience [10, 24]. Bernadas et al. demonstrate communication services based on this type of self-organizing algorithms, with individual agents having a more complex internal behaviour and manifesting features like proactivity and a certain amount of reasoning [2]. A model for a self-organizing system for communication systems is proposed by Marinescu et al., using cognitive agents that have local goals and specific available actions. The model is verified formally and relevant results are shown [23]. However, these models address particular concerns in the functionality of a system and are not very flexible. It is a considerable problem to build a model that will address more than one concern at a time, and in a generic manner [23, 24].

It is worth mentioning in this section the related work on holonic systems. Holonic systems are based on entities that are, at the same time, wholes in themselves and parts of larger wholes [19]. Self-organization is also present in holonic multi-agent systems. Moreover, in holonic systems organization may be present on multiple levels [33]. However, there is little relation between this work and holonic systems. At present, our goal is to study emergence in a system that works on a single level, but a closer look to holonic systems will be considered for future work.

In this paper we present a multi-agent system that uses cognitive agents to obtain emergent properties. In the design of the system we used techniques from self-organizing reactive agent systems, but considering the cognitive features of the agents. This resembles the work on cognitive stigmergy [32], but uses direct interaction instead of stigmergic mechanisms. Different from other works on emergence in cognitive agent systems, our work, while trying to remain generic, is focused on the exchange and distribution of information through a multi-agent system. Our goal is not a function that needs to be calculated, nor a certain well-defined result that must be obtained, but a behaviour that needs to be achieved. And this by means of cognitive agents, because of the additional features they offer, which will not only favour a more adaptive behaviour and the possibility to work at the semantic level, but will also reduce the quantity of needed communication by having knowledge about and reasoning on their environment.

3. A Cognitive Multi-Agent System for the Distribution of Data

An application has been developed and experiments have been performed in order to demonstrate how emergents may arise in a system formed of cognitive agents. The system that was designed models a network for the storage, replication and distribution of data. The purpose is to store pieces of data inserted into the system and distribute them, as well as provide the user with the data that is requested.

The requirement that led the design of the system was to use local interaction and local knowledge about the system in order to obtain global emergent properties. The implementation was to be generic, without referring to a certain domain of application, this way establishing a more general methodology for building a system with emergent properties.

The presented experiments were performed with a set of agents placed on a rectangular grid. Each agent can communicate only with its 8 neighbours. Each agent has a limited capacity for the storage of data. The agents must always have some capacity left, ready for data that might be injected from the environment. However, agents must try to store as much data as possible, in order to not waste the capacity of the system. The information stored in the system (the information that is useful to the users) is represented as generic pieces of data, or, in short, *Data*.

The desired emergent properties of the system are the replication of data, uniform distribution and availability. This means that, after letting the system evolve for some time, the following properties should exist: each piece of data should be held by more than one agent; any (reasonably small) area of the grid should contain copies of all pieces of data present in the system; if requested to, any agent should be able to get hold of any piece of data, i.e. copies of all pieces of data should exists in the proximity of any agent.

While the experimental setup is simple, the purpose of this research was to show how a system should be designed in order to obtain emergent properties like the ones specified. Even if the system is generic (not having requirements related to a specific setup), there are still many challenges that arise from the full distribution of the system and from the use of local knowledge and interactions: the agents must not become overwhelmed with messages or with beliefs, nor must they become unresponsive or lose all knowledge about them and their surroundings. All that while keeping the behaviour basic enough to be efficient on devices with limited capabilities.

To design agents that will lead, by means of local interaction, to global properties, we have followed ideas from previous work on systems with emergent properties [3, 4, 9, 21, 22, 38]. We apply these ideas to a system of cognitive agents (as opposed to reactive agents), envisaging a behaviour that relates to information, rather than more simple concepts. First, the emergent properties cannot be of different nature than the local capacities of the agent. For example, if the agent can only move around, what we can expect as a global behaviour of the system would be for the agents to form structures of a certain

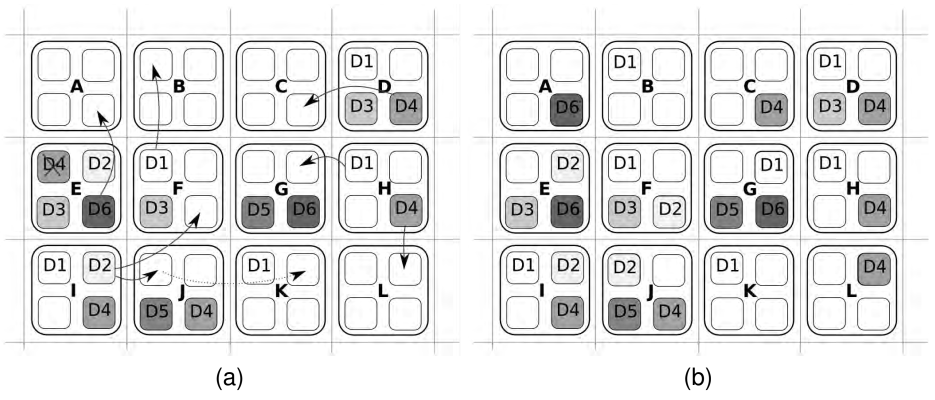


Fig. 1. Group of agents (named **A** to **L**), before (a) and after (b) the data transfers indicated by the arrows. There are 6 data pieces in the system: D1 to D6.

shape [38]. Local behaviour should reflect the features of the global desired behaviour. There should also be feedback, that will lead to the stabilization of formed structures and to the dynamical equilibrium of the system [18].

In our design, a lot of inspiration has been taken from the human behaviour (as it is with the design of software agents in general), providing the agents with capabilities that will keep their knowledge bases, goal lists and message queues at reasonable loads. In order to obtain data replication and distribution, an agent is ”curious” and, if capacity is available, it will be interested in and it will request a copy of a piece of data it does not hold. In order to obtain variation and uniformity in the data distribution pattern, an agent will ”lose interest” in pieces of data that are already held by most of the neighbour agents and may ”forget” them. In order for an agent to be able to get hold of pieces of data not held by itself or any of the neighbours, agents will be able to collaborate for the common goal of finding a certain piece of data that, since data is well distributed, cannot be very far away. More on the implementation of these features is presented in the description of the agent behaviour.

Figure 1 gives an example of a step in the functioning of a group of agents in the system. In the presented state, there are still many agents that don’t have any data. Consider, however, every agent has sufficient knowledge about his vicinity. For example, agent **I** has the following beliefs about surrounding data:

$\langle I\ has\ D1\rangle$ $\langle I\ has\ D2\rangle$ $\langle I\ has\ D4\rangle$
$\langle J\ has\ D4\rangle$ $\langle J\ has\ D5\rangle$
$\langle E\ has\ D4\rangle$ $\langle E\ has\ D2\rangle$ $\langle E\ has\ D3\rangle$ $\langle E\ has\ D6\rangle$
$\langle F\ has\ D1\rangle$ $\langle F\ has\ D3\rangle$
$\langle G\ has\ D5\rangle$ (knows that from F)

Each agent that has less than 75% of its capacity full will ask for some piece of data from one of its neighbours. Agent **E** will discard data $D4$, as both **I** and **J** also have it (it has to discard something, so it will discard the data that is most present in its vicinity). Supposing that agent **K** needs D2 (for example

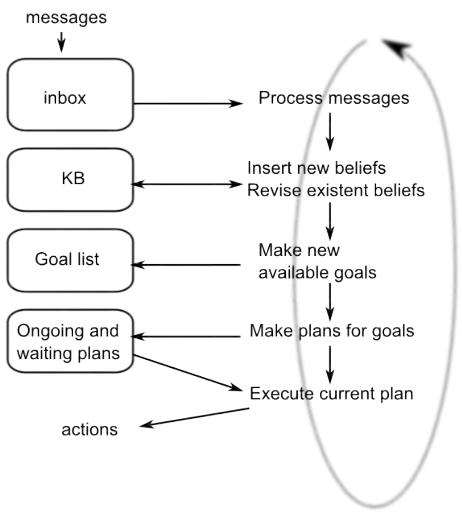


Fig. 2. The basic execution cycle of an agent. The main structural components involved are the knowledge base (KB) and the list of available goals (*Goal list*)

because there is an external request for it) it will communicate its intentions to its neighbours and **J** will collaborate with it and retrieve D2.

The system is using cognitive agents and the implementation is based on the Beliefs-Desires-Intentions model [31]. There are several reasons for using cognitive agents. First, considering our long-term AmI-related goal, there is an obvious necessity for the agents to be cognitive, so that they are capable to aggregate and reason on the information that they hold. But, more important than that, cognitive agents are more capable and can act better in many situations [32]. If their behaviour is flexible, they can even run on resource-constrained devices. Although the behaviour of a cognitive agent system may be harder to control and to predict, the use of cognitive agents can bring many advantages in the future, and the study of emergence in such a system is vital for future research.

The structure of an agent is presented in Figure 2. The working cycle of an agent is straightforward: the agent processes the messages that it receives (there are no other types of perceptions – new data is also inserted into the system as messages); revises its beliefs and its current available goals; chooses a goal and makes a plan for it; executes the actions in the current plan. Details about these steps and the structures involved are given in the sections below.

3.1. Agent Beliefs

The information in the agent's knowledge base (KB) is stored in Facts. In general, a Fact can be either a basic, atomic notion (like *Agent* or *Data*) or a relationship between two other facts. The resulting structure is a concept map,

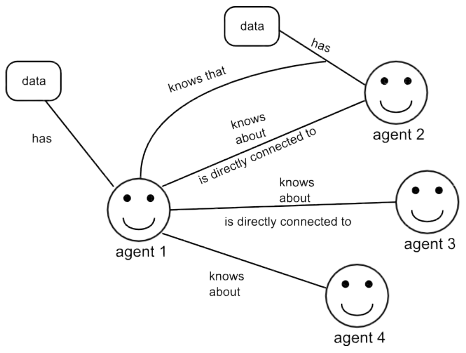


Fig. 3. Example content for an agent's knowledge base (here, the knowledge base of the agent called agent-1). Several relationships are present: *knows about*, *is connected to*, *has*, *knows*. This is only an example and actual agent knowledge bases rise to a greater number of facts.

formed of basic notions and relationships between them. At this point in the development of the system, a simplified approach is used, and a Fact may have one of these forms:

$\langle Agent, knows\ about, Agent\rangle$
$\langle Agent, is\ directly\ connected\ to, Agent\rangle$
$\langle Agent, has, Data\rangle$
$\langle Agent, has, Goal\rangle$
$\langle Agent, knows, Fact\rangle$

Note that the last definition is recursive, and the recursion must end with one of the other four forms. Most Facts are of the last three types. Considering the structure of Goals (described in the next section) and the agent behaviour, most facts will refer, in the end, to a piece of $Data$.

An example is presented in Figure 3. We consider that the displayed facts are in the knowledge base of the agent named *agent-1*. Relationships of the type "knows" link the agent to all facts that it has: the agents that it knows about, the facts that it knows and what it knows about the knowledge of other agents. Actually, the "knows" relationship between the agent and a fact is equivalent to the fact belonging to the agent's knowledge base, so there is no need to represent it explicitly.

Agents may also know about other agent's goals, so that it is possible for them to cooperate for the realization of goals. *Data* are pieces of information that are not further represented as facts, and that are considered as atomic.

This concept-map-like structure allows agents to represent partial information on the system. It is also possible for agents to easily attach new information (also represented as a concept map) to their knowledge base, or to send only parts of their knowledge base to other agents. This representation was chosen because it is flexible but also does not consume much space. Even if the presented structure allows very long chains and graphs of $\langle Agent, knows, Fact\rangle$

relationships, the behaviour of the agent has been designed so that information about farther agents will be discarded sooner. Moreover, forgetting may be flexible according to the agent's KB capacity.

3.2. Agent Goals

Following the ideas presented above, and taking inspiration from existing studies of emergent properties in reactive agent systems, the goals of the agents have been designed so that they will be local but will, however, reflect the spirit of the desired global emergents. According to this policy, goals of an agent are of the following types (in parenthesis – the name of the type in the internal representation):

- Get interesting data that the agent knows about and fulfill external requests, if any (*GET*).
- Maintain capacity free, ready for potential data injected from the exterior (*FREE*).
- Inform neighbour agents about new data (*INFORM*).

Apart from that, the agent will also respond to requests for data from neighbour agents, if it has the requested data. This is done before treating any other types of messages, as it does not require any reasoning and is a reactive behaviour.

Goals are represented as pairs of the form $\langle Goal\_type, Object \rangle$, where the element $Object$ may be the description of a piece of data (its identifier), in the case of *GET*, or a fact about a piece of data, in the case of *INFORM*.

The goal to free capacity has no *Object* and is activated when the free capacity reaches a percentage of the total capacity below a certain threshold. In the experiments presented in this paper, this threshold was 25%. As the pieces of data had equal size, and an agent had a maximum capacity of 4 pieces of data, 25% was chosen as a tradeoff between appropriate use of an agent's capacity and always having room for one more piece of data.

When the used capacity of an agent is under 75% (free capacity is over 25%), it will choose to get a piece of data that is held by a neighbour but not by itself. In order to support the replication of data, it will choose the least well represented in the vicinity (i.e. present in the least number of copies in neighbour agents).

3.3. Agent Behaviour

The types of goals that are presented above will be instantiated according to the beliefs of the agent. For example, the first type of goal (called *GET*) may be instantiated as $\langle GET, D3 \rangle$, meaning that the agent has the goal to get hold of data $D3$.

Instantiated goals are generated permanently, based on the information that the agent receives, and they are put in the list of *available goals*. This list is sorted according to the *importance* of the goals. Importance is primarily computed depending on the type of the goal. *FREE* is the most important type. Then *INFORM* and *GET* follow. This way, facts about Data will spread first, and then the Data itself will spread. Between different *INFORM* and *GET* goals, more important are the goals referring to Data that is closer (i.e. the chain of $Agent\ knows\ Agent\ knows ... Agent\ has\ Data$ is shorter). From the Goal list the most important goal will be chosen and a plan will be built with actions that the agent expects would lead to the desired result.

In order to fulfill its goals, an agent has the following available actions:

– Send a request for data to a neighbour.
– Discard a piece of data.
– Send a message to a neighbour, containing relevant information (for example that the sending agent has a new piece of data), in the form of facts.
– Send a message to a neighbour, containing a goal of the sending agent, also in the form of a fact.

Say, for example, that agent **A** needs to get data $D3$. If the agent knows that any of the neighbours have the data, i.e. it has a belief $\langle Agent, has, D3\rangle$, where $Agent$ is a neighbour, then it will send a direct request for data $D3$. If $Agent$ is not a neighbour, but **A** knows that $Agent$ is a neighbour of **B**, and **B** is a neighbour of **A**, it will plan to first send a message containing information about its goal $\langle GET, D3\rangle$ to **B**, and then put the plan in wait. Eventually it will be informed that **B** has acquired data $D3$. If that doesn't happen in a certain amount of time, **A** will contact **B** and other neighbours and send them its goal. Finally, in the case when **A** does not know of any agent holding data $D3$, it will send the message containing its goal to all the neighbours.

The behaviour of an agent, i.e. the stages that an agent goes through during a step of the system's evolution, is described below:

– **Receive data from and send data to** neighbour agents. Data is transmitted only as a response to previous requests. These operations are simple and need no reasoning, and that is why they are handled separately and before any other decision is made.
– **Revise beliefs**. This is done based on information received from the other agents. Duplicate facts are found and removed. Also, circular facts are identified and discarded.
– **Check waiting plans** for completion (was the goal achieved?). In the case of completion the plan is discarded. Most plans are discarded immediately after all actions are completed. Plans waiting longer than that exist only for goals of type *GET*.
– **Make plans**. Take the goal with the highest importance. If there is no plan for it, make a plan composed of the actions needed to be taken and put it in the list of ongoing plans.
– **Execute plans**. Take the plan associated with the most important goal and execute its next action. If there are no more actions to be performed, move the plan to the list of waiting plans. It will be checked for completion in the next cycle.

– **Fade memory**. Discard old facts and facts that refer to data that is further away (calculated as described above). To prevent the Goal list from overloading, unimportant goals are also discarded.

In order to keep the system in constant change (away from static equilibrium), which is one of the factors that leads to self-organization [18], when there are no more available goals the agent will ”lose interest” in a piece of data, the one that is present in a greater number of copies in the neighbours (at least according to the agent’s beliefs) and it will discard it. Note that even if an agent will lose interest in data, he is able to regain interest in that data when the data will become more rare in its vicinity. This, of course, leads to a constant movement of data through the system, but is also a necessity for dynamic equilibrium, which in turn helps maintain a robust decentralized system.

Being a complex system (experiments were performed on a system composed of 400 agents), and all agents acting locally, it is clear that the system as a whole will behave, many times, unpredictably. That is why there are several tweaks that had to be performed, across many experiments, in order for the system to function as expected, so that agents do not become overwhelmed with messages, or with available goals, or with useless facts in their KB. These tweaks were implemented in the agents’ policies.

First, there must exist a good balance between how fast the agent can learn new knowledge and revise its beliefs and how often an agent broadcasts its own knowledge and intentions to its neighbours. If an agent broadcasts all its knowledge each time it learns something new, all agents will be flooded with messages they don’t have time to process. If the agents broadcast their knowledge too rarely, this might result in the agents having outdated beliefs about their neighbours.

Another element that needs good balance is the rate at which an agent forgets. If an agent forgets too quickly, it might have already forgotten some important information at the time it is supposed to be using it. If an agent forgets too slowly, it will have a lot of information that is outdated and of no use anymore.

It is arguable that these tweaks will always depend on the exact application the system is used for, but it is true that in order to achieve self-organization and coherent emergent properties, any complex system must be in a delicate balance between fixed, static equilibrium and chaotic behaviour, so this sort of tuning will still be necessary (the system is not self-tuning).

4. Results and Discussion

Many experiments were carried out throughout the development of the system, first for assuring that the agents were not overwhelmed by messages and actions to perform, second for the actual study of the evolution of the system. The presented experiments involved a square grid of 400 agents, the pieces of data in the system were of equal size, and each agent had a capacity of 4 pieces of data. As said before, data is generic, and is characterized only by an identifier.

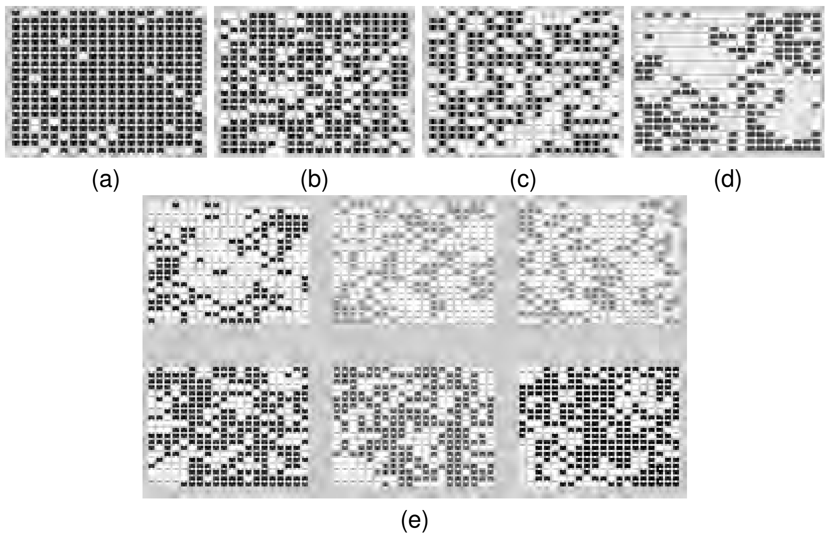


Fig. 4. The distribution of one piece of data after 150 steps, in a system with agents of capacity 4 in the context of a total number of pieces of data of: (a) 3, (b) 4, (c) 6, (d) 8. The simultaneous distributions of 6 pieces of data after 150 steps (e).

The resulting pseudo-stable distributions of data were monitored, but also the dynamic evolution of the system. The purpose of the experiments was to obtain a system that distributes data uniformly throughout the system, while using only local knowledge and interactions.

The system was implemented as a Java application. The execution of the agents is synchronous – at any moment of time all agents are executing the n^{th} step of their evolution. Besides the objectives stated above, one other objective was to make the system run as fast as possible, making the behaviour of individual agents more efficient. This also lead to lower simulation times and the possibility to perform more experiments. A typical experiment lasted 200-250 steps in the multi-agent system's time and about 15 seconds in real time on a machine with an Intel Core 2 Duo 3.33GHz CPU, with 2GB of RAM memory.

Figure 4 (a-d) presents, in several cases, the resulting distribution of a certain piece of data inserted into the system. What changes between the four distributions is the number of other pieces of data that also exist in the system.

In the first case (Figure 4(a)) there are only 2 other pieces of data in the system, of equal size. The system is at 75% capacity, so normally all agents should have all three pieces of data. The image reflects that, however, because the system is intentionally unstable, and agents have the ability to forget, there are some agents that do not have the piece of data that was monitored.

In the following two cases (Figure 4(b, c)), featuring other 3, respectively 5 pieces of data in the system, the resulting distribution becomes more scarce, but nevertheless uniform. In the last case (Figure 4(d)) the distribution is less

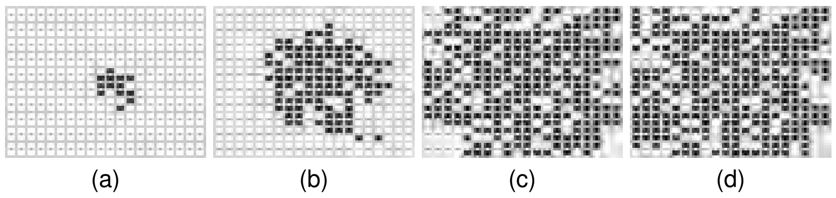


Fig. 5. The distribution of a certain piece of data over several steps of the system's evolution, in a system already holding 4 other pieces of data.

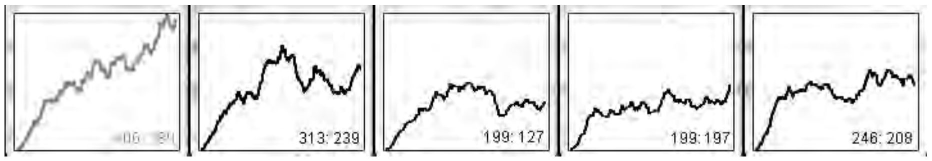


Fig. 6. Each graph corresponds to a piece of data in the system, and represents the evolution of the number of facts about that piece of data (over the whole system), since the beginning of the system's evolution to the current step (step 200). Every image also contains a quantitative indication of the maximum value of the graph as well as the current value.

uniform (there is a total of 8 pieces of data in the system), but still the data is in a range of 3 agents from any agent.

It is also interesting to follow how the distribution of one piece of data evolves as the system runs. Figure 5 shows the distribution of a piece of data injected in a system already holding other pieces of data. The distribution grows fairly slow, and no solid distributions are found.

Figure 6 presents the evolution over time of the number of facts (over the whole system) about the 5 pieces of data in the system. This number is slightly larger than the number of agents actually having that data. On these graphs one can easily follow the way the agents become interested in data and then lose interest and forget many of the facts regarding it. Observe that facts about the different pieces of data do not exist in equal numbers throughout the system. That is due to the fact that this is a complex system. Feedback and non-deterministic behaviour result in outcomes that are not fully predictable. However, the numbers are well in the same order of magnitude and considering the little amount of knowledge that each agent has, the distributions are very uniform.

It is relevant to note that the distributions that are displayed in the figures are never stable. The ”holes” in the distributions – the agents that do not hold the data – change all the time, as they lose interest, forget, and become interested again. This instability is also shown in Figure 6, where the curves are not smooth, but present many small spikes.

Although all the examples presented here involve the same number of agents, this number can be changed easily and the system scales with no problem, as every agent interacts only with and has knowledge only about its close vicinity.

What is important to point out is that very good distributions are obtained although none of the agents has that as an objective. Moreover, agents are not capable of measuring in any way the distribution of the data in the system, nor do they have any perception on the global state of the system. Their objectives are local and mostly selfish, but a coherent global result is obtained, which is an emergent property. As shown in the previous section, the system and the agents have been specifically designed to produce the emergents, by expressing the desired global result at the local scale of the individual.

The developed system makes a good platform for the study of emergent properties in a relatively simple cognitive agent system. Over the experiments, many monitoring tools have been developed, as following in detail the behaviour and evolution of a system formed by so many agents is not an easy task.

Although the designed system is still very simple, there are important differences between the implemented agents and reactive agents. It must be pointed out that the agents reason permanently about what data they should get and what they should keep. A simple form of collaboration has been implemented – agents are able to share goals. Reactive agents would not have been able to have beliefs about their neighbours and would not have been able to reason about what action would be best in the context.

5. Conclusion

Emergence is an essential notion in the field of multi-agent systems. It provides the possibility of obtaining a more complex outcome from a system formed of individuals of lower complexity. Although there are many recent studies concerning emergence, there is yet no clear methodology on how to specifically design a system so that it would manifest emergence. Moreover, most implementations use reactive agents, that have limited or no cognitive or planning abilities.

The paper presents a cognitive multi-agent system in which the agents' interactions allow the emergence of specific properties required for solving the problem. The system has been designed with the purpose of manifesting emergent properties: the agents have been given local goals that naturally lead to the global, desired, goal of the system.

As future work, the system will be improved, so that the emergents will be more nuanced. Data and facts will have certain context measures relating to their domains of interest and to their importance. Agents will also be able to have interests in particular domains. New tools for monitoring the system will be developed and the configuration of the network will be able to change – the agents that an agent communicates with will change over time, in order to simulate agent mobility and changes in the environment. Closer to the requirements in Ambient Intelligence, different agents will be able to have different capabilities, different storage capacity and different computing power.

Acknowledgements

This work was supported by CNCSIS - UEFISCSU, project number PNII - IDEI 1315/2008 and Grant POSDRU 5159.

References

1. Amaral, L., Ottino, J.: Complex networks: Augmenting the framework for the study of complex systems. The European Physical Journal B-Condensed Matter 38(2), 147–162 (2004)
2. Bernadas, A., Alfano, R., Manzalini, A., Solé-Pareta, J., Spadaro, S.: Demonstrating communication services based on autonomic self-organization. Proceedings of the 2008 International Conference on Complex, Intelligent and Software Intensive Systems-Volume 00 pp. 101–107 (2008)
3. Beurier, G., Simonin, O., Ferber, J.: Model and simulation of multi-level emergence. Proceedings of IEEE ISSPIT pp. 231–236 (2002)
4. Boschetti, F., Prokopenko, M., Macreadie, I., Grisogono, A.: Defining and detecting emergence in complex networks. Lecture notes in computer science 3684, 573–580 (2005)
5. Bourjot, C., Chevrier, V., Thomas, V.: A new swarm mechanism based on social spiders colonies: From web weaving to region detection. Web Intelligence and Agent Systems 1(1), 47–64 (2003)
6. Cabri, G., Ferrari, L., Leonardi, L., Zambonelli, F.: The LAICA project: Supporting ambient intelligence via agents and ad-hoc middleware. Proceedings of WETICE 2005, 14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises 5, 39–46 (2005)
7. Capera, D., Georgé, J., Gleizes, M., Glize, P.: Emergence of organisations, emergence of functions. In: AISB03 symposium on Adaptive Agents and Multi-Agent Systems (2003)
8. De Wolf, T., Holvoet, T.: Emergence versus self-organisation: Different concepts but promising when combined. Engineering Self Organising Systems: Methodologies and Applications 3464, 1–15 (2005)
9. De Wolf, T., Holvoet, T.: Design patterns for decentralised coordination in self-organising emergent systems. Lecture Notes in Computer Science 4335, 28–49 (2007)
10. Dolev, S., Tzachar, N.: Empire of colonies: Self-stabilizing and self-organizing distributed algorithm. Theoretical Computer Science 410(6-7), 514–532 (2009)
11. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.: Istag scenarios for ambient intelligence in 2010. Tech. rep., Office for Official Publications of the European Communities (2001)
12. Gleizes, M., Camps, V., Glize, P.: A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. In: Fourth European Congress of Systems Science (1999)
13. Goldstein, J.: Emergence as a construct: History and issues. Emergence 1(1), 49–72 (1999)
14. Grasse, P.: La reconstruction du nid et les interactions inter-individuelles chez les bellicositermes natalenis et cubitermes sp. la thorie de la stigmergie: essai d'interprtation des termites constructeurs. Insectes Sociaux 6, 41–83 (1959)

15. Hales, D., Edmonds, B.: Evolving social rationality for MAS using "tags". Proceedings of the second international joint conference on Autonomous agents and multi-agent systems pp. 497–503 (2003)
16. Hellenschmidt, M.: Distributed implementation of a self-organizing appliance middleware. Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies pp. 201–206 (2005)
17. Hellenschmidt, M., Kirste, T.: A generic topology for ambient intelligence. Lecture notes in computer science pp. 112–123 (2004)
18. Heylighen, F.: The science of self-organization and adaptivity. The Encyclopedia of Life Support Systems pp. 1–26 (2002)
19. Koestler, A.: The Ghost in the Machine. Macmillan (1968)
20. Mamei, M., Vasirani, M., Zambonelli, F.: Selforganising spatial shapes in mobile particles: the TOTA approach. Engineering Self-Organising System pp. 138–153 (2004)
21. Mamei, M., Zambonelli, F.: Spatial computing: the TOTA approach. Self-star Properties in Complex Information Systems Conceptual and Practical Foundations 3460, 307–324 (2005)
22. Mano, J.P., Bourjot, C., Leopardo, G., Glize, P.: Bio-inspired mechanisms for artificial self-organised systems. Special Issue: Hot Topics in European Agent Research II Guest Editors: Andrea Omicini 30, 55–62 (2006)
23. Marinescu, D., Morrison, J., Yu, C., Norvik, C., Siegel, H.: A self-organization model for complex computing and communication systems. Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems pp. 149–158 (2008)
24. Mills, K.: A brief survey of self-organization in wireless sensor networks. Wireless Communications and Mobile Computing 7(7), 823–834 (2007)
25. Picard, G.: Cooperative agent model instantiation to collective robotics. In: Engineering Societies in the Agents World V: 5th International Workshop, ESAW 2004, Toulouse, France, October 20-22, 2004: Revised Selected and Invited Papers. Springer (2005)
26. Prokopenko, M.: Design vs self-organization. Advances in applied self-organizing systems pp. 3–17 (2007)
27. Prokopenko, M. (ed.): Advances in Applied Self-Organizing Systems. Springer-Verlag, London, UK (2008)
28. Ramos, C., Augusto, J., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. IEEE Intelligent Systems pp. 15–18 (2008)
29. Randles, M., Taleb-Bendiab, A., Miseldine, P.: Harnessing complexity: A logical approach to engineering and controlling self-organizing systems. International Transactions on Systems Science and Applications 2(1), 11–20 (2006)
30. Randles, M., Zhu, H., Taleb-Bendiab, A.: A formal approach to the engineering of emergence and its recurrence. Proc. of EEDAS-ICAC pp. 1–10 (2007)
31. Rao, A., Georgeff, M.: BDI agents: From theory to practice. In: Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95). pp. 312–319. San Francisco, CA (1995)
32. Ricci, A., Omicini, A., Viroli, M., Gardelli, L., Oliva, E.: Cognitive stigmergy: Towards a framework based on agents and artifacts. Lecture Notes in Computer Science 4389, 124–135 (2007)
33. Rodriguez, S., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: An analysis and design concept for self-organization in holonic multi-agent systems. Lecture Notes in Computer Science 4335, 15–27 (2007)

34. Sen, S., Airiau, S.: Emergence of norms through social learning. Proceedings of the Twentieth International Joint Conference on Artificial Intelligence pp. 1507–1512 (2007)
35. Serugendo, G.D.M., Gleizes, M.P., Karageorgos., A.: Self-organization and emergence in MAS: An overview. Informatica 30(1), 45–54 (2006)
36. Standish, R.: On complexity and emergence. Arxiv preprint nlin.AO/0101006 pp. 1–6 (2001)
37. Unsal, C., Bay, J.: Spatial self-organization in large populations of mobile robots. Proceedings of the 1994 IEEE International Symposium on Intelligent Control pp. 249–254 (1994)
38. Zambonelli, F., Gleizes, M., Mamei, M., Tolksdorf, R.: Spray computers: Frontiers of self-organization for pervasive computing. Proceedings of the 13th IEEE Int'l Workshops on Enabling Technologies, WETICE pp. 403–408 (2004)

Andrei Olaru is a PhD student under joint supervision between University ”Politehnica” of Bucharest and University Pierre et Marie Curie, Paris. He is also a teaching assistant in the Department of Computer Science and a member of the Artificial Intelligence and Multi-Agent Systems Laboratory at University ”Politehnica” of Bucharest. His research interests include multi-agent systems, ambient intelligence and self-organization.

Cristian Gratie is a PhD student and teaching assistant at University ”Politehnica” of Bucharest, Department of Computer Science and also a member of the Artificial Intelligence and Multi-Agent Systems Laboratory. His main interests are multi-agent systems and abstract argumentation frameworks.

Adina Magda Florea is Professor in the Department of Computer Science and Head of the Artificial Intelligence and Multi-Agent Systems Laboratory at University ”Politehnica” of Bucharest. Her main research interests are multi-agent systems architectures and intelligent systems.

Received: January 19, 2010; Accepted: May 3, 2010.

UDC 519.863, DOI:10.2298/CSIS100118013B

Distributed Parameter Tuning for Genetic Algorithms

David F. Barrero[1], Antonio González-Pardo[2], David Camacho[2], and María D. R-Moreno[1]

[1] Department of Computer Engineering. University of Alcalá.
Ctra Madrid-Barcelona, Km. 33,6.
28871 Alcalá de Henares (Madrid), Spain.
{david,mdolores}@aut.uah.es
[2] Department of Computer Science. Autonomous University of Madrid.
C/Francisco Tomás y Valiente, n 11, 28049 Madrid, Spain.
{antonio.gonzalez,david.camacho}@uam.es

Abstract. Genetic Algorithms (GA) is a family of search algorithms based on the mechanics of natural selection and biological evolution. They are able to efficiently exploit historical information in the evolution process to look for optimal solutions or approximate them for a given problem, achieving excellent performance in optimization problems that involve a large set of dependent variables. Despite the excellent results of GAs, their use may generate new problems. One of them is how to provide a good fitting in the usually large number of parameters that must be tuned to allow a good performance.

This paper describes a new platform that is able to extract the Regular Expression that matches a set of examples, using a supervised learning and agent-based framework. In order to do that, GA-based agents decompose the GA execution in a distributed sequence of operations performed by them. The platform has been applied to *Language induction* problem, for that reason the experiments are focused on the extraction of the regular expression that matches a set of examples. Finally, the paper shows the efficiency of the proposed platform (in terms of fitness value) applied to three case studies: emails, phone numbers and URLs. Moreover, it is described how the codification of the alphabet affects to the performance of the platform.

Keywords: Genetic Algorithms, parameter tuning, agents.

1. Introduction

Sometimes system administrators, programmers or the data mining community need to retrieve some strings which satisfy a given pattern, for processing logs, or detecting spam. Instead of review all the information to find those strings, users can use some pattern matching tools. One of them is called Regular Expressions, or *regex* [8]. Although they provide a powerful and flexible notation to define and retrieve patterns from text, the syntax and the grammatical rules of these regex notations are not easy to use, and even to understand.

Language Induction [10, 20, 5] is a well known problem in Machine Learning and it consists of learning a grammar from a set of samples. There are several approaches from the Formal Languages and Theoretical Computer Science perspectives [23], however it is still an open problem. A recent approach to language induction is provided by the Evolutionary Computation community, where regex are evolved by means of evolutionary algorithms like Genetic Programming (GP) [15] or Genetic Algorithms (GAs) [19].

In this paper we present a method based on GA able to generate automatically regex from a set of positive and negative samples and we propose a new chromosome codification based on messy Genetic Algorithms (mGA) [11] and crossover operators. To carry out with the experimental phase, an existing multiagent framework, named Searchy [3], is adapted to allow the implementation of ours GA-based agents. This framework has a double goal. On one hand, to reduce the execution time of the experiments and, on the other hand, to improve the search capacity in the space problem considered, allowing agents to find better solutions.

The paper is structured as follows. First, a review of Regex, Variable Length Chromosomes, mGA and MAS is provided. Sections 3 and 4 present the codification and crossover operators respectively. The agent-based framework used in the experimentation is presented in section 5. Then, they are evaluated in section 6. Finally, some conclusions and future research lines are outlined.

2. Introduction

The Intelligent Agents and Multi-Agent Systems (MAS) research fields have experimented a growing interest from different research communities like Artificial Intelligence (AI), Software Engineering, Psychology, etc... Those research fields try to solve two distinct goals.

The first goal is to define and design software programs (usually called agents) which implement several characteristics like autonomy, proactiveness, coordination, language communication, etc... This goal tries to obtain an adaptive and intelligent program which is able to provide the adequate request to the inputs received from the environment [14].

The second goal is to create societies of agents. It is possible to coordinate several of those agents to build complex societies. When considering those societies new issues arise, like social organization, cooperation, knowledge representation, coordination, or negotiation. In this situation it is possible to speak about Multi-Agent Systems and the previous problems can be studied within different perspectives [16].

One of the aims of this paper is to study the use of a MAS within a GA framework. The merger of those perspective is not a new research area. There are some work related with this idea as [22, 17, 18, 9]. Nevertheless, the approach taken on [17, 18, 9] is quite different from the one of this work. While the aim, in the cited work, is the use of a MAS that benefits from a Genetic Algorithm, in this work the roles are interchanged, and Genetic Algorithm is benefited from the MAS, to evolve regex.

2.1. A Brief Introduction to Regular Expressions

Regular Expressions can be described as a particular kind of notation for describing sets of character strings. They provide a compact and expressive notation for describing patterns of text. When a particular string is in the set described by a regex, it is often said that the regex matches the string. Most of those characters in the pattern simply match themselves in a target string, so the regular expression ”www” matches that sequence of three letters wherever it occurs in the target. However, very few characters are used in patterns as *meta-characters* to indicate repetition (i.e. +), grouping (i.e. |), or positioning (i.e. $). The powerful pattern matching facilities provided by regex in different programming languages such as Perl, PHP, JavaScript, PCRE, Python, Ruby, or Java have not been conveniently exploited by the programmers or the computer scientists due the difficulty to write and understand the syntax, as well as the semantic meaning of those regular expressions.

Following the basic wildcards from IEEE POSIX Basic Regular Expressions (BRE) standard, and POSIX Extended Regular Expressions (ERE) notation, four wildcards are considered in this work:

– **Plus +**. Repeats the previous item once or more. Greedy, so as many items as possible will be matched before trying permutations with less matches of the preceding item, up to the point where the preceding item is matched only once.
– **Start ***. Repeats the previous item zero or more times. Greedy, so as many items as possible will be matched before trying permutations with less matches of the preceding item, up to the point where the preceding item is not matched at all.
– **Question mark ?**: Makes the preceding item optional. Greedy, so the optional item is included in the match if possible.
– **Pipe** |: Causes the regex engine to match either the part on the left side, or the part on the right side. Can be strung together into a series of options.

An example might be useful to better understand regex. The regex *pet* only match the string ”pet”, however if the plus symbol is introduced, *pet+*, the regex matches strings ”pet” as well as ”pett” and ”petttt”. If the start symbol is used instead of the plus, the regex *pet** matches all the previous strings as well as ”pe”. If the question mark is used, the regex *pet?* only matches two strings, ”pe” and ”pet”. Finally, the regex ”pet|r” can match ”pet” and ”per”. It should be noticed that meta-characters only affect the precedent character. In case it is needed to affect a group of characters, a parenthesis should be used.

The study of extracting the regex that describes a given set of examples is not new. Some authors have used Genetic Algorithm to perform this task [7],

[12], [2], nevertheless this work is completely different due to the use of messy Genetic Algorithm with a Multi Agent platform.

The following table, Table 1, shows an example of a regex that describes a basic URL. This is a good example the regex contains all the wildcards explained in this section. Table lists some strings matched by the regex and some other which are not recognized.

Regex	(http(s)?|ftp)://(a-z)+(\\.(a-z)+)*\\.com
Recognize	http://myweb.com
	https://my.web.is.secure.com
	ftp://webftp.com
Not Recognize	http://myweb01.com
	ftps://.com
	ftp://WeBfTp.com
	https://my..web.com

Table 1: Example of a regex and the strings that the regex matches and the strings not recognized.

2.2. Variable Length Chromosomes and messy GA review

GAs are part of the Evolutionary Computation, a computing paradigm inspired in the biological process of evolution. It can be considered as a stochastic search algorithm that represents the solutions of a problem as chromosomes using some codification. Chromosomes explore the search space through two genetic operators: mutation and sexual reproduction (*crossover*). The metric of how well a chromosome solves a problem is given by a fitness function. The genetic material contained in the chromosome is named genotype meanwhile the realization of such genetic material is referred as phenotype.

The number of variables involved in a GA problem is closely related to the chromosome length. Sometimes it is possible to determine the number of parameters that certain problem requires, and therefore to determine the chromosome length to introduce in the GA. However, there are many problems in which such approach is not an adequate solution because it is not possible to limit the size of the solution. The number of nodes in a neural network or the size of an evolved program in GP [15] are not easy to set prior to the execution of the algorithm. Of course, it is possible to set a maximum number of variables. But this approach sets an arbitrary limitation to the complexity of the solution. A more desirable solution is to use an algorithm able to adapt the size of the chromosome dynamically. This is the goal of the Variable-Length Genomes (VLGs) [13].

The main difference between fixed-length chromosomes and VLGs is the crossover operator. The simplest crossover used in VLGs is *cut and splice*.

Given two chromosomes (likely with different lengths), this crossover operator selects a random point in each chromosome and use it to divide them in two parts, then the parts are interchanged.

An early work in VLGs is the one developed by Goldberg with the messy GA [11]. It is a variable-length, gene position independent representation. Basically, mGA decouples the position of the genes from the phenotype that they represent and thus any gene can be placed anywhere within the chromosome. It is done representing each bit with a tuple $(locus, value)$ where the position, or locus, of the bit is specified as well as its value. Common genetic operators such as bit inversion mutation and cut and splice crossover are then applied to the chromosome constructed in this way.

Codification in mGA may generate two special situations that must be handled. Since the locus is coded within the allele, it might happen that not all the genes are defined, generating a problem called *underspecification*. Original mGA handles this situation by means of templates. A second problem arises when there are several alleles coding the same gene, i.e., the *overspecification*. mGA solves overspecification by means of a first-come-first-served philosophy.

mGA defines three phases: initialization, primordial and juxtapositional phase. The initialization phase deals with the generation of the initial population that feed the GA. It is composed by the set of all chromosomes of size k, where k is the gene size. This algorithm generates an unnecessary number of individuals, so the second phase filters the individuals trying to select individuals with a high density of good building blocks. The selected individuals are used then as the initial population in the GA. The generational based evolution is done in the juxtapositional phase.

3. Chromosome codification

There are a number of questions that must be answered in order to successfully implement a GA. One of these questions is how to represent in the chromosome the problem that is addressed. In this section three codifications able to represent a regex in a binary chromosome are presented: one based on a plain VLG and two codifications inspired in mGA.

The lexical approach that we have adopted requires an alphabet Σ of *atomic regex* x_i such as $\Sigma = \{x_0, x_1, ..., x_N\}$. Σ is constructed using the positive and negative samples. Atomic regex are identified applying Lempel-Ziv law [25]. This law states that texts are not composed by a uniform distribution of tokens, instead, a few tokens appear many times while many tokens have a reduced weight in the text. We build tokens using a set of symbols to divide the samples and then, those tokens that appear more times are selected to be part of the alphabet. The second subset of Σ is composed by a fixed set of symbols. This is an automatic and domain independent method that can be used with almost any codification schema.

3.1. Plain Variable-Length Genomes

Despite the inherent difficulty to determine *a priori* the length of the regex, it is possible to imitate a VLG by means of an island model [1] with immigration of individuals. It can take benefits of a parallel algorithm implemented with agents, however the population must be increased and it imposes a maximum length for the chromosomes. In this context selecting a codification able to deal with VLG seems to be a natural solution.

A simple way to compose and evolve the set of atomic regex in Σ is the traditional VLG, a binary chromosome of arbitrary length that is recombined using cut and splice, as it was described in sec. 2.2. The correspondence between the genes and the atomic regex in Σ is done as follows. Each gene contains l_g bits that code an integer number $i < 2^{l_g}$, then the gene represents the element x_{imodN} of Σ. Initial population has chromosomes randomly created with lengths uniformly distributed between a minimum chromosome length l_{min} and a maximum length l_{max}. The rest of the paper will use the term plain VLG to mean the codification schema described in this section.

3.2. Modified messy GA and biased messy GA

Plain VLG provides a simple variable-length genome coding, however variable-length genomes usually present some problems. One of them is the tendency to bloat the chromosome length, as Chu observed [4]. Another problem is the *genetic linkage*, i.e., the tendency of some alleles to remain joint due, for example, to crossover biases [21]. A solution to genetic linkage is mGA because it decouples the position of the gene in the chromosome from its semantics. It is still a simple codification and it seems to be a good choice to study how genetic linkage affects the regex evolution in VLGs. However, some modifications are needed in order to use mGA in the context of our work.

Like original mGA does, in our proposal genes are coded as $(locus, value)$, however $value$ follows the same coding scheme as the one described in 3.1 instead of being a single bit. It represents a symbol $x_i \in \Sigma$. We have integrated the original mGA initialization and primordial phases into one phase that generates Σ from the data set using Lempel-Ziv law, following the same philosophy that the used with VLG codification. With this approach there is no need to implement the primordial phase because the building blocks are integrated in Σ.

Initial mGA and revised versions of the algorithm such as fast mGA [6], do not generate the initial population randomly. There is rather a slight control about how to initialize them, as it was described in sec. 2.2. We propose also a modification of the initialization. Given a random number l uniformly distributed between l_{min} and l_{max}, a chromosome with l/l_g genes is created. $value$ is filled with a random value that codes an atomic regex following the same mechanism than plain VLG, while $locus$ takes a value from 0 to $l/l_g - 1$.

A second modification to the mGA named biased messy GA (bmGA) is also proposed. The biased messy GA instead of initializing the loci field with positions from 0 to $l/l_g - 1$, they are initialized with a biased loci, and thus their values range from $bias$ to $l/l_g + bias - 1$, where $bias$ can take several values. However, we have used $bias = 2^{l^{locus}}/2$, i.e., alleles begin to be placed in the middle position.

4. Recombination operators

The main role of the crossover is to recombine good chunks of chromosomes generating offspring [24] with better genetic information. Some authors have argued that the crossover performs better when it recombines two similar chromosomes [13], however this point is controversial and there are not a general consensus in the GA community. In the context of regex evolution, the disruptive properties is crossover is a main issue because of the rough nature of the fitness, a very small difference in the chromosome might lead to a dramatic change in the fitness. Following these ideas it seems natural to hypothesize that using a less destructive crossover operator will increase the performance of the GA in regex evolution.

The goal of the new crossover mechanism is to use the knowledge about the codification to recombine chromosomes in a less destructive way compared with the cut and splice crossover. Crossover is not directly performed with the chromosomes, instead an intermediate table is constructed. Our crossover proposal is divided in five phases as described.

1. *Integer chromosomes construction*. Alleles in the chromosomes (including their loci and values) are transformed into an integer representation. The order in which the alleles appear is respected.
2. *Intermediate table construction*. The intermediate table is a table composed by three columns and as many rows as the sum of not underspecified genes in the chromosomes. One column contains the sorted loci while the latter two columns contain each one the values (if any) defined for such locus.
3. *Crossover*. The intermediate table can be seen as two chromosomes, and thus any traditional crossover operators (one point, two points and uniform crossover) can be applied just interchanging the values of the chromosomes columns in the table.
4. *Recombined integer chromosomes construction*. Two integer chromosomes are constructed using the recombined intermediate table, it is the inverse operation of the phase two. It should be noticed that because of the lack of genetic linkage the position of the alleles is irrelevant for the crossover and thus their position can be changed without loss of relevant information.
5. *Recombined binary chromosomes construction*. The integer chromosomes are representated with a binary codification.

An example of modified one-point crossover is shown in Fig. 1. Two chromosomes are recombined in the example. Both use seven bits to code each gene, divided in three bits for the locus and four bits for the value. Chromosome

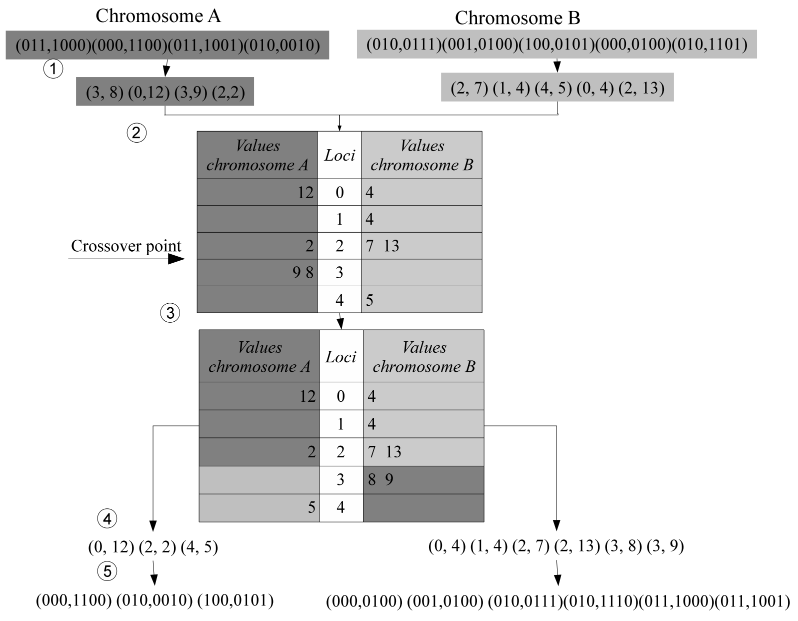


Fig. 1: Modified one point crossover example

A is composed by four alleles and chromosome B is composed by five alleles. Chromosome A and B present overspecification as well as underspecification. The intermediate table is constructed and, as it can be seen in the figure, underspecified genes correspond to empty cells. On the other hand, overspecified genes correspond to cells with several sorted values. A random point is used to interchange cells in the table, generating the recombined chromosomes. Any other traditional crossover mechanism may also be applied.

5. The GA evaluation framework

Due to the high number of GA runs that must be performed and the parallel nature of the GA, the set of experiments were run in a MAS that decomposes the GA evolution in a sequence of operations performed by different agents. This MAS has been deployed using the Searchy platform [3]. In this way the experimentation can be divided into different simple operations that are composed and executed in parallel, increasing the performance and the search capability of the algorithm.

There are six roles defined in the MAS: control, population, crossover, fitness evaluation, codification and alphabet agent. Each role is implemented using a specialized agent and there are several agents to implement crossover and codification. A description of each role is briefly presented.

1. *Control Agent*. It is responsible for the execution of the experiment, and has to fulfill some tasks, such as the initialization of Population Agents and control the execution of the experiments. It also gathers measures from the populations and generate statistics, averaging the measures of all the GA executions.
2. *Population Agent*. This agent contains a population of individuals represented by a binary chromosome. It also performs the generational evolution of the population using the services provided by the crossover, fitness evaluation and coding agents services.
3. *Crossover Agent*. A crossover agent is an agent that performs a crossover between two chromosomes. Actually, there are four different crossover agents that implement the four crossover operators under study. Cut and splice crossover can be performed in any codification under study while modified one, two and any point crossover requires a mGA or bmGA.
4. *Fitness Evaluation Agent*. This is an agent that, given a string regex is able to evaluate its extraction capabilities using a training set. It should be noticed that since it takes a string as input, this agent in not affected by the chromosome codification.
5. *Codification Agent*. The codification generates the phenotype associated to a given chromosome, i.e., it transforms a chromosome into a string that contains a regex. This regex is used by the Population Agent prior to evaluate any individual's fitness. There are two codification agents, the Plain VLG Coding Agent and the mGA Coding Agent. Since the only difference between mGA and bmGA is the initialization of the populations there is no need to use a bmGA Coding Agent.
6. *Alphabet Agent*. The alphabet agent takes as input the set of positive examples and using the Lempel-Ziv law identifies a set of tokens that are used to generate the atomic regex alphabet. The alphabet is used by the Codification Agents to generate the string regex.

Fig. 2 depicts the MAS architecture. First, the Control Agent (1) initializes several Population Agents (2) and associate each population with a Crossover Agent (3) and a Codification Agent (4). In this way each Population Agent contains an experiment involving a certain crossover operator and codification. Once the Population Agents have been initialized they evolve their populations for a number of generations, then they return to the Control Agent several measures. The Control Agent repeats this process a given number of times and then averages the measures.

The Alphabet Agent (5) reads the positive examples and generates the alphabet once, then it is provided to the Coding Agents which set a correspondence between each element in the alphabet and the codification used in the genome. It should be noticed that no agent with the exception of the Coding Agents need to know how the chromosome is coded, they manipulate the chromosome as a sequence of bits. The only agent that does not require a binary

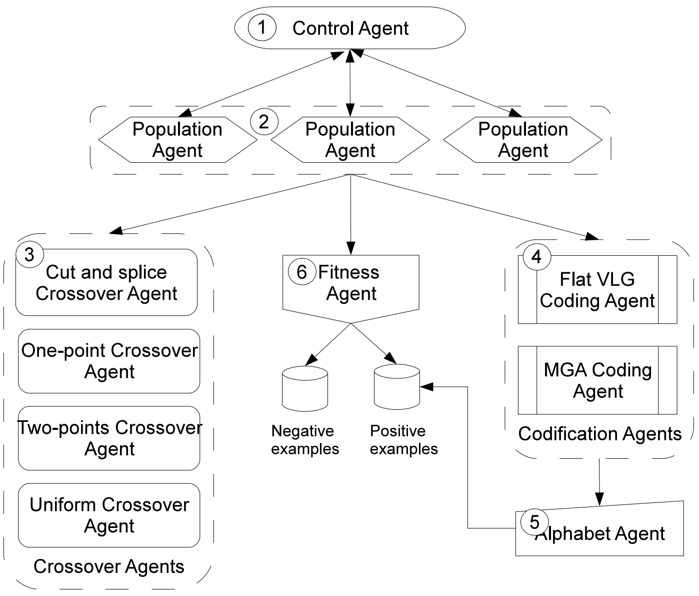


Fig. 2: MAS architecture used in the evaluation of the proposed crossover operators and codifications.

chromosome is the Fitness Agent (6) because it receives the regex in form of string, instead of a binary chromosome.

6. Experimental study

This section describes the behaviour and extraction capabilities of the coding and crossover mechanisms described in sections 3 and 4 using a MAS.

6.1. Experimental setup

Experiments have been carried out in three phases: parameter tuning, codification and genetic regex. In the first one, several GA runs are performed with different parameters to select the optimum parameters in order to use them in the remaining experiments. Then the codification (VLG, mGA and bmGA) and genetic operators (modified one-point, two-points and uniform crossover) presented in this paper are studied.

Three case studies are used in the experiments where regex able to extract emails, phone numbers and URLs are evolved. These are three well known problems in data mining literature. Each case study uses a dataset with positive examples that have been divided into a training set and a testing set. Meanwhile the negative examples are shared among the study cases. Due to the stochastic nature of the GAs, each experiment has been run one hundred times, and the data has been averaged.

The fitness evaluation that has been used in all the experiments can take values from 0 to 1, where 1 is the maximum fitness that any chromosome can achieve. The calculus of the fitness is performed as follows. For each positive example the proportion of extracted characters is calculated. Then, the fitness is calculated subtracting the average proportion of false positives in the negative example set to the average of the characters correctly extracted. In this way, the maximum fitness that a chromosome can achieve is one, and it happens when the phenotype has extracted correctly all the elements of positive examples while no element of the negative examples has been matched. Let us name it as ideal individual. Then, an ideal individual is able to extract correctly all the elements of positive examples while no element of the negative examples is matched.

6.2. Experimental results

The first experimental phase is a study about the behaviour of the algorithms under study under different parameter settings, whose aim is to select the GA parameters. Results are shown in Table 2. Optimum parameter values are similar for all the investigated algorithms with one notable exception, the mutation probability. Algorithms that use cut and splice crossover operator (VLG and bmGA cs) have an optimum mutation probability around one order of magnitude lower than the others algorithms (bmGA with any form of our proposed crossover). The higher disruptive capabilities of cut and splice operator compared to the proposed crossover operator may explain this difference. Parameters shown in Table 2 are the ones used along the rest of the experimentation described in this paper.

A second set of experiments were executed to study the performance of the three described codifications. In order to obtain comparable results, a cut and splice recombination operator has been used in all the experiments belonging to this second experimental stage. The three case studies yield similar experimental results, as can be seen in Fig. 3. Results suggest that plain VLGA achieves higher best fitness, however Fig. 3(b) shows, in generation 60, a slightly higher fitness for bmGA. In any case, plain VLG increases its best fitness faster than messy codifications due to its smaller chromosome: plain VLG does not need to codify the locus.

Compared to mGA, bmGA performs slightly better, specially in the phone numbers case study (see Fig. 3(b)). The better performance of bmGA compared to mGA in our experiments can be explained by the dynamics of the construction of the phenotype. Using a pure mGA, the first position of an atomic regex is 0, and thus the regex cannot be expanded to the left because there is no natural number lower than 0. BmGA places the first regex in $bias$ and thus by

Table 2: Parameters for the experiments carried out using a basic VLG (VLG), bmGA with cut and splice crossover (mbGA cs), bmGA with modified one-point, two-points and uniform crossovers (bmGA one, bmGA two and bmGA uni)

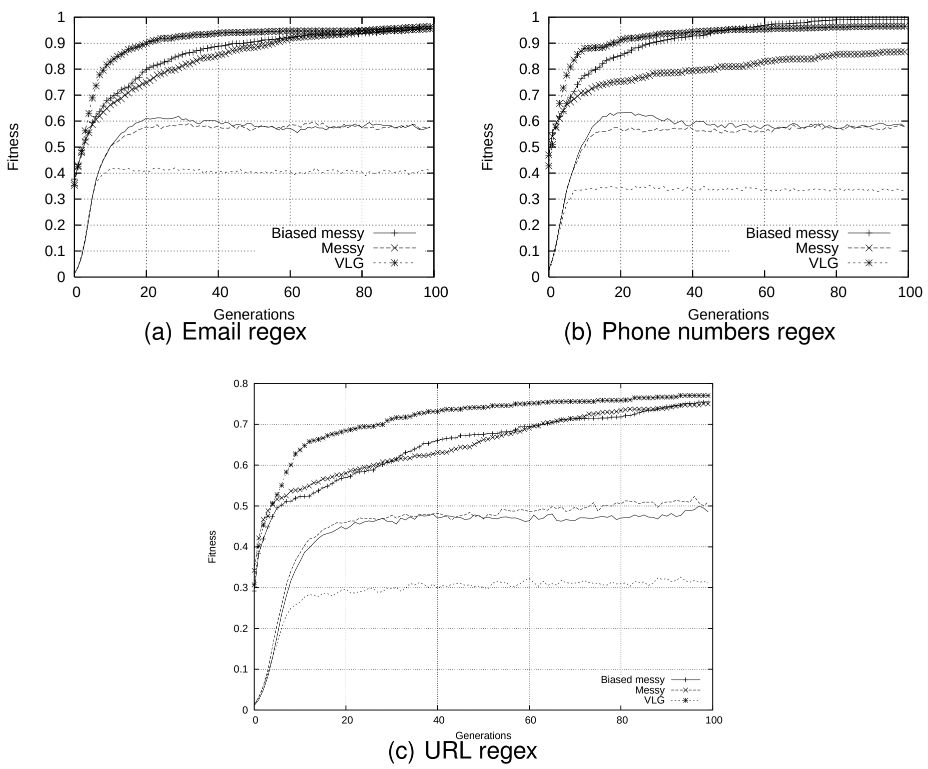
Settings	VLG	bmGA cs	bmGA one	bmGA two	bmGA uni
Mutation probability (P_{mut})	0.005	0.002	0.02	0.01	0.015
Population size (n)	50	50	50	50	50
Tournament size (t)	3	3	3	3	3
Min. chromosome length (l_{min})	4	9	9	9	9
Max. chromosome length (l_{max})	40	90	90	90	90
Gene length (l_g)	4	9	9	9	9
Loci length (l^{loci})	-	5	5	5	5
Values length (l^{values})	-	4	4	4	4
Crossover probability (P^c)	-	-	-	-	0.3

Table 3: Comparison of crossover operators for email regex induction: Cut and splice crossover (cs), modified one-point (one), two-points (two) and uniform crossovers (uni).

	cs	one	two	uni
Best fitness	0.96	0.94	0.9	0.94
Avg. fitness	0.58	0.42	0.58	0.46
Prob. ideal	0.86	0.78	0.64	0.77

Table 4: Comparison of crossover operators for phone number regex induction: Cut and splice crossover (cs), modified one-point (one), two-points (two) and uniform crossovers (uni).

	cs	one	two	uni
Best fitness	0.99	0.97	0.95	0.98
Avg. fitness	0.58	0.43	0.6	0.43
Prob. ideal	0.90	0.77	0.66	0.80



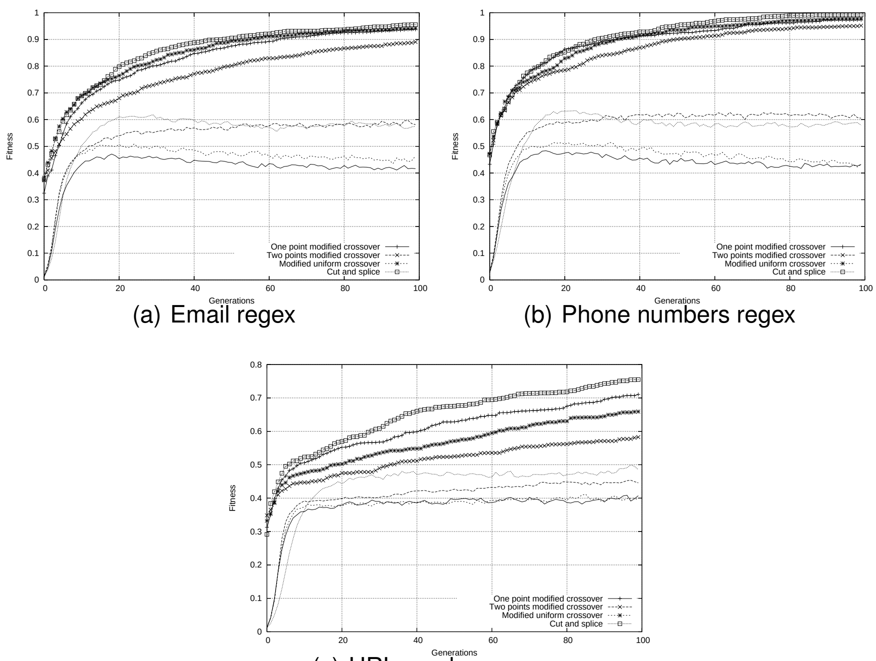
(a) Email regex (b) Phone numbers regex (c) URL regex

Fig. 3: Comparison of codifications in regex evolution. Best individual and average fitness are shown.

means of mutation and crossover regex can grow to the left, avoiding premature convergence.

The third group of experiments deals with the empirical study of several crossover mechanisms for messy algorithms. Experiments showed that bmGA performs better than mGA, and therefore bmGA is used in this section to compare several crossover operators, including cut and splice and the proposed operators described in sec. 4. Tables 3 and 4 show the best fitness, mean fitness and probability of finding an ideal individual for both case studies being investigated. Results show that the crossover operator has a limited effect in the fitness. Cut and splice seems to outperform the other operators, however it would be desirable to use hypothesis contrast to proof it.

The evolution of best and average fitness for the crossover operators under study are depicted in Fig. 4 for the three study cases: email (a), phone number (b) and URL evolution (c). It can be seen that the crossover operator has a limited effect the in fitness evolution, cut and splice performs slightly better that the other operators, however there is a small difference. The operator that per-



(a) Email regex (b) Phone numbers regex

(c) URL numbers regex

Fig. 4: Dynamics of fitness for crossover operators under study with a bmGA codification.

forms worse is the two points crossover operator, while there is no substantial difference between the modified one-point and uniform crossover.

7. Conclusions and future work

We have presented a method to generate regular expressions using supervised learning and an agent based testing framework. The distributed testing framework used has been a satisfactory approach due to the enhanced performance and easy composition of tasks involved in the GA. Additionally a brief empirical analysis of how different codifications and crossover mechanism influence the evolution of regex has been presented. The set of experiments carried out showed that the best performance is achieved with a direct codification of the alphabet using a plain VLG.

These results leads us to conclude that there are some intrinsic limitations in the evolution of regex regardless of the codification and crossover operator used. The main one is the linear nature of GAs that incited us to use a lexicographical codification of regex, there is not a trivial way to code regex operators that affect a groups of characters or nested operators. From this point of view, it seems that the use of pure GAs to evolve grammars and languages has serious constrains.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Innovation under the projects COMPUBIODIVE(TIN2007-65989), V-LeaF (TIN2008-02729-E/TIN), Castilla-La Mancha project PEII09-0266-6640 and HADA (TIN2007-64718).

References

1. Barrero, D.F., Camacho, D., R-Moreno, M.D.: Data Mining and Multiagent Integration, chap. Automatic Web Data Extraction Based on Genetic Algorithms and Regular Expressions. Springer (Aug 2009)
2. Barrero, D.F., González, A., R-Moreno, M.D., Camacho, D.: Variable length-based genetic representation to automatically evolve wrappers. International Conference on Practical Applications of Agents and Multi-Agents System, vol. 2, p. To Appear. Springer (2010)
3. Barrero, D.F., R-Moreno, M.D., López, D.R., García, Ó.: Searchy: A metasearch engine for heterogeneus sources in distributed environments. In: Proceedings of the International Conference on Dublin core and Metadata Applications. pp. 261–265. Madrid, Spain (Sep 2005)
4. Chu, D., Rowe, J.E.: Crossover operators to control size growth in linear GP and variable length GAs. In: Wang, J. (ed.) 2008 IEEE World Congress on Computational Intelligence. IEEE Computational Intelligence Society, IEEE Press, Hong Kong (1-6 Jun 2008)
5. Cicchello, O., Kremer, S.C.: Inducing grammars from sparse data sets: a survey of algorithms and results. J. Mach. Learn. Res. 4, 603–632 (2003)
6. Deb, K.: Binary and floating-point function optimization using messy genetic algorithms. Ph.D. thesis, Tuscaloosa, AL, USA (1991)
7. Dunay, B.D., Petry, F., Buckles, B.P.: Regular language induction with genetic programming. In: Proceedings of the 1994 IEEE World Congress on Computational Intelligence. pp. 396–400. IEEE Press, Orlando, Florida, USA (27-29 June 1994)
8. Friedl, J.E.F.: Mastering Regular Expressions. O'Reilly & Associates, Inc., Sebastopol, CA, USA (2002)
9. Galinho, V.J.P., Thierry, G., Franck, L., Alain, C., Rouen, I.D., Blondel, P.E.: Genetic algorithms in a multi-agent system (1997)
10. Gold, E.M.: Language identification in the limit. Information and Control 10(5), 447–474 (1967)
11. Goldberg, D., Deb, K., Korb, B.: Messy genetic algorithms: motivation, analysis, and first results. Complex Systems 3(3), 493–530 (1989)
12. González-Pardo, A., Barrero, D.F., Camacho, D., R-Moreno, M.D.: A case study on grammatical-based representation for regular expression evolution. International Conference on Practical Applications of Agents and Multi-Agents System, vol. 2, p. To Appear. Springer (2010)
13. Harvey, I.: The saga cross: the mechanics of recombination for species with variablelength genotypes. In: In R. Manner & B. Manderick, (Eds.), Parallel Problem. pp. 269–278. North-Holland (1992)

14. Jennings, N.R., Sycara, K.: A roadmap of agent research and development (1998)
15. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems). The MIT Press (December 1992)
16. Lander, S.E., L, S.E., Lesser, V.R.: Understanding the role of negotiation in distributed search among heterogeneous agents (1993)
17. Lymperopoulos, D.G., Tsitsas, N.L., Kaklamani, Dimitra, I.: A distributed intelligent agent platform for genetic optimization in cem: Applications in a quasi-point matching method. IEEE Transactions on Antennas and Propagation 55(3), 619–628 (March 2007)
18. Maione, G., Naso, D.: A genetic approach for adaptive multiagent control in heterarchical manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans 33(5), 573–588 (September 2003)
19. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE Transactions on Evolutionary Computation 5(4), 349–358 (August 2001)
20. Parekh, R., Honavar, V.: Grammar inference, automata induction, and language acquisition. In: Handbook of Natural Language Processing. pp. 727–764. Marcel Dekker (1998)
21. Rana, S.: The distributional biases of crossover operators. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 549–556. Morgan Kaufmann Publishers (1999)
22. Ricardo Aler, D.C., Moscardini, A.: The effects of transfer of global improvements in genetic programming. Computing and Informatics (formerly: Computers and Artificial Intelligence) 23(4), 377–394 (2004)
23. Sakakibara, Y.: Recent advances of grammatical inference. Theor. Comput. Sci. 185(1), 15–45 (1997)
24. Spears, W.M.: Crossover or mutation. In: Foundations of Genetic Algorithms 2. pp. 221–237. Morgan Kaufmann (1993)
25. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on Information Theory 23(3), 337–343 (1977), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1055714

David F. Barrero is a Lecturer at the Computer Engineering Department of the University of Alcal. He holds a Telecommunications Engineering degree and he recived a MSc. in Computer Science from the University of Alcal. Nowdays he is PhD candidate at the same University. He has previously spent six months in the Centre National d'Etudes Spatiales (CNES) in Toulouse, France. His research focuses on experimental methods in Evolutionary Computation and automatic learning of regular expressions.

Antonio González-Pardo holds a BSc in Computer Science from Universidad Carlos III de Madrid (2009). Nowadays, he is a Computer Science PhD candidate at Escuela Politcnica Superior (UAM) ¡http://www.ii.uam.es¿. Currently he is involved with ACIDA interest research group and GNB group ¡http://arantxa.ii.uam.es/main
research interests are related to Genetic Algorithms, Information Theory (compression-based metrics), information extraction (grammatical and evolutionary-based of regular expressions) and Signature Neural Networks.

David Camacho is an Associate Professor in the Computer Sciennce Department at Universidad Autnoma de Madrid (Spain). He received a Ph.D. in Computer Science (2001) from Universidad Carlos III de Madrid for his work on coordination of planning heterogeneous agents to solve problems with information gathered from the Web. He received a B.S. in Physics (1994) from Univesidad Complutense de Madrid. He has published over 50 journal, books, and conference papers. His research interests include Multi-Agent Systems, Distributed Artificial Intelligence, Web Service Technologies, Knowledge representation, Automated Planning and Machine Learning. He has also participated in several projects about automatic machine translation, optimising industry processes, MultiAgent technologies and Intelligent Systems. He is the managing editor of the International Journal of Computer Science & Applications (IJCSA), and has been selected as a chairman and member of the organizing committee for several international conferences.

María D. R-Moreno received a M.S. degree in Physics from the Universidad Complutense de Madrid (Spain) and her PhD in Computer Science from Universidad de Alcalá in Madrid (Spain). She has spent one year at NASA Ames Research Center as a postdoc, and nine weeks as a Research Visitor at ESA's European Space Research and Technology Centre (ESTEC).She is currently Associate Professor in the Departamento de Automatica at the Universidad de Alcala. She has previously worked as an Associate professor and as a consultant. She has publications in prestigious journal such as AI Magazine, Expert Systems with Application or IEEE Transactions on Knowledge and Data Engineering.

Dr. R-Moreno is actively collaborating in ESA projects and participating with research groups at NASA. She has served in the program committee of several international AI conferences and reviewer of the IEEE Transactions on Knowledge and Data Engineering journal. She is member of the Editorial Board of IJCSA and IAENG Journals, and member of the Experts Group of the Editorial Idea Group Inc. Her research focuses on Automated AI Planning & Scheduling, Monitoring and Execution applied to real applications (i.e. aerospace, e-learning or the web) and Genetic Programming.

Received: January 18, 2010; Accepted: May 3, 2010.

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

004

COMPUTER Science and Information Systems : the International journal / Editor-in-Chief Mirjana Ivanović. – Vol. 7, No 3 (2010) - . – Novi Sad (Trg D. Obradovića 3): ComSIS Consortium, 2010 - (Belgrade : Sigra star). –30 cm

Polugodišnje. – Tekst na engleskom jeziku

ISSN 1820-0214 = Computer Science and Information Systems
COBISS.SR-ID 112261644

Cover design: V. Štavljanin
Library Administration: M. Petković
Printed by: Sigra star, Beograd

ComSIS Computer Science and Information Systems

Volume 7, Number 3, June 2010

Contents

ISSN: 1820-0214

Computer Science and Information Systems

Vol 7, No 3, June 2010