Computer Science and Information Systems 11(4):1555–1580 DOI: 10.2298/CSIS131218073F

Semantic Representation of Multi-platform 3D Content

Jakub Flotyński, Krzysztof Walczak

Poznań University of Economics
Niepodległości 10, 61-875 Poznań, Poland
{flotynski, walczak}@kti.ue.poznan.pl
http://www.kti.ue.poznan.pl

Abstract. In this paper, a method of semantic representation of multi-platform 3D content is proposed. The use of the semantic web techniques enables content representation that is independent of particular content presentation platforms and may facilitate content creation based on different ontologies and knowledge bases. The proposed method significantly simplifies building 3D content presentations for multiple target platforms in comparison to the available approaches to 3D content creation.

Keywords: 3D web, semantic web, 3D content, multi-platform, ontology.

1. Introduction

Widespread use of interactive 3D web technologies—also on mobile devices—has been recently enabled by the significant progress in hardware performance, the rapid growth in the available network bandwidth as well as the availability of versatile input-output devices. 3D technologies become increasingly popular in various application domains, such as education, training, entertainment and social media, significantly enhancing possibilities of presentation and interaction with complex data and objects.

However, reaching large groups of recipients of 3D content on the web requires support for a diversity of hardware and software systems and—thereby—support for a multitude of available 3D content presentation platforms. Currently, wide coverage of different hardware and software systems by 3D content presentations is typically achieved by providing separate implementations for particular content browsers and presentation tools available on individual systems, which is generally problematic.

Conversely, compatibility of 3D content representations with various presentation environments could improve the reuse of common 3D content components and the overall use of 3D content. In such an approach, once 3D content is created, it can be presented using multiple platforms in different hardware and software systems. Moreover, such approach does not require users to install additional software, but it can leverage well-established 3D content browsers and presentation tools that may already be installed on the users' systems (e.g., Adobe Flash Player and WebGL or X3DOM-compliant web browsers). Although several standardization efforts have been undertaken in the field of 3D content representation, development of 3D platforms is driven by large industry players in a competitive environment and the issue of cross-compatibility of 3D content is still neglected, resulting in fragmentation of content and presentation technologies. This is an important obstacle preventing the mass use of 3D content technologies on the web.

Compatibility of 3D content with different presentation platforms can be achieved by the use of the semantic web standards for content modeling. The research on the semantic web aims at evolutionary development of the current web towards a distributed semantic database linking structured content and documents of various types, such as text, images, audio, video and 3D content. The semantic description of web content makes it understandable for both humans and computers achieving a new quality in building web applications that can ”understand” the meaning of particular components of content and services as well as their relationships, leading to much better methods of searching, reasoning, combining and presenting web content. However, the use of the semantic web standards is not only limited to content description, but may as well cover content creation.

The main contribution of this paper is a method of semantic representation of multi-platform 3D content. The method encompasses a semantic content model and a semantic content transformation that permit flexible and efficient creation of 3D content for a variety of target presentation platforms, including visualization tools, content representation languages and programming libraries. In the presented method, once the representation of 3D content is designed, it can be automatically transformed into different final presentation forms, which are suited to different 3D content presentation platforms. The selection of the target platforms to be used is an arbitrary decision of the system designer, and it does not affect the content design process. Referring to the semantics of particular 3D content components and conformance to the well-established semantic web standards enables 3D content representation that is independent of particular browsers and presentation tools, and permits reflection of complex dependencies and relations between content components. Moreover, the proposed method is intended for creating content that is to be further accessed and processed (indexed, searched and analyzed) by different content consumers on the web, as the content may be represented with common ontologies and knowledge bases.

The remainder of this paper is structured as follows. Section 2 provides an overview of selected approaches to semantic and multi-platform representation of 3D content. Section 3 introduces the new method of representing multi-platform 3D content. Section 4 outlines the implementation of the method. Section 5 explains an illustrative example of the creation of a multi-platform 3D presentation. Section 6 presents and discusses the results of the evaluation of the method. Finally, Section 7 concludes the paper and indicates the possible directions of future research.

2. State of the Art

The background of the proposed method covers issues related to both semantic modeling of 3D content and multi-platform 3D content presentation, which are combined in the proposed solution to enable efficient creation of multi-platform content presentations by using semantic web techniques. In this section, several works conducted in both fields are discussed.

2.1. Semantic Description and Modeling of 3D Content

Numerous works have been devoted to semantic description and semantic modeling of 3D content. The first group of works are mainly devoted to describing 3D content with semantic annotations to facilitate access to content properties. In [26], an approach to designing interoperable RDF-based Semantic Virtual Environments, with system-independent and machine-readable abstract descriptions has been presented. In [5, 6], a rule-based framework using MPEG-7 has been proposed for the adaptation of 3D content, e.g., geometry and texture degradation, and filtering of objects. Content can be described with different encoding formats (in particular X3D), and it is annotated with an indexing model. In [31], integration of X3D and OWL using scene-independent ontologies and the concept of semantic zones are proposed to enable querying 3D scenes at different levels of semantic detail.

The second group of works are devoted to modeling of different aspects of 3D content, including geometry, appearance and behavior. In [19], an ontology providing a set of elements and properties that are equivalent to elements and properties provided in X3D has been proposed. Moreover, a set of semantic properties for coupling VR scenes with domain knowledge has been introduced. Although the use of semantic concepts enables reasoning on the content created with the approach, the semantic conformance to X3D limits the possibilities of the exchange of entire content layers including different components and properties related to a common aspect of the modeled 3D content, e.g., appearance or behavior.

In [36], a method of creating interactive 3D content on the basis of reusable elements with specific roles, which enables 3D content design by non-IT-specialists has been proposed. However, the solution does not employ semantic web techniques, which could further facilitate content creation by domain experts using domain-specific ontologies and knowledge bases. In [7, 33, 34], an approach to generating virtual words upon mappings of domain ontologies to particular 3D content representation languages (e.g., X3D) has been considered. The solution stresses spatial relations (position and orientation) between objects in the scene. It enables mapping between domain-specific objects and 3D content components, but it does not address logically complex relationships between domain-specific concepts and 3D content components and properties, such as compositions of low-level content properties and relations between content components by high-level (e.g., domain-specific) elements (properties, individuals and classes) and combinations of such high-level elements.

Several works have been conducted on modeling behavior of VR objects. In [35], the Beh-VR approach and the VR-BML language have been proposed for the dynamic creation of behavior-rich interactive 3D content. The proposed solution aims at simplification of behavior programming for non-IT-specialists. However, the solution does not enable specifying and verifying semantics of particular content components, which limits its usage. Another method facilitating modeling of content behavior [28, 29] provides a means of expressing primitive and complex behaviors as well as temporal operators. Tool--supported design approach to defining object behavior in X3D scenes has been presented in [30]. Finally, a rule-based ontology framework for feature modeling and consistency checking has been explained in [40]. This ontology-based approach addresses mainly modeling of elementary content animations.

The third group includes works devoted to the use of semantic descriptions of 3D content in artificial intelligence systems. The idea of semantic description of 3D worlds has been summarized in [21]. In [4], diverse issues arising from combining AI and virtual environments have been reviewed. In [8, 23], abstract semantic representations of events and actions in AI simulators have been presented. In [20, 22, 39], a technique of integration of knowledge into VR applications, a framework for decoupling components in real-time intelligent interactive systems with ontologies and a concept of semantic entities in VR applications have been discussed.

The aforementioned approaches address different aspects of semantic description and semantic creation of 3D content, but they lack general solutions for comprehensive conceptual modeling of 3D content, its components, properties and relations, at an arbitrarily high (conceptual) level of semantic abstraction, which is a key to enable content modeling by domain experts with the use of domain ontologies.

2.2. Multi-platform Presentation of 3D Content

Several works have been devoted to 3D content presentation across different hardware and software platforms. In [24], a specific 3D browser plug-in for different web browsers has been described. In [9], an approach to multi-platform 3D content presentation based on MPEG-4 has been explained. In [2], an approach to multi-platform visualization of 2D and 3D tourism information has been presented. In [32], an approach to adaptation of 3D content complexity with respect to the available resources has been proposed. In [16], the architecture of an on-line game with 3D game engines and a multi-platform game server has been presented. In [18], an approach to integrated information spaces combining hypertext and 3D content have been proposed to enable dual-mode user interfaces, embedding 3D scenes in hypertext and immersing hypertextual annotations into 3D scenes—that can be presented on multiple platforms on the web. In [12], an approach to building multi-platform virtual museum exhibitions has been proposed.

The aforementioned works cover the development of 3D content presentation tools and environments as well as contextual platform-dependent content adaptation. However, they do not address comprehensive and generic methods of content transformation to enable creation of multi-platform 3D content presentations.

3. Method of Multi-platform 3D Content Representation

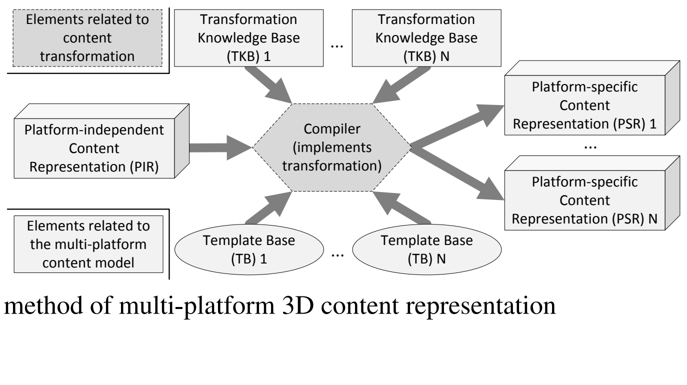
Although several solutions have been proposed for creating multi-platform 3D content representations (cf. Section 2.2), they do not enable general and flexible transformation of 3D content into different content representation languages, to present it with various content browsers and presentation tools. On the one hand, the available approaches to multi-platform content presentation focus mainly on particular use cases (e.g., tourism information), tools (e.g., web browsers) and formats (e.g., MPEG-4). On the other hand, the available parsers and plug-ins to 3D modeling tools are specific to particular input and output formats of content, and they do not provide comprehensive and generic solutions for generating platform-independent content representations, in particular when the content must be created ad-hoc or adapted to a specific context of use.

In this paper, a method of multi-platform 3D content representation is proposed. The method enables generation of 3D content for a variety of content presentation platforms. The presented solution leverages semantic web techniques to provide a generic 3D content representation that is platform-independent and to enable flexible description of transformation of content representations. The method includes two elements: the *Multi-Platform Semantic 3D Content Model* and the *Semantic Transformation* of generic platform-independent content representation to final platform-specific content representations, which can be presented using diverse content presentation tools. The paper describes both the model and the transformation. These elements are used in combination to provide a comprehensive method for creating multi-platform 3D presentations.

The proposed method is a part of the SEMIC (*Semantic Modeling of Interactive 3D Content*) approach, which aims at automation and simplification of the 3D content creation process [14]. A key element of SEMIC is the *Semantic Content Model* (SCM) [10, 11, 13]. SCM is a collection of ontologies that enable platform-independent semantic representation of 3D content at different levels of abstraction—low-level *concrete content representation*, which reflects elements that are directly related to 3D content, and arbitrarily high-level *conceptual content representation*, which reflects elements that are abstract in the sense of their final representation and not directly related to 3D content. The representations are linked by semantic *representation mappings*.

Concrete content representations conform to the *Multi-layered Semantic Content Model* (ML-SCM – proposed in [10]), which is a part of SCM. ML-SCM represents 3D content (objects or scenes) using semantic concepts, which are specific to 3D modeling and which are grouped into several partly dependent layers—geometry layer, structure layer, appearance layer, scene layer, animation layer and behavior layer—enabling separation of concerns between distinct aspects of 3D content modeling. The model encompasses concepts (classes and properties) widely used in well-established 3D content representation languages and programming libraries, such as X3D, VRML, Java3D and Away3D.

An outline of the proposed method of multi-platform 3D content representation is presented in Fig. 1. *Platform-independent 3D content representations* (PIRs) are semantic knowledge bases, which conform to ML-SCM. PIRs are processed by a *compiler* that implements semantic transformation of PIRs to *platform-specific 3D content representations* (PSRs). PSRs are documents encoded with arbitrarily selected 3D content representation languages, thus PSRs may be presented using various 3D content browsers and presentation tools. A transformation of PIRs to PSRs, which need to be compatible with a particular content presentation platform, is performed with regards to a *transformation knowledge base* (TKB) and a *template base* (TB) that have been designed for this platform. A TKB describes transformation rules that allow for the creation of final PSRs based on primary PIRs. A TB is a set of parametrized *templates* (fragments of code) of a content representation language, which are combined during content transformation to produce a final PSR (encoded in the language). In the proposed method, once a PIR is created, it may be automatically transformed to PSRs, which are presentable on different platforms, for which appropriate TKBs and TBs have been developed.



Elements related to content transformation
Transformation Knowledge Base (TKB) 1
...
Transformation Knowledge Base (TKB) N
Platform-specific Content Representation (PSR) 1
Platform-independent Content Representation (PIR)
Compiler (implements transformation)
...
Platform-specific Content Representation (PSR) N
Elements related to the multi-platform content model
Template Base (TB) 1
...
Template Base (TB) N

Fig. 1. The method of multi-platform 3D content representation

The following two subsections describe the proposed method in detail. First, the Multi-Platform Semantic 3D Content Model is presented, then the Semantic Transformation (which is implemented in the *compiler*) is explained.

3.1. Multi-Platform Semantic 3D Content Model

The *Multi-Platform Semantic 3D Content Model* (MP-SCM) (Fig. 2) extends the concepts proposed in [10, 13, 15] and enables representation of 3D content that may be presented using various content browsers and presentation tools. While ML-SCM provides components and properties that are related to various aspects of 3D content, MP-SCM provides data structures that enable transformation of content representations into different encoding formats. The model consists of four parts—*platform-independent content representations* (PIRs), *transformation knowledge bases* (TKBs), *template bases* (TBs) and *platform-specific content representations* (PSRs). The particular elements of the model are described in the following subsections.

Transformation knowledge bases. A *transformation knowledge base* (TKB) is the primary entity of MP-SCM, which is responsible for semantic transformation of content. A TKB incorporates transformation rules that allow for transformation of PIRs to the corresponding PSRs, which are to be presented on a common target platform. An individual TKB is created for a particular presentation platform or a group of presentation platforms that use a common content representation language or different languages that have equivalent structures of documents, thereby enabling the use of common transformation rules. Since the proposed method is generic and based on elementary operations on code, TKBs for different—either declarative (e.g., VRML, X3D) or imperative (e.g., ActionScript, Java)—content representation languages may be introduced.

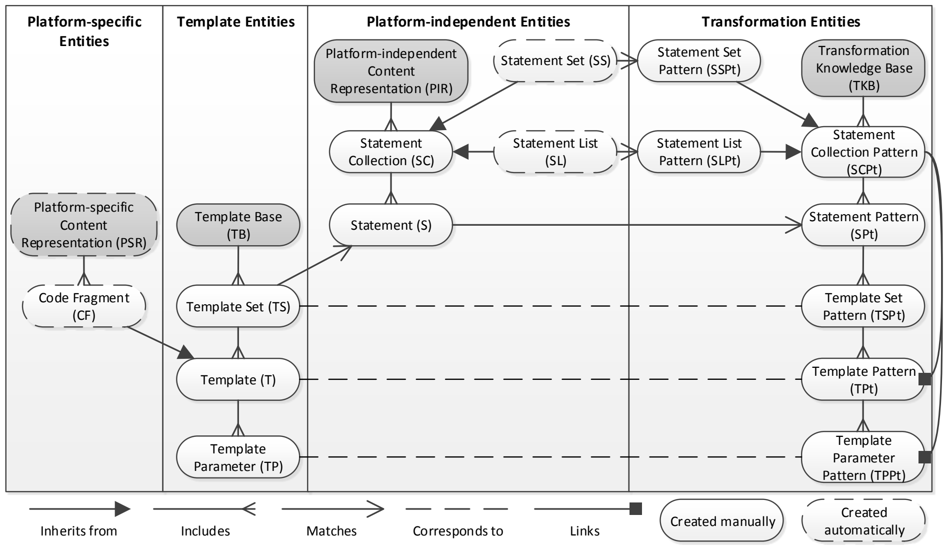

Platform-specific Entities
Template Entities
Platform-independent Entities
Transformation Entities
Platform-independent Content Representation (PIR)
Statement Set (SS)
Statement Set Pattern (SSPt)
Transformation Knowledge Base (TKB)
Statement Collection (SC)
Statement List (SL)
Statement List Pattern (SLPt)
Statement Collection Pattern (SCPt)
Platform-specific Content Representation (PSR)
Template Base (TB)
Statement (S)
Statement Pattern (SPt)
Code Fragment (CF)
Template Set (TS)
Template Set Pattern (TSPt)
Template (T)
Template Pattern (TPt)
Template Parameter (TP)
Template Parameter Pattern (TPPt)
Inherits from
Includes
Matches
Corresponds to
Links
Created manually
Created automatically

Fig. 2. The Multi-Platform Semantic 3D Content Model (MP-SCM)

The primary entity of a TKB is a *statement pattern* (SPt). Every SPt is a semantic pattern that matches a group of possible *statements* (Ss) in a PIR. Matching a group of Ss by an SPt is enabled by semantic generalization. A generalization may pertain to the subject, the property or the object of Ss. For instance, a possible generalization (an SPt) of the S `[object scm:color "red"]` in terms of property, is the S `[object scm:appearanceProperty "red"]`, while a possible generalization of the S `[object rdf:type scm:Mesh3D]` in terms of object, is the S `[object rdf:type scm:GeometricalComponent]`.

SPts include entities that semantically determine the structure of code that is selected for Ss during the transformation. An SPt may include a *template set pattern* (TSPt) with a number of *template patterns* (TPts), each of which may contain a number of *template parameter patterns* (TPPts). These concepts correspond to the entities of the parametrized code, which are used for generation of PSRs: *template sets* (TSs), *templates* (Ts) and *template parameters* (TPs), respectively. However, they do not explicitly indicate any platform-specific entities. In a TKB, the level of generality of SPts may be high to cover a wide range of Ss in a PIR. For instance, the SPt `[?subject scm:dataProperty ?value.]` may cover the Ss `[?subject scm:intensity "10".]` and `[?subject scm:color "red".]`. In a PSR, each of these Ss needs to be represented by a different T. Hence, the selection of a T can be done only for a particular S given. However, the general structure of both Ts may be known in advance and specified at the semantically generalized level (in a TKB). For instance, the aforementioned Ts may be specified as `[$object.intensity = $data]` and `[$object.color = $data]`, thus having the same `object` and `data` TPs.

Since transformation of content representations is performed on *templates* (Ts), which are fragments of code associated with *statements* (Ss), which form PIRs, the following elementary operations need to be performed on Ts:

(1) setting common values of TPs,
(2) nesting Ts into other Ts,
(3) ordering Ts.

The operations are enabled by *statement collection patterns* (SCPts): operations (1) and (2)—by *statement set patterns* (SSPts), while operation (3)—by *statement list patterns* (SLPts). Every SCPt includes multiple SPts. As an SPt matches a single S, an SCPt matches a group of Ss in a PIR.

For instance, the SLPt `[?subject rdf:type ?type. ?subject ?property ?value.]` matches the pair of Ss `[?light scm:intensity "10". ?light rdf:type scm:DirectionalLight.]`. A common subject (the `light` parameter) and the reverse order needs to be set for an imperative final content representation language (e.g., ActionScript), while the T of the first S needs to be nested into the T of the second S for a declarative final content representation language (e.g., X3D). The resulting example imperative code is `[DirectionalLight light = new DirectionalLight(); light.intensity = 10;]`. To enable linking TPs and nesting Ts between different Ss of an SCPt, appropriate semantic statements are specified for TPPts and TPts (which reflect TPs and Ts) at the level of the SCPt.

Platform-independent content representations. A *platform-independent content representation* (PIR) is a knowledge base that conforms to ML-SCM, thus it is presentation platform agnostic. However, PIRs can be presented on different platforms when transformed with appropriate TKBs and TBs. A PIR includes semantic 3D content components, which are described by semantic 3D content properties. Both the components and the properties reflect different aspects of 3D content, such as geometry, structure, appearance, scene, animation and behavior.

The primary entity of a PIR, in terms of the semantic content transformation, is a *statement* (S), which is matched by a SPt in a TKB. During the transformation, Ss are gathered into *statement collections* (SCs), which are dynamically created by the *compiler*. Assembling Ss into SCs allows to perform the elementary operations (setting TPs, nesting Ts and ordering Ts) on Ts that are associated with the Ss. SCs are assembled from Ss that match appropriate SCPts, which are declared in the TKB. Two types of SCs are distinguished: *statement sets* (SSs) and *statement lists* (SLs), which are built with respect to SSPts and SLPts, respectively.

Template bases. A *template base* (TB) is a set of parametrized fragments of code that may be combined to create PSRs on the basis of PIRs. The primary entity of a TB is a *template set* (TS), which may include a number of *templates* (Ts), which are parametrized fragments of code—may include a number of *template parameters* (TPs). TBs may leverage various 3D content representation languages (e.g., X3D, Java), programming libraries (e.g., Java3D, Away3D) or game engines (e.g., Unity, Unreal). In the proposed method, every S in a PIR may be linked with a TS. Linking is dynamic and based on the actual *signature* of the S processed, which is a triple `[class of the subject, property, class of the object]`. For every S, the TS whose *signature* matches the S is selected. Since the selection of a TS requires a particular S, the link between TSs and Ss cannot be given in a TKB on the level of SPts.

Every TS may include a number of Ts that need to be individually processed, e.g., injected into different TPs of a parent T. The processing of particular Ts, which are included is a TS, is specified in the TKB. For instance, for a presentation platform that uses an imperative content representation language without navigation implemented, it may be necessary to inject a T implementing proper functions next to the `main` function and to inject another T switching on the functions within the `main` function.

An individual TB and its corresponding TKB are created once every time a new presentation platform that uses a new content representation language is added to the system. Since a TB and a TKB are introduced, they may be used for the development of various 3D presentations that are presentable on the new platform.

Platform-specific content representations. A *platform-specific representation* (PSR) is a set or a sequence of instructions encoded in a selected content representation language. A PSR is generated automatically by the *compiler*, and it is a composition of *code fragments* (CFs), which are automatically created upon Ts by setting the values of their TPs. Once created, a PSR may be presented on the selected 3D content presentation platform.

3.2. Semantic Transformation of Content Representations

In the proposed method, PIRs are transformed to PSRs by a transformation algorithm (TA). The general idea of the TA is based on the elementary operations on Ts, which have been introduced in Section 3.1 and are presented in detail below:

1) *setting common values of TPs* that are associated with different Ss included in common SSs,

2) *nesting Ts into other Ts* that are associated with different Ss included in common SSs,
3) *ordering Ts* that are associated with different Ss included in common SLs.

The operations are performed regarding the description of the transformation, which is included in the appropriate TKB (the TKB associated with the selected target content representation language). Due to the use of only basic operations on fragments of code, the proposed method is generic and it can be applied to specify transformations for various target content representation languages, which may conform to either imperative or declarative programming paradigms. The TA has five phases, each of which comprises several steps. First, semantic queries are created on the basis of SSPts specified in the TKB. Second, the queries are issued against a PIR to create SSs. Third, the values of TPs associated with the Ss that are included in the SSs are set. Next, Ts associated with the Ss are nested one into another. Finally, Ts are enumerated in a specific order. For each phase, its computational complexity has been determined. The following notation has been used:

- N(SSPt) – the total number of SSPts in the TKB,
- N(SLPt) – the total number of SLPts in the TKB,
- N(SPt) – the total number of SPts in the TKB,
- N(S) – the total number of Ss in the PIR,
- N(SPt/SSPt) – the average number of SPts per SSPt,
- N(TP/S) – the average number of TPs per S,
- N(T/S) – the average number of Ts per S.

Creation of queries. In this phase, semantic queries to knowledge bases (PIRs) are created on the basis of the appropriate TKB (that is specified for the selected 3D content representation language). Each query corresponds to an individual SSPt, as its clauses (triples `[subject, property, object]`) correspond to particular SPts that are included in the SSPt. The following step is performed in this phase:

1. For every SSPt from the TKB, create a query (e.g., in the SPARQL query language) as follows:
   (a) Create an empty query,
   (b) For every SPt from the SSPt, which is given as `[?subject ?property ?object.]`:
      i. Append the following clause (given in the pseudo code) to the query `[?prop rdfs:subPropertyOf ?property.]`,
      ii. If the `object` of the SPt is a literal or an individual, append the following clause to the query `[?subject ?prop ?object.]`,
      iii. If the `object` of the SPt is a class, append the following clauses to the query `[?subject ?prop ?obj. ?obj rdfs:subClassOf ?object.]`.

Inserting the `rdfs:subPropertyOf` and the `rdfs:subClassOf` properties to a query enables not only the use of this query to search for Ss of a PIR that exactly match the SPt (its `property` and its `object`), but also to search for Ss that use `sub-properties` and `sub-classes` of the `property` and the `class` that are specified in the SPt.

In this phase, every SPt occurring in an SSPt is processed once to be included in a query. Therefore, the computational complexity of this phase is polynomial and it is equal to `O(N(SSPt)*N(SPt/SSPt))`.

Creation of statement sets. In this phase, the queries created in Step 1 are issued against a PIR to create SSs. Each resulting SS is a group of logically related Ss. The following step is performed in this phase:

2. For every query created in Step 1:
   (a) Issue the query against the PIR and create an SS incorporating all the Ss from the PIR that are included in the result of the query,
   (b) For every S, remember the SPt from the query that has been satisfied by the S.

The SSs created in this step will be further used for setting some common TPs for different Ts and for nesting some Ts into other Ts.

Creation of SSs requires, in the worst case, for every SSPt (which determines a query), checking every SPt included in this SSPt against every S, which is included in the PIR. Therefore, the computational complexity of this phase is polynomial and it is equal to `O(N(SSPt)*N(SPt/SSPt)*N(S))`.

Setting template parameters. In this phase, TPs of logically related Ts are fixed regarding semantic dependencies between the Ss, the Ts are associated with. The following step is performed in this phase:

3. For every SS created in Step 2:
   (a) For every S, which is given as `[?subject ?property ?object]` and which is included in the SS:
      i. Load a TS whose *signature* matches the semantic pattern determined by the S—the concatenation of the name of the class, the `subject` belongs to, the name of the `property` and the name of the class, the `object` belongs to.
      ii. Query the TKB of TPPts that are included in the TPts that are included in the TSPt linked to the SPt that is associated with the S processed
      `[?TPPt tkb:isParameterOf ?TPt. ?TPt tkb:isIn ?TSPt. ?TSPt tkb:isTemplateSetOf ?SPt.]`.
      Remember the *template parameter pattern set* (TPPtS) associated with the S.
      iii. For every TPPt1 that is included in the TPPtS and that has not yet been set:
         A. Set a unique value of the TPPt1,
         B. If the TPPt1 is a literal parameter of the S, replace the TP that is reflected by the TPPt1 and is included in a T with the literal of the S,
         C. Query the TKB of TPPts that are equal to the processed TPPt1 and that are linked to SPts that occur in a common SSPt with the SPt that is associated with the S
         `[?TPPt1 tkb:equalTo ?TPPt2. ?TPPt2 tkb:isParameterOf ?TPt. ?TPt tkb:isIn ?TS. ?TS tkb:isTemplateSetOf ?SPt2. ?SPt2 tkb:isIncludedIn ?SSPt. ?SPt tkb:isIncludedIn ?SSPt.]`,
         D. For every TPPt2 found, set the value of the reflected TP (which is included in a T) to the value of the primary TPPt1 processed and recursively go to Step 3(a)iiiC. A T whose all TPs are set is a CF.

As the result of this phase, Ts have TPs set to proper values (literals) and common identifiers of variables. Recursive processing of TPs ensures assigning a common value to all equal TPs across different Ts.

In this phase, every TP must be set. Therefore, the computational complexity of this phase is polynomial and it is equal to `O(N(S)*N(TP/S))`.

Nesting platform-specific templates. In this phase, Ts are nested one into another creating a hierarchical structure. Nesting is indicated by assigning Ts to TPs. The following step is performed in this phase:

4. For every SS created in Step 2:
   (a) For every S that is included in the SS and that has not yet been processed:
      i. For every TPPt from the TPPtS (determined in Step 3(a)ii) that is associated with the SPt associated with the S:
         A. Query the TKB of TPts that are equal to the TPPt and that are assigned to SPts included in a common SSPt together with the SPt of the S `[?TPPt tkb:equalTo ?TPt. ?TPt tkb:isIn ?TS. ?TS tkb:isTemplateSetOf ?SPt2. ?SPt2 tkb:isIncludedIn ?SSPt. ?SPt tkb:isIncludedIn ?SSPt.]`,
         B. For every TPt found, go recursively to Step 4(a)iA,
         C. Replace all the TPs that are reflected by the TPPts with the appropriate Ts that are reflected by the TPts and produce CFs.

Like the previous phase, nesting Ts is performed recursively—processing of a T continues until all Ts that are to be nested in the T are processed (its appropriate TPs are set to appropriate Ts).

In this phase, in the worst case, every T must be nested into another T by a TP. Therefore, the computational complexity of this phase is polynomial and it is equal to `O(N(S)*N(T/S))`.

Ordering platform-specific templates. In this phase, Ts linked with mutually dependent Ss, that need to occur in the final PSR at the same level (without nesting one T into another) are set in the PSR in a suitable order. In this phase, the global list (sequence) of SPts is created and the Ss of the PIR are sorted with respect to this list. The following steps are performed in this phase:

5. Create the *global SLPt* list. Perform the steps until all SPts that are included in different SLPts are included in the *global SLPt*:
   (a) If the SPt is not yet included in the *global SLPt*, but it is included in any SLPts and there are no other SPts that precede the SPt in at least one SLPt, add the SPt to the end of the *global SLPt*.
6. Add the remaining SPts (the SPts that are not included in any SLPts) to the end of the *global SLPt* in an arbitrary order.
7. Browse the *global SLPt* in the reverse order and for every SPt included in the list:
   (a) Find Ss in the PIR that match the SPt and add their CFs (generated in the previous phases) at the beginning of the generated PSR.

Creation of a *global SLPt* requires, in the worst case, checking every SPt against every SLPt for all SPts included in the TKB. Therefore, the computational complexity of Step 5 is equal to `O(N(SLPt)*N(SPt)*N(SPt))`. Adding Ts to the generated PSR requires processing of every S for every SPt. Therefore, the computational complexity of Step 7 is `O(N(SPt)*N(S))` and the overall computational complexity of this phase is polynomial. Since the complexities of the consequent phases of the TA are polynomial, the overall computational complexity of the proposed TA is also polynomial.

4. Implementation

Implementation of the proposed method is discussed in terms of final content representation, the description of transformation rules and the transformation software.

4.1. Final 3D Content Representation

The following languages have been selected for final content representation: ActionScript with the Away3D library, VRML, and X3D with XML encoding. 3D content representations encoded in ActionScript are presented using Adobe Flash Player, while representations encoded in VRML and X3D are presented using VRML and X3D browsers, e.g., Cortona3D and Bitmanagement BS Contact, respectively. The languages have been chosen because of the following two reasons. First, they are implemented by a wide range of tools for 3D content presentation. Second, the languages differ in the syntaxes and the approaches to 3D content representation. ActionScript is an imperative language, which permits specification of steps to be accomplished to obtain desirable presentational effects, whereas VRML and X3D are declarative languages, which permit direct specification of desirable presentational effects to be obtained, without specifying steps that must be performed to achieve the effects. Covering different syntaxes and programming paradigms allows for a more thorough evaluation of the proposed method.

4.2. Transformation Description

TKBs for the languages have been implemented using the semantic web standards—the Resource Description Framework (RDF), the Resource Description Framework Schema (RDFS) and the Web Ontology Language (OWL). Similar schemes of VRML and X3D documents have allowed for the development of a common TKB for these languages, while a separate TKB has been developed for ActionScript. SSPts and SLPts are encoded as RDF `bags` and RDF `sequences`, respectively, while the SPts are encoded using RDF reification. TSPts, TPt and TPPts are encoded as instances of appropriate OWL classes and they are linked by OWL properties. Ts, which are included in TBs, are parameterized documents. The *signature* of a T is specified by its name. Ts with the same *signatures* create a common TS. TPs, which occur within Ts, are indicated by specific symbols. The implemented TKBs and TBs cover the main elements of ML-SCM, such as shapes and meshes (*geometry layer*), groups of objects, size, position and orientation (*structure layer*), textures, materials and light sources (*appearance layer*) as well as navigation (*scene layer*). The conformance of the TKBs to the semantic web standards enables transformation of PIRs with a semantic query language (e.g., SPARQL). This permits, in contrast to a typical grammar analysis, processing of PIRs with regards to complex semantic dependencies between particular Ss of the PIR.

4.3. Compiler

The *compiler*, which is an implementation of the TA, has been developed. The *compiler* performs transformation of PIRs, which are compatible with ML-SCM, to PSRs, which are compatible with the selected content representation languages. The *compiler* has been written in Java and it leverages additional software libraries. The Pellet OWL reasoner [27] is used in Step 1 of the Semantic Transformation to discover implicit Ss, which have not been explicitly specified in the processed PIRs, but may be inferred and are necessary for building complete final PSRs. The Apache Jena SPARQL engine [3] is used in the next steps of the transformation to execute queries to PIRs and TKBs when discovering SSs, SLs as well as links between TPts and TPPts.

5. An Example of Building a Multi-platform 3D Presentation

An example of a PIR representation of a scene, which is encoded the ML-SCM/RDF-Turtle format, was created and transformed to PSRs, which are encoded in VRML, X3D/XML and ActionScript (Fig. 3-5). The particular representations are described in Fig. 6. Equivalent fragments of code of the representations are marked with common numbers. Some content components and properties, which are not crucial for the presented example, have been omitted in the representations.

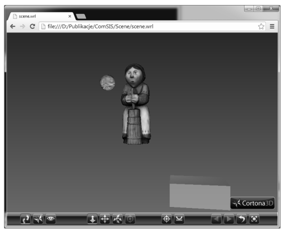


Fig. 3. The VRML representation presented in Cortona3D

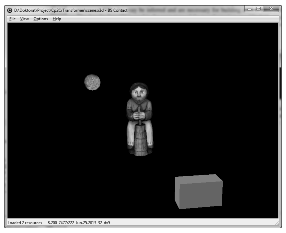


Fig. 4. The X3D representation presented in Bitmanagement BS Contact

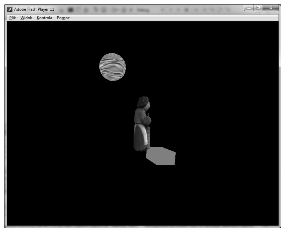


Fig. 5. The ActionScript representation presented in Adobe Flash Player

The `scene` includes a `PointLightSource` and three geometrical components: a `cube`, a `sphere` and a `mesh`. The components are the elements of some `StructuralComponents`—the `sphere` and the `cube` are included in the `complexObject`, while the `complexObject`, the `mesh` and the `light` are directly included in the `scene`. Since the components are assembled into `StructuralComponents`, they have spatial properties (e.g., `position`) specified. In addition, materials with some properties and components describing appearance (e.g., colors, textures) are assigned to the `cube`, the `sphere` and the `mesh`.

The PIR was transformed using two TKBs. In Fig. 7, an example of an SLPt, which is included in the TKB for the Adobe Flash Player presentation platform, is presented. Equivalent fragments of code are marked in gray. The SLPt includes two SPts. The `subject_type_component` SPt specifies the pattern that matches each S that is a declaration of an object in the created scene.

The `subject_type_component` SPt is to be matched by Ss whose `subject` is any semantic individual, `predicate` is a sub-property of the `rdf:type` property and `object` is a sub-class of the `scm:Component`. Every S that matches the SPt is required to be linked to a single TS with a single T that includes a single TP with the name equal to `"obj"`. The `subject_dataproperty_value` SPt specifies the pattern that matches each S that is an assignment of a literal value to a property of an object in the scene. The

ML-SCM Representation

```
1) ex:scene
       a scm:Scene ;
       scm:includes ex:complexObject ,
         ex:mesh , ex:light.
2) ex:complexObject
       a  scm:StructuralComponent;
       scm:includes ex:sphere , ex:cube .
3) ex:cube
       a scm:Box ;
       scm:material ex:cube_material ;
       scm:position ex:cube_pos .
   ex:cube_pos
       a scm:PosVector ;
       scm:isPositionOf ex:cube ;
       scm:x   "5.0"^^xsd:float ;
       scm:y   "5.0"^^xsd:float ;
       scm:z   "5.0"^^xsd:float .
4) ex:sphere
       a scm:Sphere ;
       scm:material ex:sphere_material.
   ex:sphere_material
      a scm:TextureMaterial ;
      scm:diffuseMap "texture.jpg".
5) ex:mesh
       a scm:Mesh3D ;
       scm:meshData "statue.obj" ;
       scm:isPartOf ex:scene.
6) ex:light
       a  scm:PointLightSource ;
       scm:intensity "10.0"^^xsd:float .
```

ActionScript Representation

```
package { //imports public class Main extends Sprite {
    private    var view: View3D;
1)   private    var var2:Scene3D
5)   private    var single1: Loader3D;
    private    var single3: Mesh;
    [Embed(source = "texture.jpg")]
    public static    var df52: Class;
    [Embed(source = "statue.obj", mimeType="...")]
    private static    var single2: Class;
    private function single4(event: LoaderEvent): void {
      single3 = Mesh(single1.getChildAt(0));
      var2.addChild(single3); }
    public function Main(): void {
1)     var2 = view.scene;
4)     var var5: Mesh = new Mesh(new SphereGeometry());
3)     var var3: Mesh = new Mesh(new CubeGeometry());
4)     var var4: TextureMaterial = new TextureMaterial();
6)     var var1: PointLight = new PointLight();
2)     var var0: ObjectContainer3D=new ObjectContainer3D();
3)     var3.x = -200.0; var3.y = -200.0; var3.z = 200.0;
6)     var1.brightness = 10.0;
2)     var0.addChild(var5);
1)     var2.addChild(var0);
2)     var0.addChild(var3);
4)     var5.material = var4;
1)     var2.addChild(var1);
4)     var tx52720:BitmapTexture=Cast.bitmapTexture(df52);
      var4.texture = tx52720;
5)     //3D mesh loader initialization    }  }}
```

VRML Representation

```
2)Transform {
   children[
3)  Transform {
      translation 5.0 - 5.0 5.0
      children[
        Shape {
          appearance Appearance {
            material Material {
              transparency 0.5 }}
          geometry Box {}}]}
4)  Transform {
      children[
        Shape {
          appearance Appearance {
            texture ImageTexture {
              url["weave_normal.jpg"]  }
            material Material {}}
          geometry Sphere {}}]}]}
5)  Transform {
    children[
      Shape {
        appearance Appearance {
          texture ImageTexture {
            url "…"}}
        geometry IndexedFaceSet {
          coord Coordinate {
            point[...]        }
          texCoord TextureCoordinate {
            point[...]        }
          coordIndex[...]
          texCoordIndex[...]}}]}
6) DEF light103717750 PointLight {
     intensity 10.0 }
```

X3D Representation

```
<X3D …> <head></head>
1)  <Scene>
2)    <Transform>
5)      <Shape>
          <Appearance>
            <Material/>
            <ImageTexture url="…"/>
          </Appearance>
          <IndexedFaceSet coordIndex="..."
            texCoordIndex="...">
            <Coordinate point="..."/>
            <TextureCoordinate point="..."/>
          </IndexedFaceSet>
        </Shape>
      </Transform>
4)    <Transform >
        <Transform>
          <Shape>
            <Appearance>
              <ImageTexture url=' "texture.jpg" '/>
              <Material />
            </Appearance>
            <Sphere /></Shape>
        </Transform>
3)      <Transform  translation=' 5.0  5.0  5.0' >
          <Shape>
            <Appearance><Material/></Appearance>
            <Box /></Shape>
        </Transform>
2)    </Transform>
6)    <PointLight intensity='10.0' />
1)  </Scene>
</X3D>
```

Fig. 6. Content representations encoded in the selected languages

`subject_dataproperty_value` SPt is to be matched by Ss whose `subject` is any semantic individual, `predicate` is a sub-property of the `scm:DataProperty` and `object` is any literal value. Every S that matches the SPt is required to be linked to a single TS with a single T that includes two TPs—a TP with the name equal to `"obj"` and

Action Script Transfor-mation Ontology	SLP a rdf:Seq ;
	  rdf:first tkb:subject_type_component ;
	  rdf:rest
	(tkb:subject_dataproperty_value).
	
	tkb:subject_type_component a rdf:Statement;
	  rdf:object scm:Component ;
	  rdf:predicate rdf:type ;
	  rdf:subject tkb:variable ;
	  tkb:hasTemplateSetPattern tkb:tsp1 .
	
	tkb:tsp1 a tkb:TemplateSetPattern ;
	  tkb:hasTemplatePattern tkb:tp1 .
	
	tkb:tp1 a tkb:TemplatePattern ;
	  tkb:hasTemplateParameter tkb:obj_par1 .
	
	tkb:obj_par1 a tkb:TemplateParameter ;
	  tkb:name "obj" .
		tkb:subject_dataproperty_value
		  a rdf:Statement ;
		  rdf:object "value" ;
		  rdf:predicate scm:DataProperty ;
		  rdf:subject tkb:variable ;
		  tkb:hasTemplateSetPattern tkb:tsp2 .
		
		tkb:tsp2 a tkb:TemplateSetPattern ;
		  tkb:hasTemplatePattern tkb:tp2 .
		
		tkb:tp2 a tkb:TemplatePattern ;
		  tkb:hasTemplateParameter tkb:obj_par2;
		  tkb:hasTemplateParameter tkb:data_par.
		
		tkb:obj_par2 a tkb:TemplateParameter;
		  tkb:name "obj" ;
		  tkb:equalTo tkb:obj_par1 .
		
		tkb:data_par a tkb:TemplateParameter;
		  tkb:name "value" ;
		  tkb:isLiteral "true" .

Query	SELECT DISTINCT * WHERE {
	 ?subject ?prop1 ?obj1. ?prop1 rdfs:subPropertyOf* rdf:type. ?obj1 rdfs:subClassOf* scm:Component.
	 ?subject ?prop2 ?value. ?prop2 rdfs:subPropertyOf* scm:DataProperty. }	

Template Sets	Signature: rdf:type-scm:PointLight var $obj:PointLight = new PointLight();	Signature: scm:intensity-value $obj.brightness = $value;

Fig. 7. A fragment of the ActionScript TKB, a generated query and template sets

a TP with the name equal to `"value"`. While the value of the `data_par` TP is the literal retrieved from the S that is dynamically associated with the SPt during the transformation, the value of the `obj_par2` is set to the value of the `obj_par1` TP.

On the basis of the SLPt, a semantic query to a PIR is generated by the *compiler*. The triples included in the query correspond exactly to the SPts included in the SLPt. During the transformation of a PIR, every pair of Ss that matches the SLPt is dynamically combined into a new SS. Next, TSs are selected for the Ss regarding the equality of *signatures* of TSs and Ss. For instance, the pair of Ss `[light scm:intensity "10". light rdf:type scm:PointLight.]` matches the SLPt and the *signatures* of TSs that are given in this example. The TPs of the Ss are set as indicated by the SLPt and, in the resulting ActionScript PSR, the Ss are enumerated in the reverse order, which is specified by the sequence of the Ss inclusion in the SLPt. Hence, the result of the transformation is as follows: `[var light:PointLight = new PointLight(); light.brightness = 10;]`.

6. Evaluation

The proposed solution has been evaluated in terms of the complexity of PIRs (semantic content representations) and PSRs (final content representations), profits and costs of the multi-platform method, and the efficiency of the implemented TA. The evaluation covers primary PIRs, which are created by a content developer, and secondary PSRs, which are generated automatically by the implemented *compiler*. PIRs are encoded in the ML-SCM/RDF-Turtle format, while PSRs are encoded in the VRML, X3D/XML and ActionScript languages.

The evaluation has been carried out for PIRs consisting of various numbers of content components (related to geometry, structure, space and appearance) assembled into scenes. The number of components has varied over the range of 5 to 50 with the step equal to 5. For every number of components, 20 random scenes have been generated and average results have been calculated. For each component, randomly selected properties have been set to random values. The test environment used was equipped with the AMD Athlon II X3 445 3.1 GHz processor, 4 GB RAM and the Windows 7 OS.

6.1. Complexity of Content Representations

The complexity of 3D content representations has been evaluated with the following metrics: the Structured Document Complexity Metric [25], the number of bytes, the number of logical lines of code (LLOC) [1] and the Halstead Metrics [17]. While the first metric measures the complexity of representation schemes (for both PIRs and PSRs), the other metrics measure the complexity of particular PIRs and PSRs.

Structured Document Complexity Metric. The Structured Document Complexity Metric has been used to measure the complexity of representation schemes regarding unique elements and attributes, required elements and attributes as well as attributes that need to be specified at the first position within their parent elements. The metric may be calculated for XML- and grammar-based documents. The values of the Structured Document Complexity Metric that have been calculated for the VRML, X3D, ActionScript and ML-SCM representation schemes, are presented in Table 1.

Criterion	VRML	X3D	ActionScript	ML-SCM
Unique elements	15	15	24	24
Unique attributes	21	21	27	3
Required elements	1	1	1	12
Required attributes	8	8	5	3
Elements at position 0	0	0	0	0
Sum	45	45	57	42

Table 1. Structured Document Complexity Metrics of representation schemes

The results obtained for VRML and X3D are equal, because both standards use schemes with equivalent basic elements and attributes. While in the VRML and X3D schemes, different hierarchical document elements have been classified as unique elements, in the ActionScript scheme, different classes and data types (potentially corresponding to different elements in VRML/X3D) have been classified as unique elements. In the ML-SCM scheme, unique elements cover different RDF, RDFS and OWL elements as well as semantic properties of 3D content, which are encoded by document elements (according to the RDF syntax). Unique attributes are different properties occurring in hierarchical VRML/X3D elements or properties of objects in the ActionScript scheme. Since, in the ML-SCM scheme the content properties are encoded using document elements, only a few attributes, which are primary RDF, RDFS and OWL attributes, may be classified as unique attributes in the ML-SCM scheme. In general, there have been no elements classified as required for content representations except for the scene and view, which are roots of the created scenes (representations). The calculated values of the Structured Document Complexity Metric show that the complexity of the ML-SCM scheme is a little lower than the complexity of the VRML/X3D schemes and it is much lower than the complexity of the ActionScript scheme. Unlike in the ML-SCM, VRML and X3D schemes, in the ActionScript scheme, several aspects need to be implemented with multiple instructions (e.g., texturing and navigation), which requires the use of additional unique elements (classes and data types) as well as attributes (object properties) in the ActionScript scheme.

Size Metrics. The number of bytes (Fig. 8) and the number of logical lines of code—LLOC (Fig. 9)—without comments—have been used to measure the size of representations (scenes). The graphs present the metrics in relation to the number of components included in the representations.

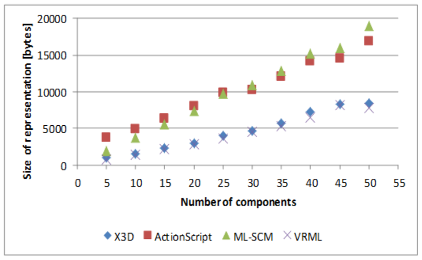


Fig. 8. The size of representation (bytes) in relation to the number of its components

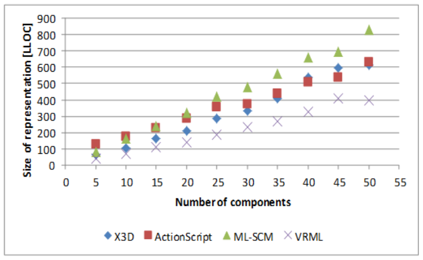


Fig. 9. The size of representation (LLOC) in relation to the number of its components

The differences between the languages are relatively high in terms of both the number of bytes and the number of LLOC. In both comparisons, VRML is the most concise language, while ML-SCM/RDF-Turtle is the most verbose language—even a little more than ActionScript.

Halstead Metrics. Halstead metrics have been used to measure the complexity of representations (scenes). The calculated Halstead metrics cover: the vocabulary and the length of content representations, the volume corresponding to the size of the representations, the difficulty corresponding to error proneness of the representations, the effort in implementation and analysis of the representations as well as the estimated time required for the development of the representations. The particular Halstead metrics in relation to the number of components included in a representation are presented in the graphs in Fig. 10-15. VRML and X3D representations have been presented together, because both standards have the same values of the metrics, since they use schemes with equivalent basic elements and attributes.

Vocabulary (Fig. 10), which is the sum of unique operators (n1) and unique operands (n2):

`Voc = n1 + n2,`

is highest for VRML/X3D, because of a high number of unique operands, which are individual scene graph nodes. In contrast to the other languages, in VRML/X3D, a relationship between two components in a generated representation is reflected by nesting one component in another component with specifying all intermediate nodes, which are also classified as unique operands, e.g., applying a `Material` to a `Shape` requires an intermediate `Appearance` node to be nested in the `Shape` node. In the other languages, such associations are more frequently described directly—without using intermediate nodes.

Length (Fig. 11), which is the sum of the total number of operators (N1) and the total number of operands (N2) of a representation:

`Len = N1 + N2,`

is lowest for VRML/X3D. In the VRML/X3D representations, as opposed to the other representations, operands typically occur once and all references to them are specified by the nesting of attributes and other nodes. Therefore, the operands do not require to be additionally explicitly indicated (e.g., by proper instructions or statements), and the length of VRML/X3D representations is lower than the length of the other representations, in which all references to operands must be explicitly declared by referring to their identifiers. ML-SCM representations are shorter than ActionScript representations because of the possibility to syntactically gather similar statements that share a common subject, e.g., `[shape rdf:type scm:AppearanceComponent , scm:Box].`

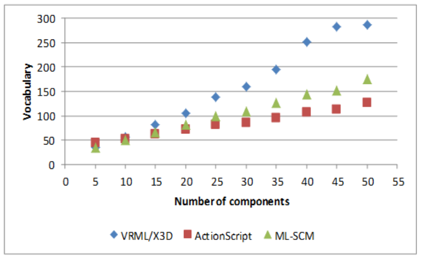


Fig. 10. The vocabulary of representation in relation to the number of its components

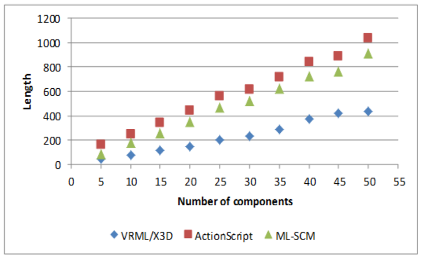


Fig. 11. The length of representation in relation to the number of its components

The graph of volume (Fig. 12), which depends on the length and the vocabulary:

```
Vol = Len * log2(Voc),
```

is similar to the graph of length.

In contrast to the other Halstead metrics discussed, difficulty (Fig. 13), which is given by the formula:

```
Diff = n1 / 2 * N2 / n2,
```

has similar values independently of the number of components in the scene. It is lowest for VRML/X3D representations (low error proneness) because of the relatively low values of the number of distinct operators and the total number of operands and a relatively high value of the number of distinct operands. A relatively high difficulty of ActionScript representations (high error proneness) is caused by relatively high values of the first two factors and a relatively low value of the third factor.

The effort (Fig. 14) and the time (Fig. 15) required for the implementation or analysis of representations, which are the products of the difficulty and the volume:

```
Eff = Diff * Vol,
Time [h] = Eff / (18 * 3600),
```

are lowest for VRML/X3D representations because of the relatively low values of their difficulties and volumes. The highest values of effort and time occur for ActionScript representations.

6.2. Multi-platform Representation

The primary goal of the proposed method is to improve the development of 3D content presentations for multiple platforms. The proposed method has been evaluated in terms of the following four metrics.

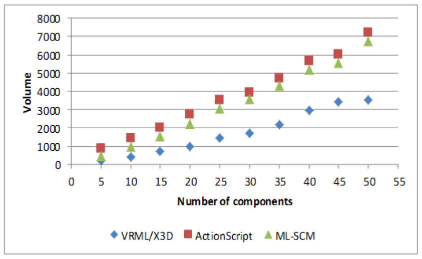


Fig. 12. The volume of representation in relation to the number of its components

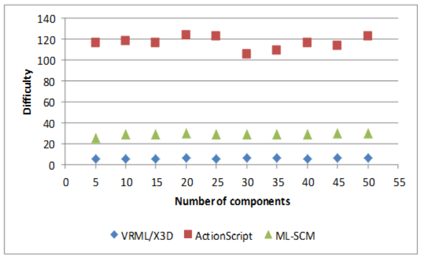


Fig. 13. The difficulty of representation in relation to the number of its components

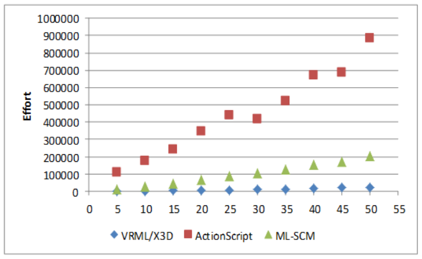


Fig. 14. The effort in the implementation of representation in relation to the number of its components

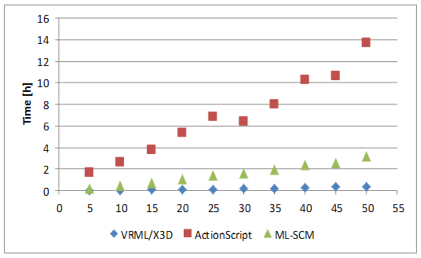


Fig. 15. The time required for the implementation of representation in relation to the number of its components

The profit from automatic generation of PSRs. The profit from automatic generation of PSRs for different platforms in relation to the number of components of the primary representation (a PIR) has been calculated as the ratio of the overall size of the PSRs, which have been generated on the basis of a primary PIR (implemented with ML-SCM) and encoded with the selected content representation languages, to the size of the PIR:

```
    Profit = Size(PSRs) / Size(PIR).
```

The metric shows, how much work on implementation can be saved when implementing a PIR and using the proposed method to automatically generate the corresponding PSRs, rather than implementing the desirable PSRs from scratch using the particular languages, independently one from another. The profit is presented in Fig. 16.

The values of the profit vary predominantly in the range 1.7 to 2.1 for the size expressed in the number of bytes and from 1.9 to 2.3—for the size expressed in the number of LLOC. Although, the profit values are decreased by the relatively high size of PIRs in comparison to the corresponding PSRs, about 50% of work may be saved when using the proposed solution with the three presentation platforms. The higher number of presentation platforms would result in the higher values of profit.

The cost of elementary changes in content representations. The cost of elementary changes in content representations is the average size of code that is required to be added, deleted or modified, to add, delete or modify a component or a property of a 3D content representation. It is assumed that a change in a representation is related to at least one of

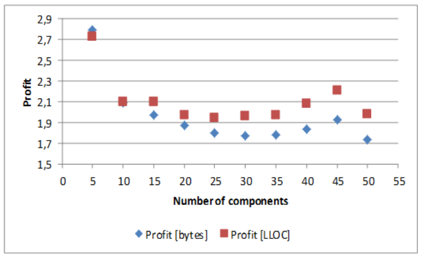


Fig. 16. The profit from automatic generation of PSRs

its basic elements. A basic element of a PIR is an S, while a basic element of a PSR that corresponds to the PIR is a TS associated with an S from the PIR. Hence, the cost of an elementary change in a content representation is the average size of an S (for PIRs) or the average size of a TS (for PSRs):

```
Cost = AvgSize(S|TS).
```

The values of the metric have been calculated for the particular content representation languages and they are presented in Table 2 (in bytes and LLOC).

In the proposed method, Ss of ML-SCM typically gather multiple instructions (multiple LLOC) of the other content representation languages. ActionScript is the most verbose of the selected languages, as to represent some aspects of 3D content on the Flash platform (e.g., navigation), a number of LLOC need to be implemented to create an appropriate T. The most concise language is VRML, which is equivalent to X3D in terms of the provided level of abstraction. A slight difference between VRML and X3D in the number of bytes is caused by the difference in their syntaxes—the syntax of VRML is more concise than the chosen verbose XML-based syntax of X3D.

The cost of the introduction of a content presentation platform. The cost of the introduction of a content presentation platform is the overall size of code that must be implemented to introduce a 3D content presentation platform into the system. Hence, the cost incorporates the size of the TKB, which describes the transformation of PIRs to PSRs, and a TB, which corresponds to the language of the new platform introduced:

```
Cost = Size(TKB) + Size(TB).
```

The metric has been calculated and expressed in the number of bytes and the number of LLOC for the selected content representation languages that determine the content presentation platforms to be used (Table 3).

Size	**VRML**	**X3D**	**ActionScript**	**ML-SCM**
NoB	35	39	130	39
LLOC	2	2	4	1

Table 2. The cost of elementary changes in content representations

	VRML		**X3D**		**ActionScript**	
	NoB	LLOC	NoB	LLOC	NoB	LLOC
TKB	22291	789	22291	789	25290	943
TB	1193	71	1312	71	4957	152
Sum	23484	860	23603	860	30247	1095

Table 3. The cost of the introduction of a content presentation platform

On the basis of the calculated values of the metric, an estimated cost of the introduction of a new platform into the system may be determined. A new platform whose functionality (a set of content components and properties) needs to be equivalent to the already implemented platforms, is anticipated to require about 860-1095 LLOC. The lower bound is more probable for hierarchical declarative languages (e.g., XML3D), whereas the upper bound is more probable for structural and object-oriented imperative languages (e.g., Java3D). However, the exact value depends on the capabilities a particular target language (and programming libraries) used. The more advanced are the capabilities of the language and the libraries provided, the lower should be the cost. For example, a single instruction may enable one of several available navigation modes.

The profit from code generation for a new platform. The profit from code generation for a new platform is directly proportional to the number of PIRs that are available in the system and that are to be transformed to PSRs compliant with the new platform, and the size of the generated PSRs. The metric is given by the formula:

`Profit = N * Size(PSR) / Cost(TKB + TB),`

where: `N` – the number of PIRs that are available in the system and that are to be transformed; `Size(PSR)` – the size of the PSR to be generated, estimated on the basis of the generated PSRs; `Cost(TKB + TB) = Size(TKB) + Size(TB)` – the overall size of the TB and the TKB that enable the transformation (calculated in the previous subsection).

The metric has been calculated for the PIR size equal to 50 components. This value has been determined on the basis of typical virtual museum scenes available in the ARCO virtual museum system [37, 38]. However, there are several more complex scenes available in ARCO, for which the profit from transformation can be higher. The values of the profit in relation to the number of PIRs are presented in the graphs in Fig. 17-18.

The profit increases linearly with the increase in the number of PIRs that are available in the system and need to be transformed. The attainable profit is higher for the X3D and ActionScript languages, as the size of X3D and ActionScript representations is typically larger than the size of VRML representations.

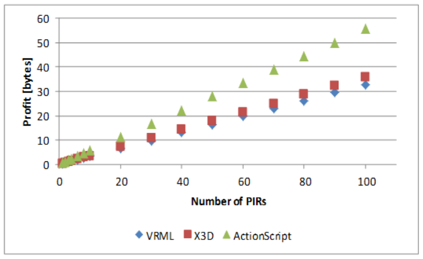


Fig. 17. The profit from code generation for a new platform (for PSR size in bytes)

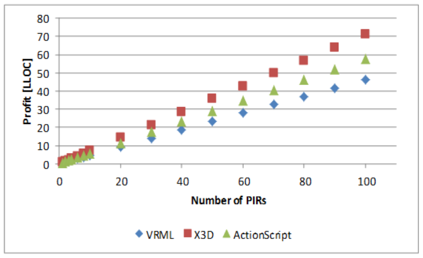


Fig. 18. The profit from code generation for a new platform (for PSR size in LLOC)

6.3. Transformation Algorithm

The efficiency of the proposed TA (which has been described in Section 3.2) has been evaluated. The time of transformation of PIRs to PSRs has been measured for different sizes of PIRs. The size of PIRs is specified in the number of components contained in the PIR and it changes over a range (Section 6). The results are presented in Fig. 19.

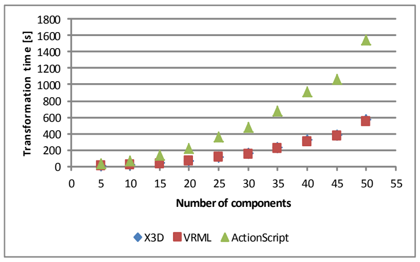


Fig. 19. Time of PIR to PSR transformation

The graph presents polynomial time required for the transformation of PIRs, which are encoded using ML-SCM/RDF-Turtle, to PSRs, which are encoded using the selected languages. Transformation times for VRML and X3D are similar, as the languages are supported by a common TKB and structurally equivalent Ts. Moreover, transformation time for VRML and X3D is more than twice lower than the transformation time for ActionScript. The difference is caused by the remarkably different structures of VRML/X3D and ActionScript representations. While transformation for VRML/X3D is based mainly on nesting Ts, which is relatively low time-consuming, transformation for ActionScript is based mainly on setting TPs, which is relatively high time-consuming, because the number of TPs is higher than the number of Ts—on average, more than two TPs are included in a T.

6.4. Discussion

The results obtained show high profit from the implementation and the use of a new transformation when adding a new platform in applications with a high number of scenes in contrast to the implementation of the counterparts of the scenes with a new content representation language supported by the new platform. The profit is also high when PSRs are encoded using programming libraries that do not provide some required components and properties, which need to be implemented from scratch. In such cases, the high profit is the result of the high re-usability of such components and properties, which, once implemented in Ts may be reused multiple times.

In the majority of tests related to the complexity of content representations, the results obtained for ML-SCM are better than the results obtained for the ActionScript language in terms of code analysis—it has a lower complexity of the scheme, vocabulary, length, volume and difficulty of representations and its use requires less effort and time for understanding and implementing content representations. However, the size (in bytes and LLOC) of representations encoded in ML-SCM/RDF-Turtle is higher than the size of representations encoded with the other languages, because of the verbose syntax of the RDF format, which is used in the ML-SCM representations. However, the cost of the PIRs is compensated by the high profit from the automatic generation of multiple PSRs.

The transformation algorithm uses the two implemented TKBs (for VRML/X3D and for ActionScript) and the three implemented TBs (an individual for each language) to transform PIRs to PSRs. The algorithm transforms PIRs in polynomial time. The test, carried out using a moderately powerful computer, indicates relatively long time required for transformation. However, the algorithm could be optimized, e.g., by using multi-threading or by the employment of a more efficient SPARQL query engine.

7. Conclusions and Future Works

In this paper, a new method of building multi-platform 3D presentations has been presented. The method leverages the semantic web standards to provide a means of platform-independent representation of 3D content and efficient generic transformation between different 3D content representations, which goes beyond the current state of the art in the field of multi-platform content presentation.

The presented solution has several important advantages in comparison to the available approaches to 3D content presentation. First, the use of the semantic web techniques enables modeling complex dependencies and relations between content components as well as rules of combining different components into composite objects and scenes. Moreover, it permits declarative conceptual content creation taking into account hidden knowledge, which may be inferred and used in the modeling process. Second, the possible use of well-established 3D content presentation tools, programming languages and libraries liberates users from the installation of additional software, which can improve the dissemination of 3D content. The method is convenient for environments, which cover various hardware and software systems (such as the web), as the development of TKBs and TBs requires less effort in implementation than the development of individual content presentations for different presentation platforms or a specific presentation platform for different systems. Third, in comparison to the available approaches, the proposed method can be used to produce content that may be managed (indexed, searched and analyzed) in a simpler way—due to the conformance to the semantic web standards.

Possible directions of future research incorporate several facets. First, the proposed method can be extended with semantic transformation of declarative rule-based descriptions of content behavior. Such transformation should be performed in a different manner than the transformation of Ss, which are logical facts, as logical rules cannot be used for inference before being transformed and executed. Second, the proposed method can be combined with the approach to transformation of 3D content formats [12]. Next, semantic queries to logical parts of the content can be used for 3D content creation. Furthermore, a visual modeling tool supporting semantic 3D content creation can be developed. In comparison to typical semantic editors, such tool could significantly facilitate design and enable efficient editing of semantic 3D scenes at different levels of abstraction, which are determined by the ontologies used. Finally, the method currently permits only uni-directional transformation of PIRs to PSRs. To synchronize PIRs with their final equivalents (PSRs) and enable decompilation of PSRs to their semantic prototypes (PIRs), persistent link between the semantic objects of a PIR and the components of the generated PSR should be maintained. Such link could also permit semantic management and exploration of 3D content in real-time.

Acknowledgments. This research work has been supported by the Polish National Science Centre Grant No. DEC-2012/07/B/ST6/01523.

References

1. Aivosto: Lines of code metrics (LOC), http://www.aivosto.com/project/help/pm-loc.html
2. Almer, A., Schnabel, T., Stelzl, H., Stieg, J., Luley, P.: A tourism information system for rural areas based on a multi platform concept. In: Proc. of the 6th int. conf. on Web and Wireless Geographical Information Systems. pp. 31–41. Springer-Verlag Berlin Heidelberg (2006)
3. Apache: Apache Jena, http://jena.apache.org/
4. Aylett, R., Luck, M.: Applying artificial intelligence to virtual reality: Intelligent virtual environments. Applied Artificial Intelligence 14, 3–32 (2000)
5. Bilasco, I.M., Gensel, J., Villanova-Oliver, M., Martin, H.: 3dseam: a model for annotating 3d scenes using mpeg-7. In: Proc. of the 11th int. conference on 3D web technology. pp. 65–74. Columbia, MD, USA (April 18-21, 2006)
6. Bilasco, I.M., Villanova-Oliver, M., Gensel, J., Martin, H.: Semantic-based rules for 3d scene adaptation. In: Proc. of the 12th int. conference on 3D Web technology. pp. 97–100. Umbria, Italy (April 15-18, 2007)
7. Bille, W., Pellens, B., Kleinermann, F., Troyer, O.D.: Intelligent modelling of virtual worlds using domain ontologies. In: Proc. of the Workshop of Intelligent Computing (WIC), held in conjunction with the MICAI 2004 conference. pp. 272–279. Mexico City, Mexico (2004)
8. Cavazza, M., Palmer, I.: High-level interpretation in virtual environments. Applied AI 14, 125–144 (2000)
9. Celakovski, S., Davcev, D.: Multiplatform real-time rendering of mpeg-4 3d scenes with microsoft xna. In: ICT Innovations 2009. pp. 337–344. Springer-Verlag Berlin Heidelberg (2010)
10. Flotyński, J., Walczak, K.: Semantic multi-layered design of interactive 3d presentations. In: Proc. of the Federated Conf. on Computer Science and Information Systems. pp. 541–548. IEEE, Kraków, Poland (September 8-11, 2013)
11. Flotyński, J.: Semantic modelling of interactive 3d content with domain-specific ontologies. Procedia Computer Science 35, 531–540 (2014)
12. Flotyński, J., Dalkowski, J., Walczak, K.: Building multi-platform 3d virtual museum exhibitions with flex-vr. In: The 18th International Conference on Virtual Systems and Multimedia. pp. 391–398. Milan, Italy (September 2-5, 2012)
13. Flotyński, J., Walczak, K.: Conceptual semantic representation of 3d content. Lecture Notes in Business Information Processing: 16th Int. Conf. on Business Information Systems, Poznań, Poland, 19 - 20 June, 2013 160, 244–257 (2013)
14. Flotyński, J., Walczak, K.: Conceptual knowledge-based modeling of interactive 3d content. The Visual Computer pp. 1–20 (2014), http://dx.doi.org/10.1007/s00371-014-1011-9
15. Flotyński, J., Walczak, K.: Multi-platform semantic representation of interactive 3d content. Technological Innovation for Collective Awareness Systems, IFIP Advances in Information and Communication Technology 432, 63–72 (2014)
16. Han, J., Kang, I., Hyun, C., Woo, J.S., Eom, Y.I.: Multi-platform online game design and architecture. In: Proceedings of the 2005 IFIP TC13 international conference on Human-Computer Interaction. pp. 1116–1119. Springer Berlin Heidelberg (2005)
17. IBM: Halstead Metrics, http://pic.dhe.ibm.com/infocenter/rtrthelp/v8r0m0
18. Jankowski, J., Decker, S.: A dual-mode user interface for accessing 3d content on the world wide web. In: Proceedings of the 21st International World Wide Web Conference (WWW'12). pp. 1047–1056. ACM (April 16-20 2012)
19. Kalogerakis, E., Christodoulakis, S., Moumoutzis, N.: Coupling ontologies with graphics content for knowledge driven visualization. In: VR '06 Proceedings of the IEEE conference on Virtual Reality. pp. 43–50. Alexandria, Virginia, USA (March 25-29, 2006)

20. Latoschik, M.E., Biermann, P., Wachsmuth, I.: Knowledge in the loop: Semantics representation for multimodal simulative environments. In: Lecture Notes in Computer Science. pp. 25–39. Springer Berlin Heidelberg (2005)
21. Latoschik, M.E., Blach, R.: Semantic modelling for virtual worlds - a novel paradigm for realtime interactive systems? In: Proc. of the 2008 ACM symp. on VR software and technology. pp. 17–20. Bordeaux, France (October 27-29, 2008)
22. Latoschik, M.E., Frohlich, C.: Semantic reflection for intelligent virtual environments. In: IEEE Virtual Reality Conference 2007. pp. 305–306. Charlotte, USA (March 10-14, 2007)
23. Lugrin, J., Cavazza, M.: Making sense of virtual environments: action representation, grounding and common sense. In: Proc. of the 12th int. conference on Intelligent user interfaces. pp. 225–234. Honolulu, HI, USA (January 28-31, 2007)
24. Mendes, C.M., Drees, D.R., Silva, L., Bellon, O.R.: Interactive 3d visualization of natural and cultural assets. In: Proceedings of the second workshop on eHeritage and digital art preservation. pp. 49–54. Firenze, Italy (October 25-29, 2010)
25. O'Reilly: Metrics for XML Projects, http://www.oreillynet.com/xml/blog/2006/05/
26. Otto, K.A.: Semantic virtual environments. In: Special interest tracks and posters of the 14th international conference on World Wide Web. pp. 1036–1037. Japan (May 10-14, 2005)
27. Parsia, C..: Pellet: OWL 2 Reasoner for Java, http://clarkparsia.com/pellet/
28. Pellens, B., Kleinermann, F., Troyer, O.D.: A development environment using behavior patterns to facilitate building 3d/vr applications. In: Proc. of the 6th Australasian Conf. on Interactive Entertainment. ACM, Sydney, Australia (2009)
29. Pellens, B., Troyer, O.D., Bille, W., Kleinermann, F., Romero, R.: An ontology-driven approach for modeling behavior in virtual environments. In: Proceedings of Ontology Mining and Engineering and its Use for Virtual Reality (WOMEUVR 2005). pp. 1215–1224. Springer-Verlag, Agia Napa, Cyprus (2005)
30. Pellens, B., Troyer, O.D., Kleinermann, F.: Codepa: a conceptual design pattern approach to model behavior for x3d worlds. In: Proc. of the 13th int. symp. on 3D web technology. pp. 91–99. Los Angeles (August 09-10, 2008)
31. Pittarello, F., Faveri, A.: Semantic description of 3d environments: a proposal based on web standards. In: Proc. of the 11th Int. Conf. on 3D web Techn. pp. 85–95. Columbia, MD, USA (April 18-21, 2006)
32. Tack, K., Lafruit, G., Catthoor, F., Lauwereins, R.: Platform independent optimisation of multiresolution 3d content to enable universal media access. The Visual Computer 22, Issue 8, 577–590 (August 2006)
33. Troyer, O.D., Bille, W., Romero, R., Stuer, P.: On generating virtual worlds from domain ontologies. In: Proceedings of the 9th International Conference on Multi-Media Modeling. pp. 279–294. Taipei, Taiwan (2003)
34. Troyer, O.D., Kleinermann, F., Pellens, B., Bille, W.: Conceptual modeling for virtual reality. In: Tutorials, posters, panels and industrial contributions at the 26th int. conf. on Conceptual modeling. pp. 3–18. Darlinghurst (2007)
35. Walczak, K.: Beh-vr: Modeling behavior of dynamic virtual reality contents. In: Proc. of the 12th Int. Conf. on Virtual Systems and Multimedia VSMM 2006. pp. 40–51. Lecture Notes in Computer Sciences, Springer Verlag Heidelberg (2006)
36. Walczak, K.: Flex-vr: Configurable 3d web applications. In: Proc. of the Int. Conf. on Human System Interaction HSI 2008. pp. 135–140. Kraków, Poland (May 25-27, 2008)
37. Walczak, K., Cellary, W., White, M.: Virtual museum exhibitions. Computer 39(3), 93–95 (Mar 2006)
38. White, M., Mourkoussis, N., Darcy, J., Petridis, P., Liarokapis, F., Lister, P., Walczak, K., Wojciechowski, R., Cellary, W., Chmielewski, J., Stawniak, M., Wiza, W., Patel, M., Stevenson, J., Manley, J., Giorgini, F., Sayd, P., Gaspard, F.: ARCO - An architecture for digitization, management and presentation of virtual exhibitions. pp. 622–625. IEEE Computer Soc., Computer Graphics International Conference (CGI 2004), Crete, Greece, Jun 16-19, 2004

39. Wiebusch, D., Latoschik, M.E.: Enhanced decoupling of components in intelligent realtime interactive systems using ontologies. In: 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS). pp. 43–51. Orange County, CA, USA (2012)
40. Zaid, L.A., Kleinermann, F., Troyer, O.D.: Applying semantic web technology to feature modeling. In: Proceedings of the 2009 ACM symposium on Applied Computing. pp. 1252–1256. Honolulu, Hawaii, USA (2009)

Jakub Flotyński received the M.Sc. degree in Electronics and Telecommunications (2011) and in Computer Science (2010) at the Poznan University of Technology in Poland. His research interests include multimedia systems, virtual reality and semantic web. He has worked at the Department of Information Technology at the Poznan University of Economics (Poland) since 2011.

Krzysztof Walczak holds the Ph.D. Hab. degree in Computer Science (Multimedia Systems, Gdansk 2010) and is an Associate Professor in the Department of Information Technology at the Poznan University of Economics in Poland. His current research interests include virtual reality and augmented reality applications, multimedia systems, semantic web and distance learning.

Received: December 18, 2013; Accepted: August 28, 2014.