UNIVERSITY OF NOVI SAD
FACULTY OF TECHNICAL SCIENCES

Amel Abdyssalam Alhaag

Model-Driven Software Architecture for the Management of Educational Resources Metadata

Ph.D. DISSERTATION

June, 2018.

Dedication

For my father Abdyssalam Alhaag, to my mother Mahjoba Atea and for all those who encouraged me to fly toward my dream.

Apstrakt

Tema ove disertacije je upravljanje obrazovnim resursima. Preciznije, istraživanje je fokusirano na pronalaženje resursa za šta je potrebno omogućiti njihovo skladištenje tako da mogu biti identifikovani i isporučeni u skladu sa zahtevima nastavne instrukcije. Pronalaženje obrazovnih resursa može biti unapređeno uvođenjem dodatnih informacija kao što su metapodaci.

Disertacija se bavi upravljanjem ovim metapodacima putem specijalizovanih softverskih aplikacija. Cilj istraživanja je bio da se omogući dinamičko prilagođavanje skupa metapodataka kojim se opisuju obrazovni resursi u nekom digitalnom repozitorijumu. Konkretno, sledeće teme su pokrivene ovim istraživanjem:

- Opis semantike obrazovnih resursa korišćenjem metapodataka. Ovo se odnosi na metapodatke koji su specifični za određeni domen, kao i na one koji su domenski neutralni
- Softverske aplikacije za upravljanje metapodacima obrazovnih resursa
- Dinamičko prilagođavanje skupova metapodataka
- Programsko generisanje modela metapodataka zasnovano na modelom vođenom pristupu

Bez obzira na njihovu upotrebu, metapodaci mogu biti podeljeni u dve generalne kategorije. Prva kategorija se odnosi na metapodatke koji opisuju one karakteristike obrazovnog resursa koje nisu striktno povezane sa oblašću na koju se obrazovni resurs odnosi. Ovakve metapodatke nazivamo domenski-nezavisni metapodaci. Ovi metapodaci su generalni i mogu biti korišćeni u različitim obrazovnim resursima bez obzira na njihovu oblast. Primeri takvih metapodataka su format dokumenta, autor, jezik itd. Ovakvi metapodaci mogu biti opisani različitim formalnim modelima među kojima su trenutno najpoznatiji IEEE LOM i Dublin Core. U drugu kategoriju spadaju metapodaci koji koriste informacije specifične za određenu oblast. Na primer, ako je obrazovni resurs iz oblasti računarstva, metapodaci mogu biti vezani za programersku tehnologiju koju resurs objašnjava. U mnogim oblastima su razvijene taksonomije koje dodatno klasifikuju obrazovne resurse u toj oblasti. Kao primere takvih taksonomija pomenimo ACM Computing Classification

System iz oblasti računarstva i Mathematics Subject Classification iz oblasti matematike.

Istraživanje prikazano u ovoj disertaciji se bavi dinamičkim proširenjem skupa metapodataka u softverskoj aplikaciji za upravljanje obrazovnim resursima. Pri tome, koristi se pristup vođen modelom za automatsko generisanje softverske aplikacije koja ima podršku za korisnički definisane skupove metapodataka. Pristup koristi izvorne domenske modele kao osnovu za generisanje ciljnih modela. Izvorni model opisuje strukturu i ponašanje sistema na različitim nivoima apstrakcije. U ovakvom pristupu, proces razvoja softverske aplikacije počinje kreiranjem izvornog modela. Izvorni model se smatra platformski-nezavisnim modelom jer je fokusiran na reprezentaciju domenskog znanja bez bavljenja detaljima implementacije. U temi kojom se ova disertacija bavi, izvorni model je osnova za programsko generisanje konačne softverske aplikacije. Ova aplikacija predstavlja ciljni model dobijen transformacijom izvornog modela putem skupa transformacionih pravila. Za razliku od izvornog modela, ciljni model je platformski-specifičan i sadrži izvorni kod konačnog softverskog proizvoda. Ovakav, modelom-vođen pristup, obezbeđuje da se pri razvoju inicijalno fokusira na domensko znanje, umesto na algoritme i programerske detalje. Obzirom da se ciljni model programski generiše, povećava se produktivnost, kao i prenosivost sistema, obzirom da isti domenski model može biti korišćen za generisanje različitih ciljnih modela. Takođe, odvajanje reprezentacija domenskog znanja od detalja implementacije olakšava uključivanje domenskih eksperata u fazu razvoja.

Generalno, softverski sistemi za upravljanje obrazovnim resursima se suočavaju sa dva izazova. Pre svega, oni treba da podrže neki generalni skup metapodataka kako bi omogućili upravljanje obrazovnim resursima koji pripadaju različitim domenima. Sa druge strane, neophodno je opisati i delove značenja resursa koji su specifični za određeni domen. Vrlo je teško implementirati softversku aplikaciju koja sadrži predefinisane skupove metapodataka za najrazličitije oblasti. Čak i kada bi postojala aplikacija koja inicijalno podržava vrlo raznolike skupove metapodataka, ostaje problem kasnijeg uvođenja novih skupova metapodataka. Ako su skupovi metapodataka statički predefinisani, aplikacija ne može da omogući izmenu postojećih skupova metapodataka niti uvođenje metapodataka iz novog domena.

Zato je cilj istraživanja predstavljenog u ovoj disertaciji da se omogući dinamičko prilagođavanje skupova metapodataka u softverskoj aplikaciji za upravljanje obrazovnim resursima. Osnovna ideja je da se omogući

korisnicima da samostalno definišu skupove metapodataka. Na ovaj način, korisnik može da prilagodi aplikaciju da koristi metapodatke iz domena kojim se on bavi. Obzirom da korisnici u pravilu nemaju veštine potrebne za razvoj softverske aplikacije koja bi bila prilagođena njegovim metapodacima, ova disertacije predstavlja izvršivu platformu koja programski generiše konačnu softversku aplikaciju za upravljanje obrazovnim resursima. Ovakvo rešenje obezbeđuje da će korisnici moći da rade sa metapodacima iz svog domena bez potrebe da razvijaju ili naručuju novu softversku aplikaciju. Predložena izvršiva platforma je ta koja pruža dinamičko prilagođavanje skupa metapodataka željenom domenu.

U skladu sa navedenim, definisana je hipoteza istraživanja: Da bi se omogućilo upravljanje obrazovnim resursima čije je značenje opisano nepredefinisanim domenski-specifičnim skupom metapodataka, potrebno je kreirati sistem koji može jednostavno biti prilagođen upravljanju obrazovnim resursima u određenom domenu. Moguće je ispuniti ovaj zahtev kroz implementaciju podrške za dodavanje različitih domenski-specifičnih metapodataka dinamički.

Za realizaciju je korišćen goreopisani pristup vođen modelom. Kao što je objašnjeno, ovaj pristup omogućuje generisanje ciljnog modela na osnovu formalno definisanog izvornog modela. Kada je reč o korišćenju ovog pristupa za upravljanje metapodacima obrazovnih resursa, izvorni model je domenski model određenog skupa metapodataka, a ciljni model je programski generisana softverska aplikacija za upravljanje metapodacima obrazovnih resursa u tom domenu. Rezultat ove disertacije je izvršiva platforma za generisanje softverske aplikacije koja upravlja obrazovnim resursima koji su opisani izmenjivim skupovima metapodataka. Platforma je proširenje Kroki alata, koji omogućuje kreiranje softverskih prototipova kroz modelom vođen pristup.

Platforma je verifikovana putem eksperimenta u kojem je 16 studenata softverskog inženjerstva evaluiralo karakteristike platforme. Studenti su imali zadatak da koristeći ovde predloženu platformu kreiraju novi model metapodataka, generišu softversku aplikaciju na bazi ovog modela i opišu obrazovne resurse koristeći metapodatke sadržane u kreiranom modelu. Eksperiment je verifikovao da platforma zadovoljava postavljene zahteve.

List of figures

List of code listings

List of tables

Acknowledgement

I would like to thank many people, especially the enthusiastic supervisor Prof. Goran Savić. My PhD was a great experience and I thank my supervisor wholeheartedly, not only for his tremendous academic support, but also for giving me a lot of wonderful opportunities.

Special thanks to the supervisor, Prof. Gordana Milosavljević, for her very useful views, comments and suggestions. You have been very supportive and always there when I have a question or concern.

I also express my sincere thanks to Prof. Zora Konjović who gave me the opportunity to join their team as an intern. Without valuable support, this research will not be possible.

Then, I would like to express my sincere gratitude to Prof. Milan Segedinac for the continued support of related research, for his patience, motivation and knowledge.

A great Thanks to my sister Iman Alhaag for her guidance helped her all the time in researching and writing this thesis. I did not imagine a better consultant and consultant to study for a doctorate.

In addition to my advisers, I would like to thank the rest of the theses committee for their encouraging comments and encouragement, but also for the difficult question that led me to expand my research from different points of view.

In addition, I am grateful to Dr Siham Sassi Abdul Rahman for helping me check this PhD dissertation.

I would like to thank all those who responded to the questionnaire and the participants who were interviewed; without their support this work would not have been completed.

My husband, Abdullah Alkash, who was a supporter throughout my education. He has enabled me to love and encourage him throughout our marriage to complete my dream.

Abbreviation

AP	Application Profile
ACM	Association For Computer Machinery
CCS	Computing Classification System
DSL	Domain Specific Language
DC	Dublin Core Standard
ER	Educational Resources
EUIS	Enterprise User Interface Specification
GUI	Graphic User Interface
LO	Learning Object
LOM	Learning Object Metadata
LTSC	Learning Technology Standards Committee
MSC	Mathematics Subject Classification
MDA	Model-Driven Approach
PIM	Platform Independent Model
PSM	Platform Specific Model
MRDB	Mathematical Reviewing Databases

Table of contents

1. Introduction

Contemporary education heavily relies upon educational resources that are distributed in digital form. A high-quality learning process demands for easily discoverable digital learning resources. Such resources are mostly stored in digital educational repositories, which provide their storage and retrieval. One of the main factors that determine the availability of educational resources is the expressiveness of metadata used for describing them.

The topic of this thesis is the management of educational resources. More precisely, the research is focused on the discovery of such resources, which relates to storing resources so that they can be identified and delivered in accordance with the specific instructional demands. The discovery of educational resources can be improved by introducing additional information through external components, such as metadata.

The research deals with managing metadata of educational resources using a software application. The purpose of the research is to enable dynamic customization of metadata that describes educational resources in digital repositories.

The research is focused on the following issues:

- Describing the semantics of educational resources using metadata. This covers domain-neutral, as well as domain-specific metadata
- Software applications for the management of educational resources metadata
- Dynamic customization of metadata sets
- Model-driven approach for programmatic generation of a software platform for managing metadata of educational resources

1.1. Educational resources

A learning environment relies on documenting learning material and other content that is used in learning process. Although the documents can vary based on the content type, format or purpose, we use the term “educational

resources" as an umbrella term that includes all kinds of such documents. Recently, the educational resources are commonly represented in digital formats to be used in software learning environments. Different formats are present, such as PDF and Office documents, images, videos, etc. Browser-readable formats are preferred since the most learning platforms are used online using an internet browser. Additionally, such format is suggested by the popular e-learning specifications such as IMS Content Packaging, SCORM and IMS Learning Design.

Traditionally, monolithic educational resources are used, which means that learning content is grouped within a single document, such as textbook. Using such resources can be inadequate in digital learning environments, since that they make difficult reuse of learning content which is one of the main demands set upon modern educational resources. To promote reusability of educational content, as well as personalization and individualization, the learning process should rely on small, durable and reusable educational resources (Littlejohn, 2003). Such resources can facilitate the creation of new courses (Cohen and Nycz, 2006) and the same learning objects can be used across different courses (Savic, Segedinac and Konjovic, 2012). This type of educational resources is commonly named learning objects. Downes (2004) explains that learning objects should be sharable, digital, modular, interoperable and discoverable.

Polsani (2005) explains that the term "learning objects" is introduced by Wayne Hodgins in 1994 when he names his working group "Learning Architectures, APIs and Learning Objects". Since then, many definitions have been created to describe this kind of educational resources.

Wiley (2002) provides definition of the learning object by adopting the term "object" from Dahl and Nygaard (1966). He defined learning object as *"elements of a new type of computer-based instruction grounded in the object-oriented paradigm of computer science*". The fact that the term originated from the terminology that was already established within computer science, influenced further description techniques of learning objects. The techniques are based on object-oriented modelling and principles such as "abstraction, concurrency, encapsulation, hierarchy, persistence, polymorphism and typing" (Friesen, 2003).

A study conducted by Young and Morrison (2002) provides a simple definition for learning object which is "*a computer mediated or delivered module or unit, that stands by itself, that provides a meaningful learning experience in a planned learning context*".

Within the IEEE standards body, there is a subgroup that focuses specifically on learning technology standards, such as learning objects and their metadata. This group is known as the IEEE P1484.12 Learning Object Metadata Working Group. The group aims to develop standards, recommended practices, and guidelines for learning technology and Learning Technology Standards Committee (LTSC).

According to this group, a learning object has been defined as "*any entity, digital or non-digital, which can be used, reused or referenced during technology supported learning. Examples of technology-supported learning include computer-based training systems, interactive learning environments, intelligent computer-aided instruction systems, distance learning systems, and collaborative learning environments. Examples of Learning Objects include multimedia content, instructional content, learning objectives, instructional software and software tools, and persons, organizations, or events referenced during technology supported learning.*" (IEEE, 2000).

Due to popularity of the standards published by IEEE, this definition has got the widest recognition. Still, Wiley (2002) argues that the IEEE's definition of learning object is inappropriate since it is too broad and too inclusive. As he explains, the definition does not exclude "*any person, place, thing, or idea*". Similarly, Polsani (2005) explains that the IEEE's definition is impractical since it does not make any distinction between physical, digital and conceptual entities.

Campbell (2007) analyses the opposing arguments of the IEEE's definition. He concludes that the definition has both its pros and cons. The broadness of the definition provides flexibility to apply IEEE standards to diverse resources. On the other hand, lack of constraints with regard to the meaning, size and format of a resource can lead to quite inconsistent use of standard.

Some researches use a narrower definition and consider only digital entities as learning objects. Lama (2001) defines that "*a learning object is any stand-alone, digital learning material that can be used and reused in technology*

supported learning environment". It should be noted that this definition sets additional requirements for an educational resource to be considered a learning object. Besides digital format, it is required for a learning object to be designed in a way to be reusable in different contexts. Todorova and Petrova (2003) additionally explain that a learning object should be built upon a single learning objective in order to provide composing larger educational units such as topics, lessons, chapters and courses.

A good overview of the concept of learning objects has been explained by (Jovanovic, Zizovic and Milosevic, 2012). The authors explain that a learning object should fulfil the following requirements:

- it is a stand-alone learning unit that can be reused in different contexts in order to achieve different learning objectives
- it can be aggregated in larger units such as lessons
- the semantics of a learning object, as well as its purpose and place within a bigger educational unit can be additionally described with metadata

Although the concept of learning objects appeared within the field of technology-supported learning, it should be focused both on the technological and instructional aspects. Boyle (2003) explains that only by combining multiple learning objects into a single pedagogical unit, the instructional purpose of a learning object can be satisfied. In this regard, he classifies learning objects into two types: simple and compound. The compound object consists of two or more simple objects. Although simple objects are reusable, a compound object can have a more significant pedagogical expressiveness.

With regard to the instructional aspects of learning objects, not all digital material used in education should be considered learning objects. L' Allier (1997) makes a distinction between learning objects and information objects. As the author explains, an information object contains a single piece of information, e.g. text paragraph, image or a web page. In contrast, a learning object must contain an instruction for using specific information.

Although the current trends promote the usage of small, reusable educational resources, we must be aware that the educational settings are usually gathered

around monolithic resources such as text books. Additionally, the idea of reusable educational resources has faced a lot of criticism. For example, Wiley (2002) states that learning objects that are primary designed to be reusable are less appropriate for automatic processing. The reason is that high level of reusability requires very small learning objects. With too granulized learning content, only humans could assemble them into a meaningful unit.

In general, there are no widely accepted technical methodologies for creating and assembling learning material (Ros, 2005). From this point we can hardly expect full reusability, accessibility and operability of learning content.

For this reason, this research is not focused exclusively to any specific kind of representing educational content. In a goal to cover as broad scenario as possible, we don't want to set any constraint with regard to learning content, purpose or the domain. In this regard, we consider educational resource as any digital or non-digital content that can be used in the learning process. In the rest of the text we are using the terms "learning object" and "educational resource" interchangeably.

1.2. Describing semantics of educational resources

The searchability of educational resources can be significantly improved by describing their semantics explicitly. This can be done by introducing metadata which give an additional description of educational resources (Laverde, Cifuentes and Rodríguez, 2007). Metadata are usually defined as data about data. This term is firstly used in the librarian community. It is used for any scheme that formally describes resources (Paunovic and Domazet, 2013).

Some of the benefits of using metadata as remarked in recent study (SREB-SCORE, 2007) are:

- The document representation is extended with a structured description that provide searching of objects based on their attributes
- The information are organized and classified in a more efficient manner
- The discovery of relevant information is facilitated

- The interoperability is improved when the semantics of document is clearly defined with metadata

With regard to the usage of educational resources metadata, IEEE specifies that metadata is "*information about an object, be it physical or digital. As the number of objects grows exponentially and our needs for learning expand equally dramatically, the lack of information or metadata about objects places a critical and fundamental constraint on our ability to discover, manage, and use objects.*" (Draft Standard for Learning Object Metadata, 2002).

Metadata provides the classification of learning objects since it represents a controlled taxonomy combined with the predefined vocabulary that must be used to describe the characteristics of learning objects (Boyle, 2003). Downes (2004) explains that the concept of metadata is inseparable from learning objects since the process of creation of a learning object actually consists of two tasks. The first task is designing the learning object itself, but altogether with this task, the annotation of the object with metadata should be done.

Roy, Sarkar and Ghose, (2010) list the benefits of tagging educational resources with metadata as follows:

- It facilitates search, retrieval, and use of learning objects
- In personalized learning environments, such as intelligent tutoring system, metadata can help retrieval of personalized learning objects for each user
- It promotes reusability of learning objects, since same learning objects can be used in different contexts through recognizing their semantics described with metadata
- The interoperability of learning objects is improved since they can be shared across different systems. Each system can process the learning objects based on their metadata

Information specified as metadata can be related to the physical characteristics (e.g. format, length etc.), object's classification, as well as the semantics of an educational resource. Besides its purpose, we can divide all metadata into two categories. The first category consists of metadata describing the object's

characteristics which are not related to the domain which the object belongs to. These metadata are general and can be applied to all learning objects beside their domain. Examples of such metadata are file format, author, language, etc.

The second category refers to metadata which use domain-specific information to describe learning objects. In many domains there have been developed classifications which categorize content in that particular domain. This section describes Computing Classification System and Mathematics Subject Classification, as two classifications specifically designed for the computer science and mathematics domain, respectively.

1.2.1. Domain-neutral metadata

In this subsection we present IEEE LOM and Dublin Core as two popular metadata sets used for describing the semantics of educational resources.

1.2.1.1. IEEE LOM

Learning Technology Standards Committee (LTSC), a group within IEEE, was among the first who recognized the importance of creating the standards for the e-learning domain. LTSC published the first real industry standard for e-learning, named Learning Object Metadata (LOM) standard.

IEEE LTSC Learning Object Metadata (LOM) is the standard for describing learning objects (IEEE, 2002). The fundamental idea of IEEE LOM specification is to define a minimal set of attributes necessary for a complete description, search and utilization of learning objects to allow these objects to be managed, located, and evaluated. Other popular e-learning specifications rely upon this standard considering the parts dealing with learning objects' description, i.e. SCORM (Advanced Distributed Learning, 2015), RDCEO (IMS Global Learning Consortium, 2002) or IMS Learning Design (IMS Global Learning Consortium, 2003).

IEEE LOM is used for annotating a learning object with metadata that defines and describes its characteristics. Introducing metadata into the representation of learning objects facilitates the discovery, retrieval and evaluation of learning objects. The applications that follow the standard are considered to be IEEE-conformant. The standard specifies the rules that an IEEE-conformant application must fulfill. The metadata specified by IEEE LOM

standard consists of various elements, whereby the standard defines element names, data types, possible set of values and value formats (e.g. field length) (IMS Global Learning Consortium, 2006).

Baker (2005) explains that the main objectives of the IEEE LOM standard are:

- Creating a well-structured description of learning resources and facilitating the participation of students and teachers in the learning process, as well as providing machine processing of educational content

- Helping developing learning objects in a standardized format, which should enable easier involvement of all stakeholders (i.e. educational institutions, government, companies, ...) into an educational system

- Providing combining LOM description with other specifications like Dublin Core, SCORM or IMS Learning Design

Additionally, other benefits of using LOM which are identified by IEEE LTSC and listed in (Campbell, 2007) are:

- encouraging creation of small, independent learning objects that can be used in different context

- enabling support for programmatic generation of personalized lessons that organized in accordance with a learning current knowledge, objectives or preferences

- making cost reduction of publishing and usage of learning resources by supporting non-profit, not-for-profit and for-profit distribution

- sharing and comparison of educational resources is facilitated through the widely recognized format as LOM. Stuempel et al. (2007) wrote that LOM is based on specifications such as IMS Learning Resource Meta-data (IMS Global Learning Consortium 2001), ARIADNE Educational Metadata Recommendation (ARIADNE, 2017). While it consider as the basis of another specifications as CanCore Learning Resource Metadata Initiative for Canada (Friesen, 2005), UK LOM

Core Profile based on LOM (Campbell, 2007), SCORM (Advanced Distributed Learning, 2017), and MERLOT(MERLOT, 1997)

The standard consists of two main parts. The first part specifies a conceptual model for the metadata, while the second part contains an XML schema used for representing data from conceptual model using the XML syntax. Duvel et al (2002) explain that LOM is a "multi-part standard" meaning that it represents semantic data model independently of its syntactical representation. The syntactical formalization can be considered as an independent standard developed as a specific binding of the LOM Data Model standard.

The data model standard, named 1484.12.1-2002, specifies the structure of a metadata instance for a learning object. The Institute of Electrical and Electronics (2002) clears that "*by specifying a common conceptual data schema, this Part of the standard ensures that bindings of Learning Object Metadata have a high degree of semantic interoperability that lead to transformations between bindings to be straightforward*".

The syntax binding for the data model standard is defined by 1484.12.3-2005 IEEE Learning Technology Standard - Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata. XML language is chosen for syntax formalization as "*a commonly used encoding, transfer, and occasional internal system storage mechanism for metadata*" (Riley, 2017). This allows system to expose API for retrieving stored metadata in the standardized and machine-readable format (Cebeci and Erdogan, 2005).

Many other education-related specifications allow for LOM metadata to be embedded within XML instances, such as: describing the resources in the IMS Content Package (IMS Global Learning Consortium, 2004) or Resource List (The University of Edinburgh, 2001); describing the vocabularies and terms in an IMS Vocabulary Definition and Exchange (IMS Global Learning Consortium, 2004) file; and describing the question items in an IMS Question and Test Interoperability (IMS Global Learning Consortium, 2015) file.

Besides XML, LOM metadata can be presented using the RDF format, to express and define some of the semantics by RDF binding. (Nilsson, Palmer & Brase, 2003). This format allows representing different conceptual models. It was originally developed as a metadata data model. Nowadays it is mostly used in the semantic web to represent web resources. Since the current trends

lead to usage of educational resources as digital documents in web environment, the support for RDF binding should improve the management of the educational resources in modern web-oriented systems.

With regard to the data model of IEEE LOM, there are nine main categories of metadata, namely General, Lifecycle, Meta-metadata, Technical, Educational, Rights, Relation, Annotation, and Classification). These categories are represented as root elements in the data model. The description of the categories is given in Table 1.1.

Name	Description
General	General information about learning object
Life Cycle	The information related to the learning object life cycle. The elements in this category describe object's history as well as its current state. For each event in the object's lifecycle, the entities involved in the event are recorded.
Meta-metadata	In contrast to other categories that describe learning object, this category describes metadata record itself
Technical	The technical aspects of a learning object, such as technical requirements for the usage of the object.
Educational	The main instructional and pedagogical information about using a learning object in the learning process
Rights	Intellectual property rights related to the usage of a learning object
Relation	Relations of a learning object with other learning objects are defined using the elements from this category

Annotation	Comments made about learning object during its lifecycle are recorded within this category
Classification	Using this element, the learning object can be categorized using an arbitrary classification system.

Table 1.1. The root elements of IEEE LOM

The LOM root categories consist of sub categories which gives 76 LOM elements in total. The hierarchy of LOM elements are illustrated in Figure 1.1. (Barker, 2005)

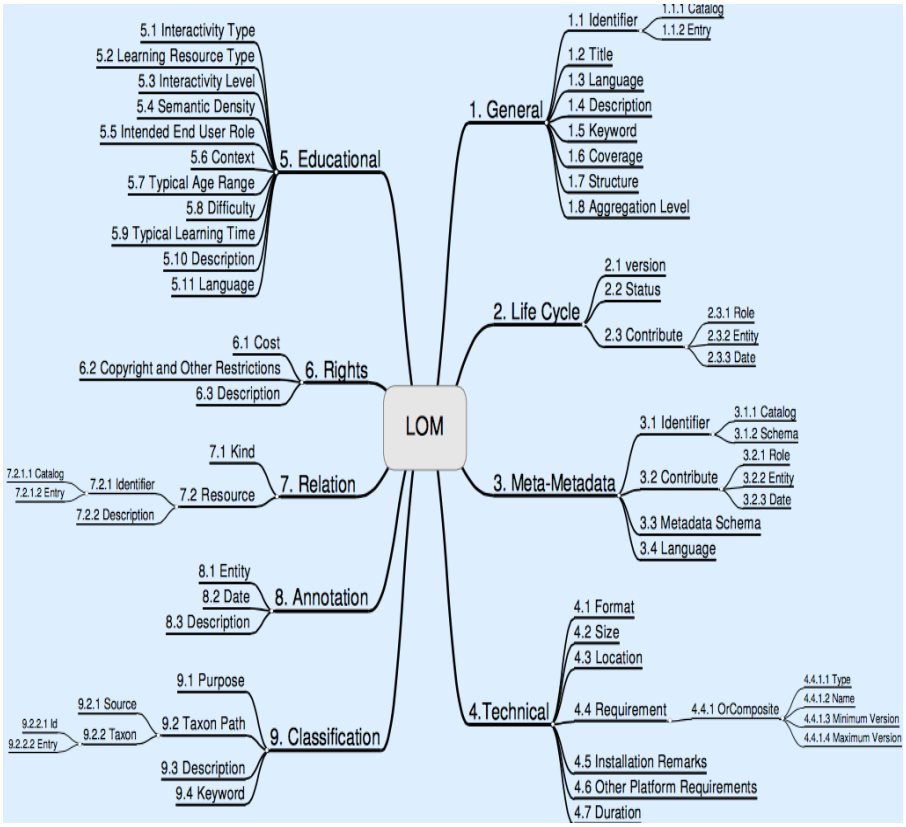


Figure 1.1. The IEEE LOM hierarchy (Barker, 2005)

For each element, name, datatype and value space are specified. An important feature of IEEE LOM specification is its subjectivity, meaning that all

elements are optional and developer can choose to specify the values only for the elements that are relevant for the particular scenario. As Duval and Hodgins (2003) explain that one can choose to store an information that a particular person recommends a particular LO (highly subjective metadata) instead of storing the metadata on the exact size of that learning object (highly objective metadata).

As illustrated in Figure 1.1, data elements are divided into two types:

- Aggregate elements which are represented as container elements. They consist of other data elements and do not have individual values. Such elements are *Identifier*, *Contribute*, and *Requirement*.

- Simple data elements which have individual values and they are leafs in the elements tree. Examples of the elements of this type are *Size*, *Location*, and *Version*.

Value space for elements specify the limitations on the possible values that can be assigned to the element. The following value spaces are supported:

1. String of Unicode characters

2. Language code (optionally accompanied by a country code)

3. Vocabulary – in this case the set of values is enumerated meaning that the value is limited to one of the values predefined in the vocabulary

4. IMC vCard 3.0 – the text that contains information commonly found on a business card

5. MIME type – the format of a resource, if the resource is given in the digital format

Listing 1.1. shows an example of the IEEE LOM document that describes a learning resource. The example contains metadata for the learning resource intended for learning databases.

```
<lom:lom>
   <lom:general>
      <lom:title>
         <lom:string language="en-GB">
            Semantic web
         </lom:string>
      </lom:title>
      <lom:language>en-GB</lom:language>
      <lom:description>
         <lom:string language="en-GB">
            Teaches the basics of Semantic web, on-
tologies and OWL language.
         </lom:string>
      </lom:description>
   </lom:general>
   <lom:technical>
      <lom:format>application/pdf</lom:format>
   </lom:technical>
</lom:lom>
```

Listing 1.1. Educational resource metadata in accordance with IEEE LOM specification

1.2.1.2. Dublin Core

The Dublin Core Metadata Element Set is a vocabulary of 22 properties for describing resources (Currier, 2008). The vocabulary contains broad and generic elements designed to cover a wide range of resources. Unlike IEEE LOM standard which is specifically designed to describe educational resources, Dublin Core is a more general standard that describes any resource by means of metadata. As stated in the specification, any content “having the identity” is considered a resource.

The components and constructs used in Dublin Core metadata are specified by Dublin Core Metadata Initiative (DCMI) Abstract Model (Powell et al., 2007). Figure 1.2 shows how a resource described by Dublin Core metadata set is represented using this model.

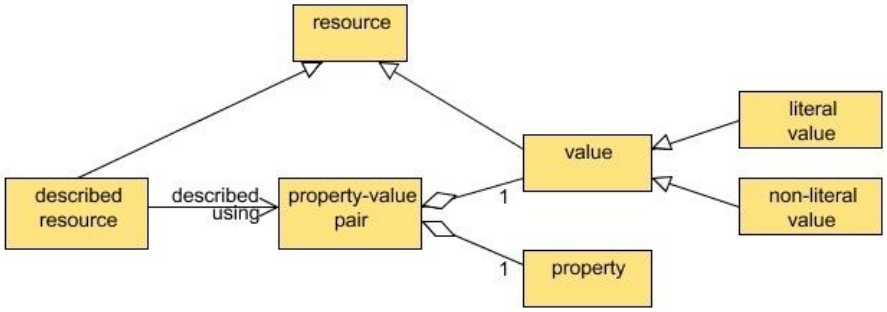


Figure 1.2. DCMI resource model (Powell et al., 2007)

A resource is described using one or more property-value pairs. It should be noted that the value of the property is also considered a resource. There are two categories of values. If the value refers to some physical, digital or conceptual entity, it is called non-literal value. The second category (literal value) represents simple values presented as strings, numbers or dates.

Due to its generality, Dublin Core may be used for different purposes. The original version of the specification contains 15 elements. Later, the specification was refined to contain 3 more elements. The elements are described in Table 1.2.

Name	Description
Title	The name of the resource
Creator	A person or an entity (e.g. an institution) that created the resource or (in case of multiple authors) is mainly responsible for the creation of the resource content.
Subject	Topic of the resource content. For this element, it is recommended to use a limited set of values taken from some vocabulary. This element serves for the resource classification. For that reason, using some standard classification system is recommended.

Description	More detail information on the content of the resource
Publisher	A person or an entity responsible for publishing the resource to be available
Contributor	Entities that contributed to the creation of the learning resource content
Date	Date of the resource creation. In addition, this element can refer to other events in the resource lifecycle, such as publishing date.
Type	The type of the resource. It should refer to the resource category, function or genre (not resource digital format). The usage of a controlled vocabulary is recommended.
Format	The type of the resource representation (physical or digital). This element should describe resource technical characteristics, such as media format, length or specific technical requirements for usage
Identifier	A unique identifier of the resource. The identifier should follow the identification system used in the particular context where the resource will be used. For example, URL for web resources, DOI for digital documents or ISBN for books.
Source	A unique identifier of a resource that this resource is derived from
Language	Language of the resource content. Standard language codes according to ISO 639 are recommended, e.g. en, en-GB, sr.

Relation	A unique identifier of a resource that this resource is related to
Coverage	The scope of the resource content. As noted in (Currier, 2008), this element can include spatial location (place or geographic coordinates), temporal period (year or date range) or an administrative entity that has jurisdiction on this resource. The values should be taken from controlled vocabularies.
Rights	Intellectual rights on the resource. The element specifies information such as Intellectual Property Rights, Copyright and various Property Rights. The element itself contains this information or references another service that provides such information.
Audience	Persons or entities that the resource is intended to.
Provenance	This element records all changes made on the resource during its lifecycle with regard to the ownership on the resource.
Rights Holder	A person or an entity that holds the ownership on rights over the resource. The owner should be uniquely identified. The recommendation is to use some global identifier such as URI

Table 1.2. Dublin Core elements

All Dublin Core elements are optional. Due to its simplicity, Dublin Core was relatively early adopted in the community. Since it is designed as a general metadata model, it has been used in various domains. In this research, we are mainly focused on using Dublin Core to represent characteristics of educational resources.

Since Dublin Core elements can be applied to any resource, they are appropriate for describing educational resources. Still, the basic Dublin Core model does not have specific elements related to the educational domain. There are ongoing efforts to include education-specific elements in the Dublin Core specification. The goal of Dublin Core Metadata Initiative is specifying metadata that lead to support cross-domain resource discovery on the Internet (Weibel and Koch, 2000). The community is developing the DC-Education Application Profile. Dublin Core Application Profile (DCAP) is a framework for extending basic Dublin Core model with domain-specific elements to design metadata applications for maximum interoperability and reusability (Nilsson , Baker & Johnston, 2008). More details on DCAPs are given in the next subsection. In this part of the text we will present the current state of the ED-Education Application Profile project. Although, the project is still not completed and there is no the final version of this profile, it should be noted which elements the profile is intended to contain. Figure 1.3 (Currier, 2008) presents the profile elements.

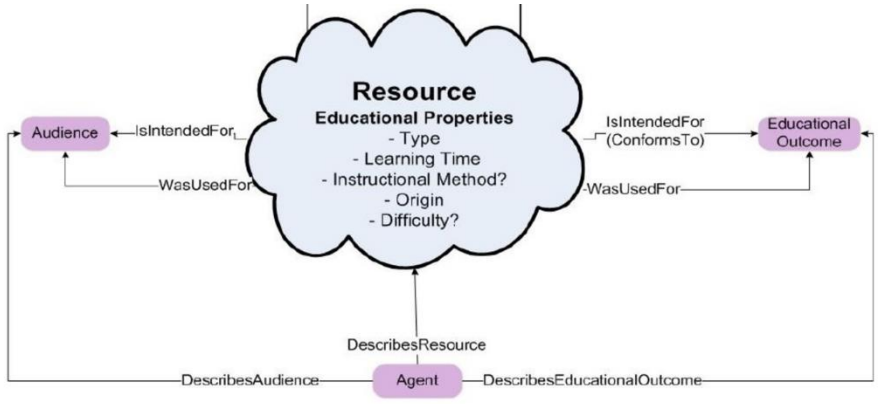


Figure 1.3. DC-Education Application Profile (Currier, 2008)

It should be noted that DC-Education Application Profile is not intended to describe educational resources in any particular domain. Although it extends the basic Dublin Core model with educational-specific elements, it still covers all educational resources besides their domain. In that context, the metadata model represented using this profiles stands at the same abstraction level as IEEE LOM.

The main requirements set for DC-Education Application Profile were to support resource discovery, educational use of resources and profile extensibility. The authors of the profile identified four groups of properties that should find the place in the profile. The first group contains elements that refer to educational resource audience. Different elements are created for this purpose, such as audience educational level, age range, language that intended users speak, etc.

The elements that relate to the instructional aspects of educational resource belong to the second group. Some of the elements from this group are instructional method, typical learning time and resource type (in the meaning of its educational purpose, not physical representation). The third groups relates to learning outcomes. This group contains elements such as learning goal and course that the resource was used in or created for. The last group contains the elements that refer to the student feedback on using resources. Elements that contain user comments, reviews and ratings are added in this group.

With regard to the profile structure presented in Figure 1.3, the above mentioned elements are defined within the *Resource*, *Audience* and *Outcome* concepts. For information stored within the model, the profile allows specifying an authority that entered the particular information. This is done by using the concept of *Agent.* Also, regarding the resource usage, the profile separates intended plan from its realization. For this reason, there are two kind of relationships between concepts. *IsIntendedFor* models the planned purpose, while the *WasUsedFor* relationship describes the actual usage of the resource.

Just like IEEE LOM, Dublin Core model can be represented in different syntax. The most widely used syntax bindings are those for XML and RDF. An example of an educational resource described by Dublin Core metadata represented in XML syntax is shown in Listing 1.2 .

```
<dublinCore>
   <title>Eclipse environment screenshot</title>
   <author type='teacher'>John Smith</author>
   <subject scheme='gmgpc'>Web programming</subject>
   <objectType>image</objectType>
   <form scheme='IMT'>image/jpeg</form>
   <identifier type='URN'>
      rs.ac.uns.ftn.kzi.wp/peo
   </identifier>
</dublinCore>
```

Listing 1.2. Educational resource metadata in accordance with Dublin Core specification

1.2.2. Domain-specific metadata

In this subsection we discuss representing domain-specific metadata about an educational resource. There are different ways to include this type of information in the resource metadata:

1. Domain-neutral metadata sets can be customized for a particular domain or

2. New metadata sets for a particular domain can be designed

In the following text we present both approaches. We illustrate designing new domain-specific metadata sets by presenting ACM Computing Classification System and Mathematics Subject Classification, as two specifications that describe educational resources from the computer science and mathematics domain, respectively.

1.2.2.1. Customization of domain-neutral metadata sets

As we mentioned, the first approach for describing domain-specific semantics of an educational resource is to adapt some domain-neutral metadata set to a particular domain. This is possible with two domain-neutral metadata sets described in the previous subsection. Both IEEE LOM and Dublin Core provide the creation of application profiles. Application profile has been defined as "*an assemblage of metadata elements selected from one or more metadata schemas and combined in a compound schema*" (Duval et al., 2002).

In case of IEEE LOM, application profiles consist of subsets of the basic LOM elements compound with the best practices and recommendations for the usage of the elements. By limiting the whole model to a specific subset, only elements relevant in a particular domain will be used. Various application profiles for IEEE LOM have been developed so far.

CanCore is a subset of LOM designed to promote better adoption of IEEE LOM by focusing on the most important elements only to facilitate the indexing of learning objects (Friesen, 2005). Among 76 elements that IEEE LOM originally contains, CanCore recommends usage of 39 elements only. Within CanCore MetaData Initiative, the set of best practices and recommendations for the elements usage is identified. UK LOM Core (Campbell, 2007) is another application profile for IEEE LOM which is designed to optimize LOM for UK educational system. It contains guidelines how to use LOM elements in the context of British education. The whole set of elements is classified into three groups that indicates is an element mandatory, optional or optional-recommended, respectively. Also, there is ANZ-LOM (Education Services Australia, 2011) as LOM application profiles for Australian education. It specifies region-specific vocabularies that should be used in Australian educational resources for the values of the LOM elements. Similarly, the application profiles for other countries, such as France, Netherlands and Greece, have been developed so far.

Obviously, such an approach with creating subsets of the existing metadata model does not introduce any additional information. Such information can be included in IEEE LOM by extending it. For example The Learning Federation Schools Online Curriculum Content Initiative (2002) have presented a new conceptual scheme based on IEEE LOM. The schema extends LOM by including new elements, as well as incorporating elements from other popular specifications such as Dublin Core.

With regard to Dublin Core, it allows creating application profiles which represent arbitrary data models that are added as an extension to the basic Dublin Core metadata model.

Coyle and Baker (2009) explain that Dublin Core application profile should specify:

1. Functional Requirements - the purpose of introducing the application profile

2. Domain Model – concepts and relationships between them that are described with this additional metadata set

3. Description Set Profile and Usage Guidelines – specification of metadata elements, as well as rules for their use

4. Syntax Guidelines and Data Formats – the syntax binding for representing information stored within the model

The process of creation of a new Dublin Core application profile is systematized in September 2007 at the International Conference on Dublin Core and Metadata Applications in Singapore, resulting in the Singapore Framework for Dublin Core Application (Nilsson , Baker & Johnston, 2008). The framework specifies the components necessary for documenting an Application Profile. The components are shown in Figure 1.4 (Coyle and Baker, 2009).

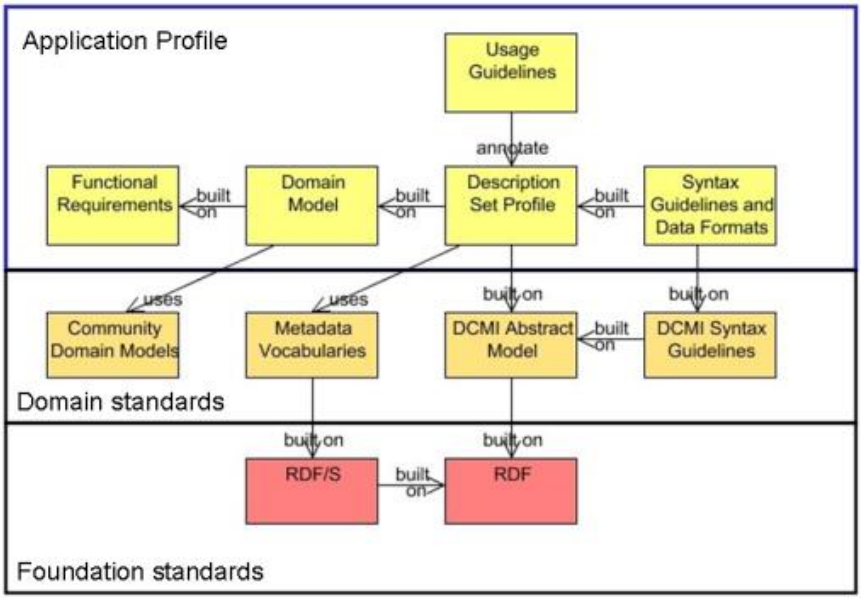


Figure 1.4. Singapore Framework for Dublin Core Application Profiles (Coyle and Baker, 2009)

We can notice that the above mentioned four components of an Application Profile are represented in the framework. The first step while creating an Application Profile is defining the functions that the profile should support. Also, in this phase, the functions that are out of the scope of the profile should be identified. The domain model is an entity-relationship model built on previously defined requirements. It should be created using the existing models developed within the community of that domain. The domain model is then converted to the exact elements of the application profiles. The elements are defined in accordance with the above described DCMI Abstract Model. The element values should be chosen from the vocabularies that are established in the domain. If the Application Profile requires binding to some specific syntax, which is different from the already supported syntax bindings for the basic Dublin Core model, the application-specific bindings can be specified as the last step. Additionally, usage guidelines can describe how to use the properties from Application Profile for representing resources from the domain.

1.2.2.2. ACM Computing Classification System

ACM Computing Classification System (Association for Computer Machinery, 2012) is used to classify content in the field of computer science as a poly-hierarchical ontology that can be utilized in semantic web applications. The classification has been designed in order to facilitate the search for related literature, ACM's Digital Library and other online resources. The first version of the classification appeared in 1964. Since then, the classification has gone through multiple revisions. The last one, made in 2012, use a new poly-hierarchical ontology appropriate to be used by semantic web applications. The old letter-and-number coding system is no longer being used from this version.

The classification describes content using (maximum) five levels of hierarchically organized categories. The full list of categories is publicly available at (Association for Computer Machinery, 2012). The categories reflect the state of the art of the computer science domain. The top level categories are *General and reference*, *Hardware*, *Computer systems organization*, *Networks*, *Software and its engineering*, *Theory of computation*, *Mathematics of computing*, *Information systems*, *Security and privacy*, *Human-centered computing*, *Computing methodologies*, *Applied computing*,

Social and professional topics, and *Proper nouns: People, technologies and companies*.

Table 1.3 presents an example of an educational resource described by the ACM classification.

Name	An Introduction to the OWL Web Ontology Language
URI	www.cse.lehigh.edu/~heflin/IntroToOWL.pdf
ACM level 1	Information systems
ACM level 2	World wide web
ACM level 3	Web data description languages
ACM level 4	Semantic web description languages
ACM level 5	Web Ontology Language (OWL)

Table 1.3. Educational resources classified by ACM Computing Classification System

1.2.2.3. Mathematics Subject Classification

Mathematics Subject Classification MSC (Narayan, 2010) is an alphanumerical classification scheme which is based on the two major mathematical reviewing databases, Mathematical Reviews (MRDB) and Zentralblatt MATH (ZMATH). The classification includes all the branches of both pure and applied mathematics, i.e.: probability and statistics, numerical analysis and computing, mathematical physics and economics, systems theory and control, information and communication theory. (De Robbio, Maguolo and Marini, 2002).

After its first version in 1940, MSC has undergone many corrections and additions. The main objective of this classification of items in the mathematical domain is to help users finding the items of potential interest to them from anywhere that makes use of this classification scheme.

The MSC structure consists of three levels. Not all levels are mandatory. An MSC record can be two, three or five digits long, depending on how many levels of the classification scheme are used. The classification has been designed so that the items from the MRDB and ZMATH databases were coded by alpha-numerical values. Each item was mapped to one primary classification representing its principal contribution. If an item contributed to different areas, the most important contribution is chosen.

First level identifies the main mathematics area that a subject belongs to. The categories in this level represent five main mathematical areas encoded by two digit numbers, as shown in Figure 1.5 (zbMATH, 2010).

Figure 1.5. First-level categories of Mathematics Subject Classification (Zbmath.org, 2017)

These five main areas are:

1. General/foundations (starts from 00)
2. Discrete mathematics/algebra (starts from 05),
3. Analysis (starts from 26),
4. Geometry and topology (starts from 51)
5. Applied mathematics (starts from 60)

Second level categories further classify the categories from the first level. The categories of the second level are encoded using a single letter from the Latin alphabet or a special second level code. The second level code is given in a form of "-xx", where xx represents two digits.

An example of representing second level categories for the *Group theory and generalizations* top-level category is shown in Table 1.4.

Code	Title
20-00	General reference works
20-01	Instructional exposition
20-02	Research exposition
20-03	Historical
20Bxx	Permutation groups
20Cxx	Representation theory of groups
20Dxx	Abstract finite groups

Table 1.4. An example of the MSC second level categories

Third level contains the most specific categories, usually corresponding to a specific research area. Table 1.5 illustrates the third-level categories for the second-level category *Representation theory of groups*.

Code	Title
20C34	Representations of sporadic groups
20C35	Applications of group representations to physics
20C40	Computational methods
20C99	None of the above, but in this section

Table 1.5. An example of the MSC third level categories

1.3. Repositories of educational resources

In this subsection we give a brief overview of different software systems that provide availability and management of educational resources. Numerous systems for this purpose, commonly named as learning object repositories, have been developed so far. These systems are mostly implemented as public internet repositories which store educational resources and/or metadata about them.

MIT OpenCourseWare (OpenCourseWare, 2001) is such a repository which allows free download of learning material used in courses from Massachusetts Institute of Technology. MIT was a pioneer in the job of publishing all the university courses online. Its initiative influenced other similar projects from the universities all across the globe.

Similarly, Carnegie Mellon University within its project Open Learning Initiative (OLI) (Oli.cmu.edu, 2017) provides open access to the learning material from the electronic courses taught at this university. The material cannot be downloaded, but it is accessible online within the web application of OLI University.

The Open University within its project OpenLearn (OpenLearn, 2017), it has developed a repository with more than 1000 courses from this university. Besides courses learning material which can be accessed on the project website, a user can download the complete course in different formats.

A large number of institutions have a local repository which provides public access to their courses. OpenCourseWare (OpenCourseWare, 2001) gathers such local repositories into a single global repository. This repository currently offers courses from 79 institutions on 26 languages. The repository does not actually store course material, but rather contains a collection of links to external resources stored within various online repositories.

Besides previously mentioned repositories related to formal educational institutions, there are also independent repositories.

MERLOT (Merlot.org, 1997) is a free and open online community that gathers educational institutions, teachers and students within the field of higher education. It is independent of any particular educational institution. They are focused on promoting the community for sharing learning material and experiences. The community has been developed a large number of educational resources. The resources are evaluated by the community members. With regard to the format of educational resources, there are no any specific constraints.

Curriki (Curriki, 2017) is another independence repository. It is focused on the primary and secondary education. The main goal of the project is to provide equal educational opportunities beside the geographic location through allowing public access to the high quality learning material. The material can be published as text, images, audio or video files.

OpenStax (Cnx.org, 1999) is another popular repository of educational resources. The repository is intended for all kind of courses regardless of the age level or the domain. The learning content is represented using textual and multimedia formats. Also, the complete courses can be downloaded as a textual document or an archive file.

Another form of providing learning material to a wide audience, especially popular in recent years are Massive Open Online Courses (MOOC). MOOCs replicate classical educational settings in the online environment, meaning that

besides learning material they support scheduled lectures, assessments, as well as communication and collaboration among participants. In that regard, MOOCs tend to be much more than just repositories of learning content. Instead, they are trying to cover all the educational activities, such as evaluation, collaboration etc.

Some popular MOOC repositories nowadays are edX (edX Inc, 2012), Coursera (Coursera Inc, 2017). Udacity (Udacity Inc., 2011) and Udemy (Udemy Inc., 2010).

1.4. Model-driven engineering

The software architecture proposed in this research relies on the techniques of the model-driven engineering (Kleppe, Warmer and Bast, 2003). In short, this technique relies on the development of domain-models which describe system data and behaviour at different abstraction levels (Brambilla, Cabot and Wimmer, 2012).

1.4.1. The concepts of the model-driven approach

This approach to software engineering is commonly named the Model-driven approach (MDA). MDA is "*a way to organize and manage system architectures; it is supported by automated tools and services for both defining the models and facilitating model types*" (Brown, 2004).

The MDA supports different types of application and platforms by convert platform independent model to platform specific model. Figure 1.6 shows the basic principles of the model-driven engineering.

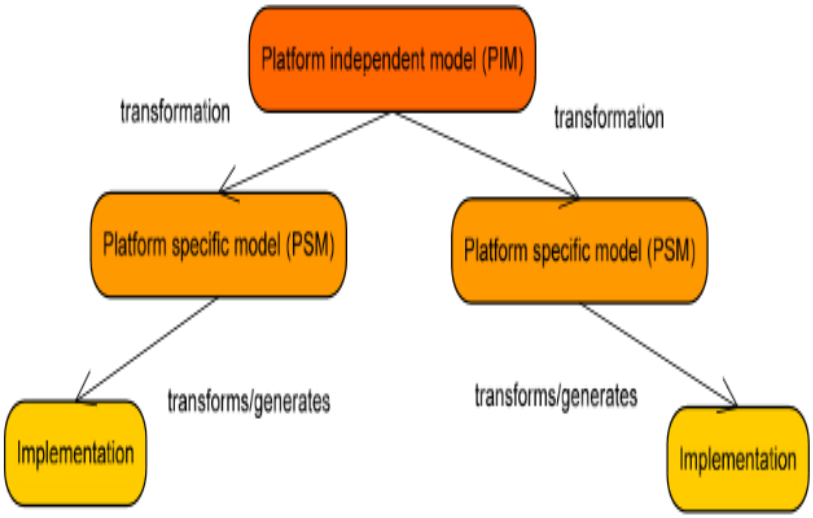


Figure 1.6.Model-driven architecture (Bizonova, 2007)

This approach to software development uses source domain models as a basis for automatic generation of target models. The source model describes system structure and behaviour at different abstraction levels (Brambilla, Cabot and Wimmer, 2012). As shown in the figure 1.6 (Bizonova, 2007), the process of developing an application using the model-driven approach starts with developing a platform-independent model (PIM), which is usually called the source model. The source model is considered as platform-independent since it is focused on representing the domain knowledge and does not deal with the application implementation details. A system may be defined as a platform-independent model through a Domain Specific Language. For example, Poole (2001) noted that the platform-independent models are initially expressed in a platform-independent modelling language, such as UML.

The PIM may then be translated to one or more platform-specific models (PSMs) for the actual implementation, using different Domain Specific Languages, or a General Purpose Language like Java, C#, Python, etc. The translations between the PIM and PSMs are normally performed using automated tools, like model transformation tools (McKay, 2017). Hence, the source model, as an abstract representation of the system, serves as the basis for the programmatic generation of the final software application. The final software application represents the target model which is obtained by

transforming the source model using a set of transformation rules. The target model is platform-specific and contains the source code of the software product.

On this way, instead of being focused on algorithms and programming, the developer creates an application by designing a source model that represents the domain knowledge. After the definition of the transformation rules, the target model which represents the final software application can be automatically generated. Such programmatic generation can increase the productivity, as well as provide system portability, since based on the same domain model, different platform-specific target models can be generated. Also, separating the representation of the domain knowledge from the implementation details facilitates the involvement of domain experts in the software design phase.

Some important principles of MDA have been remarked in a recent study by Brown (2004):

1. Expressed models must be introduced in a well-defined notation for enterprise-scale solutions

2. A system must rely on a set of models which will be transformed into other models

3. To facilitate meaningful integration and transformation among models, a formal underpinning to describe models in a set of metamodels is required. Goede and Irizarry (2008) stated that "*By describing these models through a set of meta- models, transformation amongst models is facilitated, ultimately resulting in code generation*".

In general, there are three different types of source models asserted in previous study by (Goede and Irizarry, 2008):

1. The model contains the business specifications

2. the model that represents the high level details of the platform

3. The model includes technical details of the target platform.

The MDA approach introduces a conceptual framework and standards to express models, model relationships, and model-to-model transformations. Various standards are providing the foundation for MDA. Friesen (2003) enumerates some of them Unified Modeling Language (UML), followed by related standards which are (Meta-Object Facility (MOF), XML Metadata Interchange (XMI), MOF Query/Views/Transformation (QVT) and Model to text transformation language (MOFM2T)).

With regard to the abstract nature of a source model and the implementation details contained in a target model, three main ideas are highlighted by Brown (2004):

- Model classification – the source models can less or more explicitly represent the aspects of the target platforms. This can be the criteria for the classification of different source models. Some models include information on constraints that must be followed with regard to the target platform (such as hardware requirements, multilanguage supports, etc.), while other models can stay focused on the domain-models with leaving the implementation details of the target platform out of the scope

- Platform independence – the "platform" can be quite ambiguous term and can be considered in different ways when setting the "platform independence" as a development goal. Platform can be the complete final software application, but also the software environment where the application runs. In the second case, the operating system or virtual machine environment, such as Java Virtual Machine or .NET Common Language Runtime, are considered a platform.

- Model transformation and refinement – the model transformation is the most important part of the model-driven approach. The benefits that MDA can bring to the software development strongly relates to the definition of transformation rules between source and target models. By following the best practices and conventions in designing a source model, as well as choosing a widely accepted formalism for representing this type of model, the transformation can be done in more straightforward way. Karakostas and Zorgios (2008) gives an example of expressing source models in UML and implementations in J2EE.

The transformation between these two models can be done by "*well-understood UML-to-J2EE transformation patterns that can be consistently applied, validated, and automated*".

A modeling paradigm is effective if its models make sense from the point of view of a user who has already been familiar with the domain, and also with implementing systems. The models are developed through extensive communication among product managers, designers, developers and users of the application domain. As the models approach completion, they enable the development of software and systems. Since the developers realized that the modelling process is important to the success of every enterprise-scale solution, via the transformation process from model to model and from model to code, MDA guarantees numbers of advantages.

Alhir (2003) explains that "*MDA applies platform-independent models and platform-specific models to sustain and leverage investments in requirements, technologies, and the lifecycle that bridges the gap between them as they independently change. Such an approach generally leads to long-term flexibility of implementations, integration, maintenance, testing and simulation as well as portability, interoperability and reusability*."

Poole (2001) indicated that "*The MDA has significant implications for the disciplines of Metamodeling and Adaptive Object Models (AOMs). Metamodeling is the primary activity in the specification, or modelling, of metadata. Interoperability in heterogeneous environments is ultimately achieved via shared metadata and the overall strategy for sharing and understanding metadata consists of the automated development, publishing, management, and interpretation of models*".

Bizonova and Ranc (2007) asserted the benefits that developers can get by using MDA:

- increasing flexibility by separating model design decisions from decisions related to the platform implementation. On this way, the further changes in the model design are facilitated

- the communication with domain experts and end-users is improved since it is gathered around the abstract source models which hides the implementation details that they are not familiar with

- MDA promotes using standard specifications languages so that the final software platform can be generated in a programmatic way

Moreover, Goede and Irizarry (2008) enumerate another benefits for developers and business leaders when using MDA:

- MDA reduces the development costs, as well as the time needed for new applications

- The systems are better adapted for different platforms, since different platform-specific models that fits the needs of a particular platform can be developed

- The existing systems can be modified more efficiently by introducing changes into the source model which are then programmatically propagated to the target model

- MDA enables development of industry specific applications that reflect the requirements of a particular domain. These requirements are represented within the source model

- MDA enables all the participants in the software development process to use languages and concepts they are familiar with. Each participant is involved in the specific development phase depending on his/her role in the team. E.g. the domain experts will be involved in designing the source model by using a notation easily recognized by domain experts. Such an approach improves the communication and integration within the team.

Some extra benefits of using MDA are noted by Brown and Conallen (2005), namely:

- MDA promotes use of best practices, design patterns and commonly accepted architectural designs

- MDA makes the development more predictable, since it is based on iterative source-to-target transformation cycles

1.4.2. Model-driven approach in practice

With regard to the model-driven engineering techniques used in the application development process, it has been widely used so far. Fowler and Parsons (2011) gives an overview of Domain Specific Languages (DSL). DSLs formally describes concepts from a particular application domain. In the model-driven approach, DSL usually serves as a source model in generating a final software solution. DSLs enable specifying different application features, such as data model, dynamic behaviour, custom operations, usage constraints, etc. Consequently, a DSL can have a complex structure which can be handled by experienced programmers only.

For that reason, some researches tried to get process of designing domain model closer to domain experts. This can be done by abstracting application technical details from them through representing domain model visually. Different solutions in using diagrams when working with DSLs can be found by (Cook et al., 2007; Dejanovic et al., 2010; Nguyen, Qafmolla, and Richta, 2014).

Another solution is using application mockups (wireframes) for representing application model and features (Rivero et al., 2014). Given that on this way the domain model is represented through a graphical interface prototype, domain specialists with no technical knowledge are involved in the model design phase more easily. In this approach, the application mockups must be given in the format appropriate for further processing. Researches presented by (Buchmann, 2012; Coyete et al., 2007; Coyete and Vanderdonckt, 2005; Plimmer and Apperley, 2004) deals with converting hand-drawn mockups into format that represent them unambiguously.

Besides the formal method for representing domain knowledge, it serves as a basis for generating a final application. Different code generators has been developed so far. Recently, a popular code tool for code generation is Yeoman (Yeoman, 2017). It facilitates the process of creating a new application by automatically setting up file structure, build process, application dependencies etc. Yeoman is neutral with regard to the technology used in a final application. Rather, different code generators that support generating application in particular technologies can be used on top of Yeoman.

The Yeoman generators can be classified in two groups, depending on whether a generator deals with domain-specific application data. Generators from the first group do not take domain-specific data as input, but rather they extend basic Yeoman features to generate the project structure which is more appropriate for a particular technology. The advantage of this type of generators is that they are usually simple for use and require just a single string as input, since they do not need domain model for code generation. Still, the generated application is not runnable, since it lacks domain data. An examples of this type of Yeoman generator is Angular generator (Yeoman, 2017).

The second group of generators receive domain model as input. Based on the domain model, the fully functional application can be generated. The generator produces code that works with domain data, as well as with some general application features, such as security and logging.

JHipster (Jhipster, 2013) is a very popular generator of this type. The disadvantage of this approach to generating software application is that an input point is application data model. This means that the visual behaviour of the application is not specified in the model. Such an approach usually leads to generic and uncustomized user forms. Also, as explained above, domain specialists are involved more easily if they can design application mockups, instead of designing the data model directly.

Generators that work with the Yeoman tool lacks customization during the code generation. The application code that will be generated is mostly predefined which means that all the generated applications will look the same.

Generic engines are designed to provide more flexible solution enabling programmers to customize how the final application will behave (Cerny, 2013). This is especially important in the enterprise systems that must answer to strict demands with regard to the application performance, as well as user interface. Using generic engines facilitates the adoption of agile software development techniques. The need for constant changes in the application design, makes static and inflexible code generators inappropriate.

1.5. Research motivation and goals

In general, software systems for managing educational resources face two challenges. At first place, they should support some general metadata set to

allow management of educational resources that belong to different domains. On the other hand, it is necessary to describe parts of resources' semantics which are domain-specific. It can be quite difficult to implement a software application which contains predefined metadata sets for various domains. Even if such application with various initially supported metadata sets would be implemented, still there is a problem with further adding of new metadata sets. With statically defined metadata sets, an application would not be appropriate for modifications of existing metadata sets as well as for describing the semantics of educational resources in an entirely new domain.

The goal of this research is to enable dynamic customization of metadata sets that a software application for the management of educational resources supports. The idea of the research is revolving around allowing users to define metadata sets on their own. In this way, a user can customize the application for describing educational resources semantics in his/her domain. Since users are mostly unskilled for developing an application for managing educational resources according to their models, our goal is to provide an executable platform which would generate the final software application programmatically. Such solution ensures that users will be able to manage educational resources using the semantics from the specific domain with no need to develop or order a new software application. The executable platform would then provide a dynamic adaptation of metadata sets to the required domain.

In accordance with the above mentioned, we formulated the hypothesis of our research: In order to provide the management of learning objects which are described by non-predefined domain-specific metadata sets, it is necessary to create a system which can be easily adapted to manage learning objects in the specific domain. It is possible to fulfill such requirement by implementing a support for adding different domain-specific metadata models dynamically.

To achieve the research goal, we use the model-driven approach. As explained, this approach enables generating target data model based on formally defined source models. With regard to the implementing model-driven approach in the management of educational resources metadata, the source model is a domain model of the specific metadata set, while the target model is the programmatically generated application for managing educational resources according to the specified metadata set. The result of this research is an

executable platform for generating software application that provides management of educational resources described by customizable metadata sets. The platform is an extension of Kroki prototyping tool (Kroki team, 2018b) which has been built on the principles of model-driven engineering.

The rest of the text is organized as follows. The next chapter introduces the Kroki tool which was a basis for implementing the model-driven approach in this research. We present the architecture and main features of Kroki.

Chapter three is the main part of this text. It represents our platform for managing educational resources. We present the created models of educational resources, as well as the generated software application that provides the management of educational resources in conformance with the created models.

Chapter four is the case study. It presents an experiment we conducted in order to verify the characteristics of the proposed software platform.

Concluding remarks will be given in the last chapter of the dissertation. We will analyse pros and cons of the proposed solution. Finally, we will present possible improvements of the solution, as well as general guidelines for future work within this research.

2. Kroki tool

As a basis for implementing a model-driven approach in the management of educational resources metadata, we decided to use Kroki software application (Kroki team, 2018b), which is described in this section.

Kroki is an open source executable platform for generating prototypes of software applications based on domain models. Its source code is available on (Kroki team, 2018a). It is a prototyping tool which enables the cooperation of different roles that participate in the development of business information systems. The main idea behind Kroki is involving end-users in early phases of the software development process. Such an approach should ensure better communication with end-users in the requirement analysis phase, resulting in less changes in later phases. The end-users are involved by enabling requirements elicitation based on executable prototypes, using the means familiar to the end users - drawing user interface (UI) mockups. On this way, an application prototype is available to a user during the requirement analysis phase which increases the comprehension what the final application will look like.

Using mockups in the early development stages is not a novel approach by itself. There are different techniques to manage the Software requirement such as UML models, task analysis, and prototyping for fulfilling the needs of users and their environments. Technological solutions allow the creation of mockups. With a specific tool it is very easy to create mockups tools (Tiexeira et al., 2014) to sketch the user interface of a final application. Filipović et al. (2017) have explained that the mockups are then manually or semi-automatically transformed to UI elements of the software application which is developed. In contrast to that approach, mockups created by Kroki represent the View component of the final application, meaning that sketching of mockups creates the application user interface directly. In that context, Kroki mockups have a twofold aim. Besides being used for the requirements elicitation, Kroki mockups have important part in the design and implementation phases as well given that they are basis for the automatic generation of the final application.

2.1. Architecture

The components of the Kroki architecture are presented in Figure 2.1 (Filipović et al., 2017).

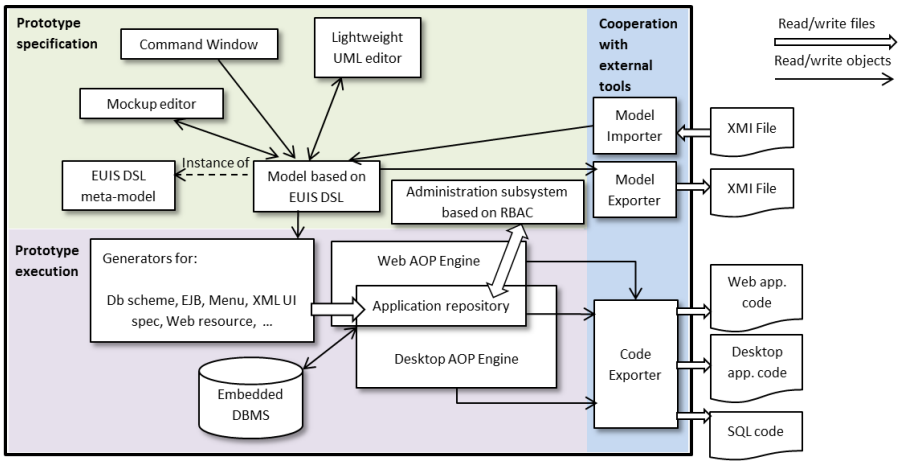


Figure 2.1. Kroki tool components (Filipović et al., 2017)

The architecture has been designed with taking development agility and reuse as two main goals. The chosen model-driven approach ensures agility by providing programmatic generation of a final software application. One can get a fully-functional application with no need to develop it. Still, it is necessary to design the application model. In order to save up user's time, Kroki provides a convenient graphical interface which enables making model in an efficient way, even for the unskilled users. In addition, the model artifacts can be reused across different models to reduce development time. Finally, Kroki has a built-in support for configuring authentication and authorization in the generated application, which is one of the common features needed in any business application. By providing this feature in a flexible and configurable way, Kroki saves the time needed for its manual implementation from the scratch.

The ability of exporting and importing application models ensures that the model created in Kroki can be reused in other similar applications, as well as that the Kroki application model can be created on the basis of an imported model which is created by another software tool.

Given that the Kroki tool is an executable platform for designing application models as well as generating final applications based on these models, it consists of two main components that are responsible for prototype specification and execution, respectively.

The central part of the prototype specification component is the representation of application mockups. The mockups represent the model of a business information system. With regard to the formal representation of Kroki mockups, Kroki uses EUIS (Enterprise User Interface Specification) DSL (Domain Specific Language) for specifying user interfaces (UIs) of enterprise applications at a high-level of abstraction (Perišić et al, 2011). The elements supported by EUIS DSL are specified by its meta-model (EUIS DSL meta-model component in the Figure 2.1.).

EUIS DSL has three concrete syntaxes, namely:

- mockup-based graphical syntax – represented by the Mockup editor component
- UML-based graphical syntax – represented by the Lightweight UML editor component, and
- textual syntax – represented by the Command Windows component

By supporting different formats for the application model, Kroki enables users to design the model with respect to their personal backgrounds and preferences.

Kroki's mockup editor provides graphical syntax for specifying application elements. It allows users to visually arrange application panels by specifying data contained on the panels, as well as their presentation layout. This kind of syntax is primarily intended for users without the background in information technologies, but can also be used by software specialists to make their work more efficient. Within the mockup editor, users manipulate graphical components, whereby each component corresponds to a particular element from the EUIS DSL meta-model.

Besides sketching mockups of the application forms, the model of a business information system can be specified by using Kroki's lightweight UML editor

which enables users to create application form. It can be executed by using Kroki's desktop or web engine (Kaplar et al., 2015). The exact graphical notation is similar to the one used for UML class diagrams, enabling users to define elements of a business information system and establish relations among them. This kind of syntax is primarily intended for IT specialists with strong modeling experience.

The third syntax provided by Kroki for specifying application model is the textual syntax supported within Command Window component. This component is a shell for textual commands that perform actions on EUIS DSL model. This syntax formalism is reserved for the most skilled users who are familiar with using command shells in getting more efficient.

More details on the features of the mockup editor, lightweight UML editor and command window, a reader can find in the next subsection.

Besides the syntax used for designing an application model, the final result of the model design phase is an instance of the EUIS DSL model, which can be further used for the programmatic generation of the business information system which graphical interface and data model are in conformance with the designed EUIS DSL model. The mockup editor, UML editor and command window are just different views on the same underlying EUIS DSL model. This means that changes made through any of the mentioned components are reflected to the underlying model. Consequently, all the changes introduced through one of the components are immediately reflected to other two components. Such automatic synchronization ensures that different users can cooperate on the same model by analyzing and editing model in the component they prefer.

The intended purpose of a designed EUIS DSL model is to be used by the Kroki's *prototype execution* component as an input model for the programmatic generation of a business information system. Before we present the details on the prototype generation and execution, it should be mentioned that the created EUIS DSL model can be exported to a common format. Namely, we support exporting the model to the XMI format which represents the Eclipse UML2 (Filipović et al., 2017), it used by Eclipse Modeling Framework (EMF) via Extensible Markup Language (XML) to exchange metadata information (Boldt and Steinberg, 2006).

Exporting in such commonly used format provides the usage of designed EUIS DSL model in a different context from generating a prototype of business information system in Kroki. Similarly, Kroki supports importing Eclipse UML2 model. This feature transforms input XMI model into EUIS DSL model, which servers as a starting point for developing a business information system using Kroki.

The *prototype execution* component relies on a previously designed EUIS DSL model. More precisely, the data contained in this model are input parameters for the set of different generators. The generators programmatically create various files that represent different aspects of the business information system, i.e. *DB scheme* represents the system database model, *EJB* describe in-memory data model, *XML UI spec* contains information on the graphical interface of the system, etc.

The files created by the generators are stored within the *Application repository* component. Those are configuration files which are used by application engines in the process of the generation of a business information system. With regard to the supported engines, the first one provides generation of desktop application, while the second one creates a web application. The engines parse configuration data stored within the *Application repository* and use *Code exporter* component to generate the program code of a business information system. Depending on the engine, the code for the desktop or web application is generated. Different programming languages can be supported by the *Code exporter* component. Currently, there are exporters for Java and Python programming languages. Besides the application program code for desktop or web application, the exporter generates SQL code for creating relational database schema of the application. The architecture supports various external database management systems (DBMS) which are connected to the generated application using corresponding drivers. In addition, if the external DBMS is not specified, the internal embedded DBMS is supported. After the application code is generated, Kroki executes it, resulting in the running prototype of a business information system.

As mentioned, Kroki enables configuration of the user access rights in the generated application. This is done by the *Administration subsystem* component, which is implemented using role-based access control (RBAC) methods, this subsystem enables registering application users as well as

mapping them to the specific roles in the application (Kaplar et al, 2015). Further, for each role a specific set of allowed actions can be defined. The whole process is done just by configuring application with no need to implement the access control manually. The configuration is done using a graphical interface which speeds up configuration and make it available to users with less technical knowledge.

2.2. Features

This subsection presents the main features of the Kroki tool. Figure 2.2 presents the main window of the Kroki tool.

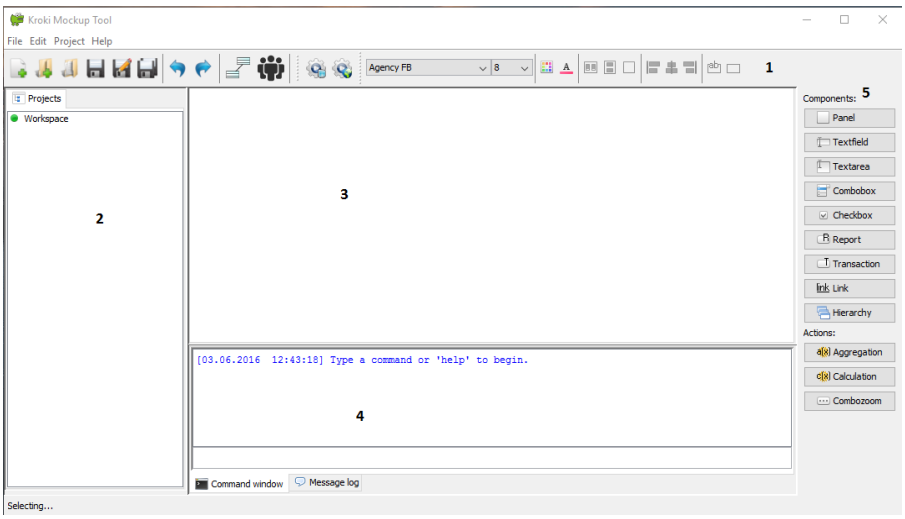


Figure 2.2. Kroki – main window

The window contains five sections, labeled with the corresponding numbers in the figure.

Section 1 is a toolbar with the buttons performing different actions:

- Create, open and save the project
- Undo and redo
- Display current model in the UML Kroki editor

- Display administration panel for controlling access rights
- Generate and launch web application based on the current model
- Generate and launch standalone application based on the current model

Section 2 represents the project explorer. It is a hierarchical outline of projects in the workspace. Each project contains multiple files which are also shown in this section. Actions from the toolbar or context menu refer to the selected element in the project explorer.

Section 3 shows the mockup editor for editing information and layout of the particular model element.

Section 4 contains a panel that consists of two tabs. The first tab presents the Command window that allows communication with the Kroki tool using textual commands. The second one is Message log tab which displays log messages about the progress of actions in the tool.

Section 5 is a tool palette for the mockup editor. It displays the elements that can be added to the panel presented in the mockup editor.

As explained, Kroki represents the model of a business information system using EUIS DSL. Instead of representing entities of an information system, which is a common approach in other meta-models, EUIS DSL represents the graphical components of an information system, such as fields, panels, etc. Since each graphical component specifies data contained within it, the EUIS DSL model simultaneously represent both data model and the model of application graphical interface.

2.2.1. Mockup editor

Mockup editor is a primary option for representing EUIS DSL model elements. It enables user to define graphical components of the application, data presented on them, as well as visual layout of the data presentation.

The basic graphical component that Kroki recognized is the standard panel. A Kroki project contains a set of standard panels, whereby each panel can be edited using the mockup editor. The mockup editor is shown in Figure 2.3.

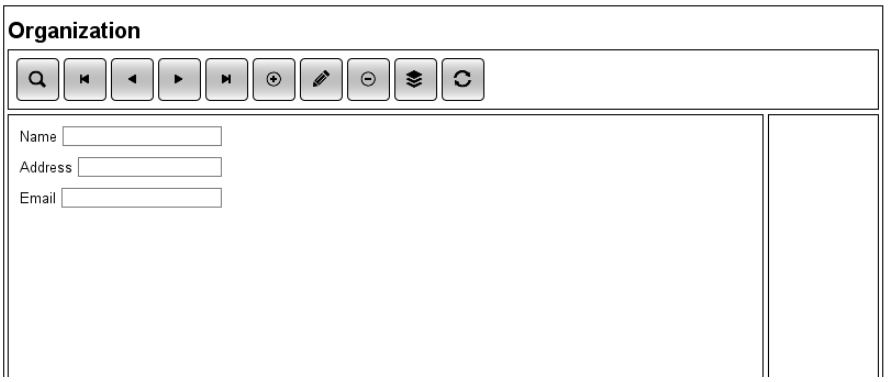


Figure 2.3. The mockup editor

The standard panel in the Kroki tool looks similar to the corresponding window or web page in the generated application (it will be displayed as a window in the stand-alone application, while in the web application its content is presented in a form of a web page). The tool bar on the top of the panel does not have a function in the design-time. It is rather present to give user a more accurate impression on the final panel look. The central part of the panel is reserved for input fields. The fields are added to the panel using the palette shown on the right-hand side of Figure 2.2. Each field represents a single data item that will be stored for the corresponding entity. In terms of the information system data model, a standard panel corresponds to entity in the system, while a field in the panel refers to an entity attribute. Hence, the mockup editor actually presents a GUI component that will be used for editing data of a particular entity in the generated application.

Different field types are supported, namely text box, text area, combo box and check box. Text box field is the most commonly used field and it provides entering a single line of text. Text area is intended for a longer text that spread across multiple lines. These two fields allow entering free text. However, sometimes is necessary to constrain input data on a limited set of possible values. For this purpose, combo box field is used. Finally, if the field stores the logical value (true or false), the check box field is used.

The mockup editor consists of mockup drawing area, UI component palette, and property editor panels used for setting the properties of the panel fields. The property editor is divided into two tabs: the first contains basic settings, which can be adjusted by non-programmers; the second contains advanced

settings, intended for advanced users and developers. Unless the advanced properties are set, defaults are used, so that prototype execution is always possible.

For each field, numerous properties can be set. Field properties are displayed in a separate panel shown in Figure 2.4.

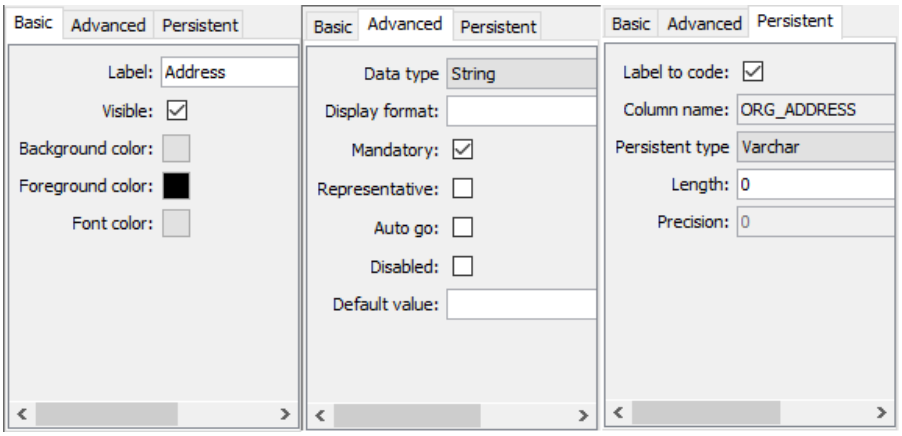


Figure 2.4. Field attributes

The properties can be categorized in three groups (each showed on a separate tab in the panel):

1.Basic options

- Label – the caption displayed next to the field
- Visible – is the field visible in the generated application
- Background/Foreground/Font color – the colors used when the field is displayed

2.Advanced options

- Data type – data type that will be used for the storage. Possible values are String, Integer, Long, Big Decimal, Date, and Email
- Display format –additional customization of the field value displaying (e.g. date format)

- Mandatory – is it allowed for the field to be blank

- Disabled – is the field read-only. If the field is disabled, it will be displayed, but the user could not enter any data

3.Persistent options (for specifying details on the field data storage)

- Label to code – how the field label corresponds to the name used in program code for the field. If checked, the code name will equal the label

- Column name – name of the database column that stores data from the field

- Persistent type – database data type for the storage of the field value. Possible types are char, varchar, text, integer, number, and decimal

- Limit - database column size for the field

- Precision – precision of the database column (used for the fields of type decimal)

2.2.2. UML editor

As mentioned, the mockup editor is just one of the possible options for specifying underlying application model. The second option is using the Kroki lightweight UML editor. This editor provides modeling application entities and their attributes using graphical UML-like notation. More precisely, it models panels and fields that should be presented in a business information system. In addition, it allows establishing relations among panels. Figure 2.5 shows the Kroki UML editor.

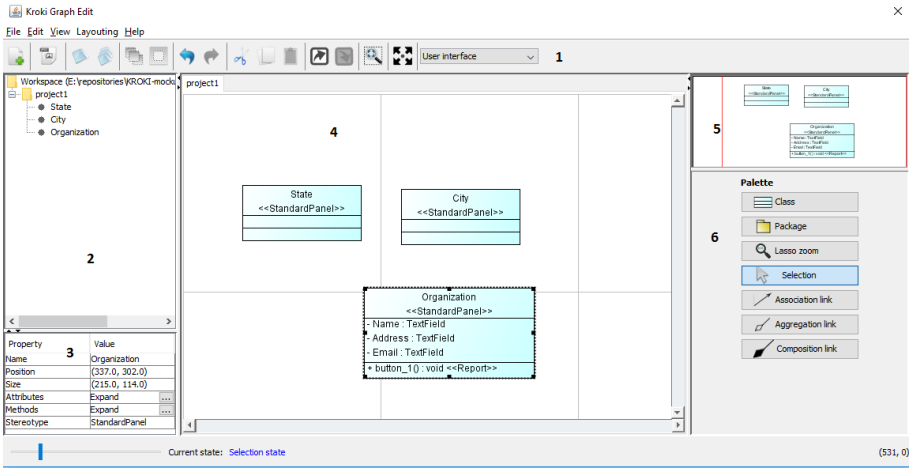


Figure 2.5. Kroki UML editor

The UML editor consists of six sections.

Section 1 is a toolbar with the buttons used mainly for navigation and undo/redo mechanisms.

Section 2 is the already described package explorer for displaying workspace content.

Section 3 is the property editor for the field selected in the diagram.

Section 4 is a main section that represents the canvas of the UML-like diagram of application elements. It displays the elements, their fields as well as links among elements.

Section 5 is a mini map of the canvas shown in section 4. Its purpose is presenting a big picture of the whole canvas, as well as enabling quick navigation to different canvas parts.

Section 6 is the tool palette that enables different actions on the diagram. The actions include adding elements to the diagram, selecting the elements, and creating relationships among them.

Elements on the diagram can be connected using the association links. The association between diagram elements specifies that data presented in these elements in the generated information systems are related. For example, if an

information system stores data on states and their cities, these two groups of data will be presented on separate standard panels. Still, these data are related, given that a city belongs to a particular state. In the generated application, it will be possible to display the state which the selected city belongs to. This is the Kroki feature named *zoom form*. In addition, it is possible to follow the opposite direction of the relationship between states and cities. For the specific state, a user can display all the cities located in it. Kroki names this feature *next form*. An association between two panels in the Kroki UML editor is shown in Figure 2.6.

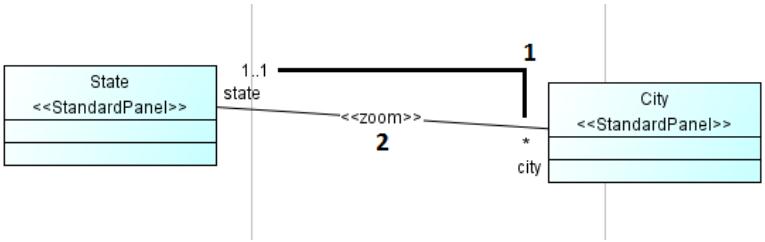


Figure 2.6. Association between standard panels in the Kroki UML editor

Two important properties of an association link are labeled in the figure. The label 1 marks the cardinality of the relationship. We can notice that there can be many cities located in some state (it is 1:N relationship). The second label marks the stereotype used for the relationship between these two standard panels. In this example the stereotype is zoom, meaning that a user can display the details on the state of the particular city.

As explained, the changes introduced into the model in the particular Kroki editor will be reflected immediately to other editors. Hence, the zoom form association between states and cities added within the UML editor will be immediately presented in the Kroki mockup editor. This type of association will be shown in the mockup editor as a link to the panel City within the panel State. This is illustrated in Figure 2.7.

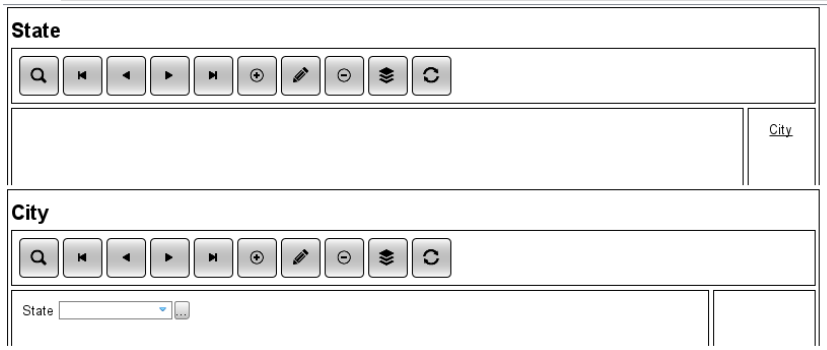


Figure 2.7. Association between standard panels in the Kroki mockup editor

Previously presented association between standard panels will present data on separate panels with a link for navigation between them. In addition, Kroki supports presenting related data within a single panel. The panel will present the parent entity as well as the child entities related to this parent entity. Instead of using the Standard panel stereotype for a diagram element, this method requires setting ParentChild as the element stereotype. The panels that should form parent-child hierarchy should be connected with the ParentChild panel using association links. Figure 2.8 presents this kind of relationship in the UML editor.

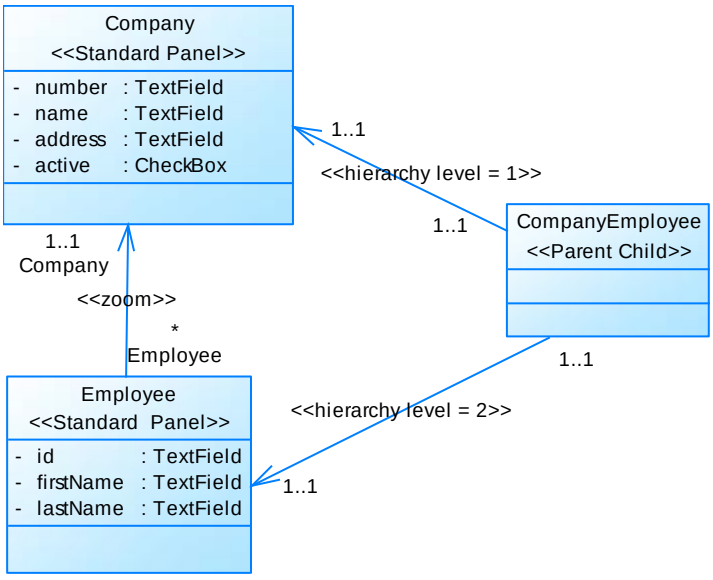


Figure 2.8. Parent-child association in the Kroki UML editor

The *Parent Child* panel presented in Figure 2.8 will be presented in the mockup editor as a composite panel that contains both standard panels that forms the *parent-child* association. Figure 2.9 presents this *Parent Child* panel in the mockup editor.

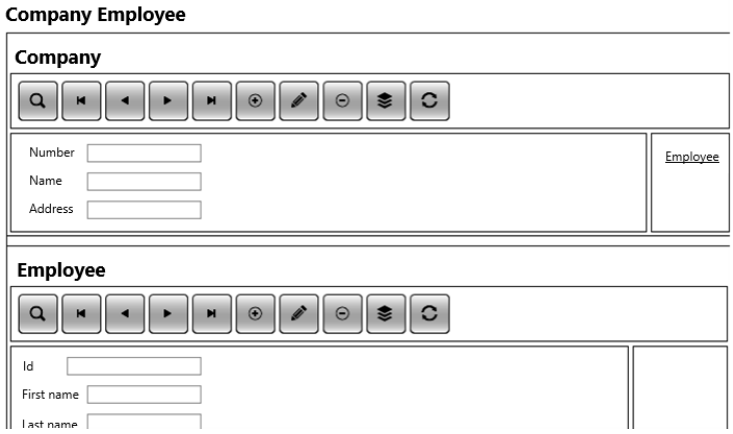


Figure 2.9. *Parent Child* panel in the Kroki mockup editor

2.2.3. Command window

Command window is the third way how the underlying EUIS DSL application model can be designed in Kroki. As shown in Figure 2.2, command window is presented on a separate tab below the Kroki mockup editor. Command window allows editing EUIS DSL application model using textual commands. Three supported commands are:

1. make project – creates a new project in Kroki
2. make package – creates a new package in a project
3. make std-panel – creates a new standard panel

Each command receives input parameters that additionally describe the command. An example of a command for creating new standard panel is presented.

```
make std-panel "Workers" in "Resources/Human resources"
{textfield-First name, textfield-Last name, textarea-
Address, checkbox-Married}
```

The presented example creates a new standard panel named *Workers* within the package *Resources/Human* resources. The panel will contain 4 fields, i.e. *First name*, *Last name*, *Address*, and *Married*. First two fields are text boxes. The third one is the text area, while the last one is the check box.

2.2.4. Administration subsystem

This subsection presents the support for authorization and authentication in the Kroki tool. As mentioned, it is based on RBAC. According to RBAC model, each user can take specific roles in the application. For the specified roles, different permissions for executing actions on application resources can be set.

An application generated from a Kroki application model has a built-in support for setting access rights (Kaplar et al., 2015). The access rights are set using the graphical interface within the administration subsystem. As mentioned, the permissions are set on the application resources. Within the generation process, the list of all application resources (forms, reports, etc.) is generated. This list can be seen on a separate window in the application subsystem. This window is shown in Figure 2.10.

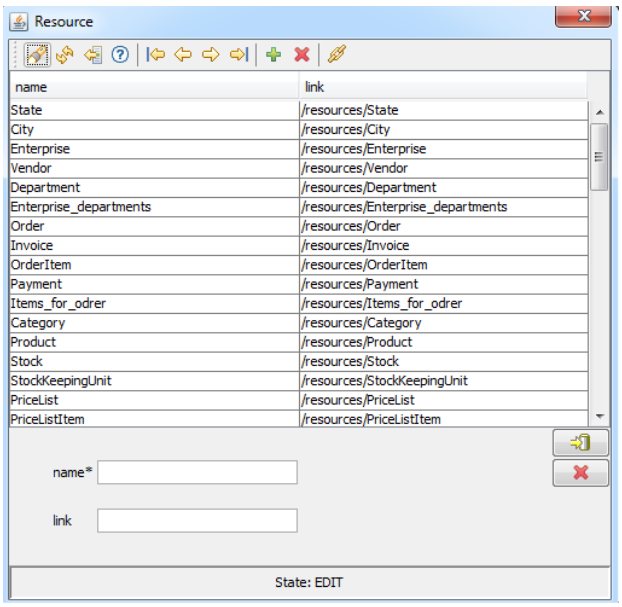


Figure 2.10. The list of the resources in the Kroki administration subsystem

For the listed resources, specific permissions can be set. The permissions are set using the form shown in Figure 2.11.

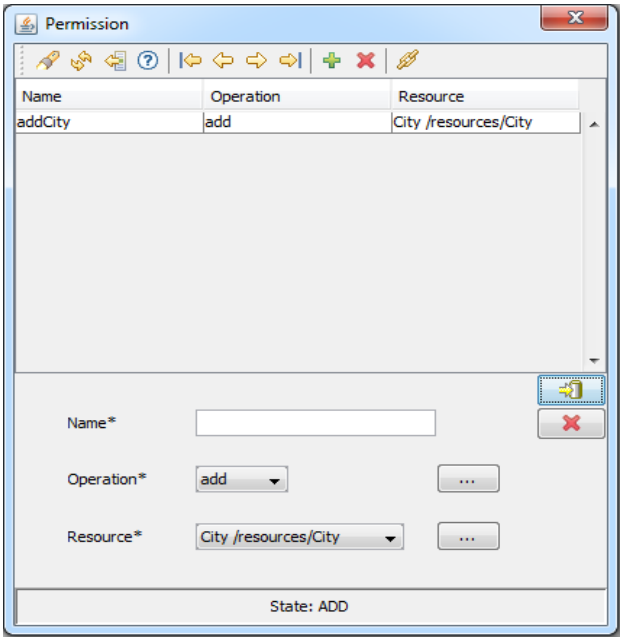


Figure 2.11. Setting permissions in the Kroki administration subsystem

In conformance to RBAC model, permission specifies which operation can be executed on a specific resource. In the presented example, we define permission for adding new cities in the application.

Specific permission is assigned to a set of user roles. User roles allows grouping of users. Figure 2.12 presents the form for administrating user roles in the administration subsystem.

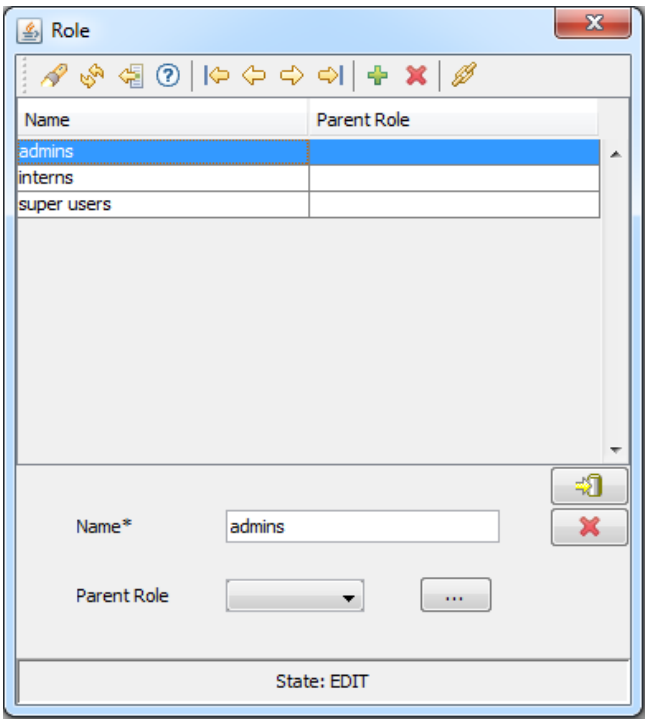


Figure 2.12. Administering user roles in the Kroki administration subsystem

The application menus can be customized in conformance with user's role. By this, each user will be provided with different set of actions, depending on his/her role. The menu customization is also done using GUI forms within the administration subsystem. These forms are shown in Figure 2.13.

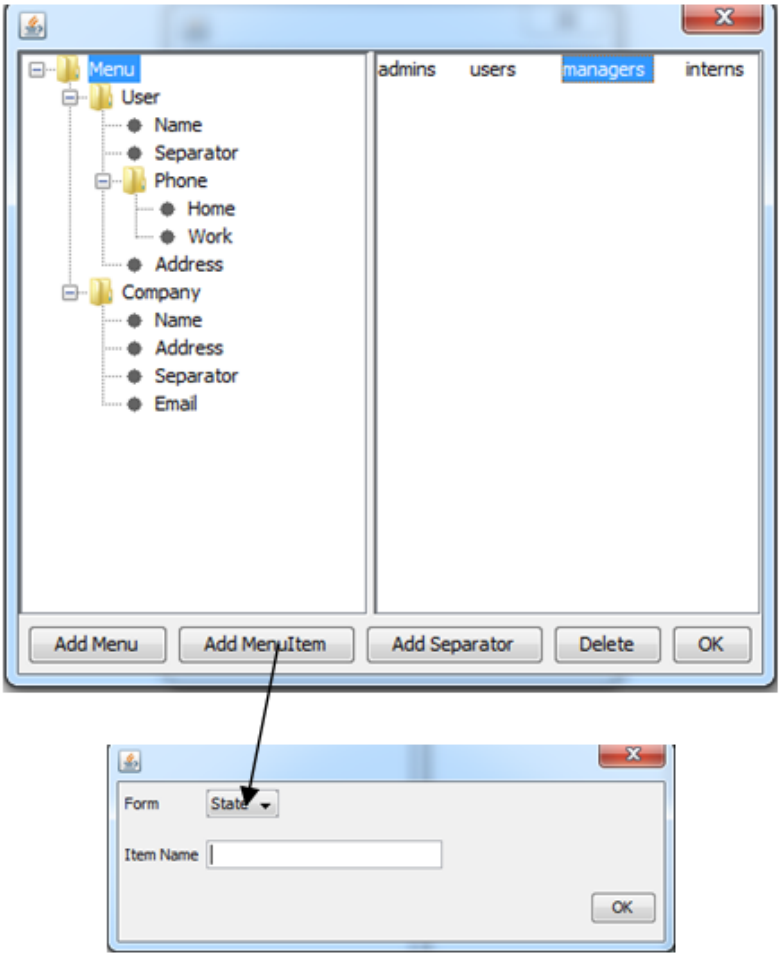


Figure 2.13. Customization of menus depending on the user roles

2.3. Generated application

As mentioned, based on the application model created within Kroki editors, Kroki can generate a software application for managing entities represented in the model. It supports generating both stand-alone and web applications. Since we are focused on the management of educational resources in online digital repositories, we are going to present the features of a web application generated using the Kroki tool.

The generated application provides the management of all entities contained in the model. Although the entities can contain various information and bring

different semantic, the generated application manages all types of entities in a uniform way. The basic component of user interface for managing any type of entity is the standard panel. Standard panel for managing entity Organization is shown in Figure 2.14.

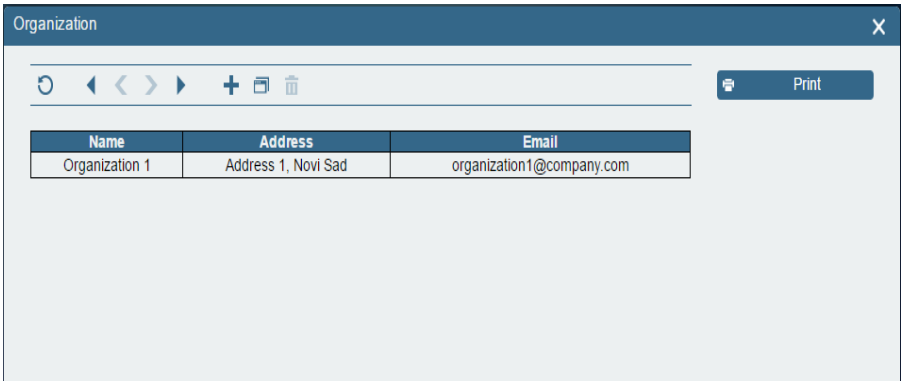


Figure 2.14. Standard panel of a generated application (*View* mode)

The standard panel has been designed to display data and all available operations so the user can choose a data item and invoke an operation on it without memorizing commands. Standard operations common to all entities are represented by buttons/icons at the top of the form. Operations common to all entities include search (query by form), display, addition, update, and removal. The standard panel has two different views. *View* mode presents a list of all entities in the form of a table (Figure 2.14). A new entity can be added to the list by switching the panel to *Add* mode. *Add* mode displays input fields for a single entity previously selected in the table presented in a view panel mode. Figure 2.15 presents the standard panel in the *Add* mode. Besides for adding new entities, add mode is used for editing data of existing entities.

Figure 2.15. Standard panel of a generated application (*Add* mode)

Besides standard operations on the entities, the standard panel also provides a set of specific operations represented by links/buttons on the right-hand side of the panel.

Specific operations include complex data processing procedures associated with the given entity (transactions), invocation of related (*next*) screen forms, and invocation of reports (Milosavljevic et al., 2011).

Related forms are inferred from association links between entities defined within Kroki UML model. Standard panel provides invocation of *next* forms. The *next* form present entities which are related to the selected entity through a 1:N association link. In other words, those are the entities that have the relational database foreign key which references the selected entity. The opposite direction of the association link is named *zoom* form. A *zoom* form displays details on an entity that another entity points to. Figure 2.16 presents a *next* and *zoom* form in the generated application.

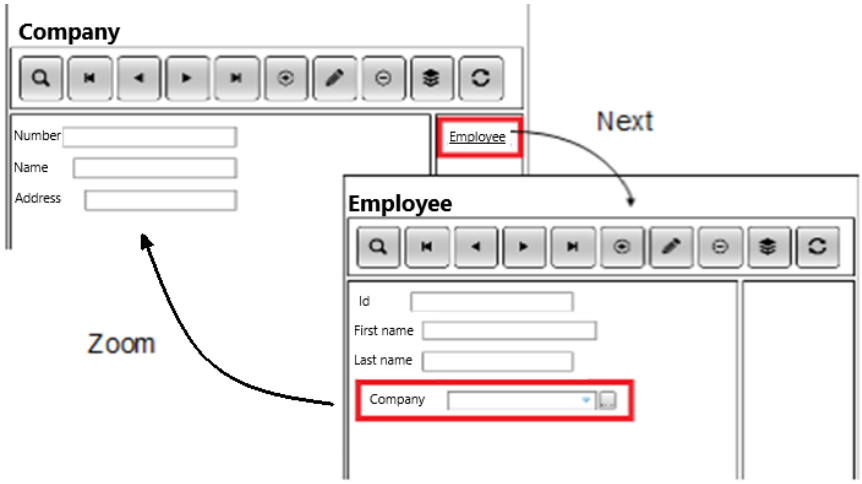


Figure 2.16. *Next* and *zoom* forms in the generated application

3. Executable platform for the management of educational resources metadata

In this research, we use Kroki as a basis for the development of the platform for management of educational resources. The platform is presented in the paper (Alhaag et al., 2018). We have extended Kroki to be used for modeling different metadata sets for describing such resources. Based on the model, one can get an executable, three-tiered application for administrating educational resources which semantics is described by different metadata sets.

Since Kroki supports generating a prototype of a software application based on different domain models, we decided to modify Kroki so that it can be used for designing educational resources metadata models. Kroki itself provides generating a prototype of an information system based on the previously created application model. Such a prototype can be used to improve communication with a client during the requirements analysis phase. We have modified Kroki to generate a software application for managing educational resources and their metadata, based on the different metadata models. Our platform provides further adding of new metadata models which extend the semantics of the already recorded educational resources. The functionalities which are added to the already generated software application, as well as previously defined semantics using initial metadata models will not be affected by adding new metadata models. In this way, we have provided an extendable application with dynamic metadata sets which, at the same time, preserves the current features and models used in the application.

3.1. The platform architecture

An overview of the proposed platform is presented in Figure 3.1.

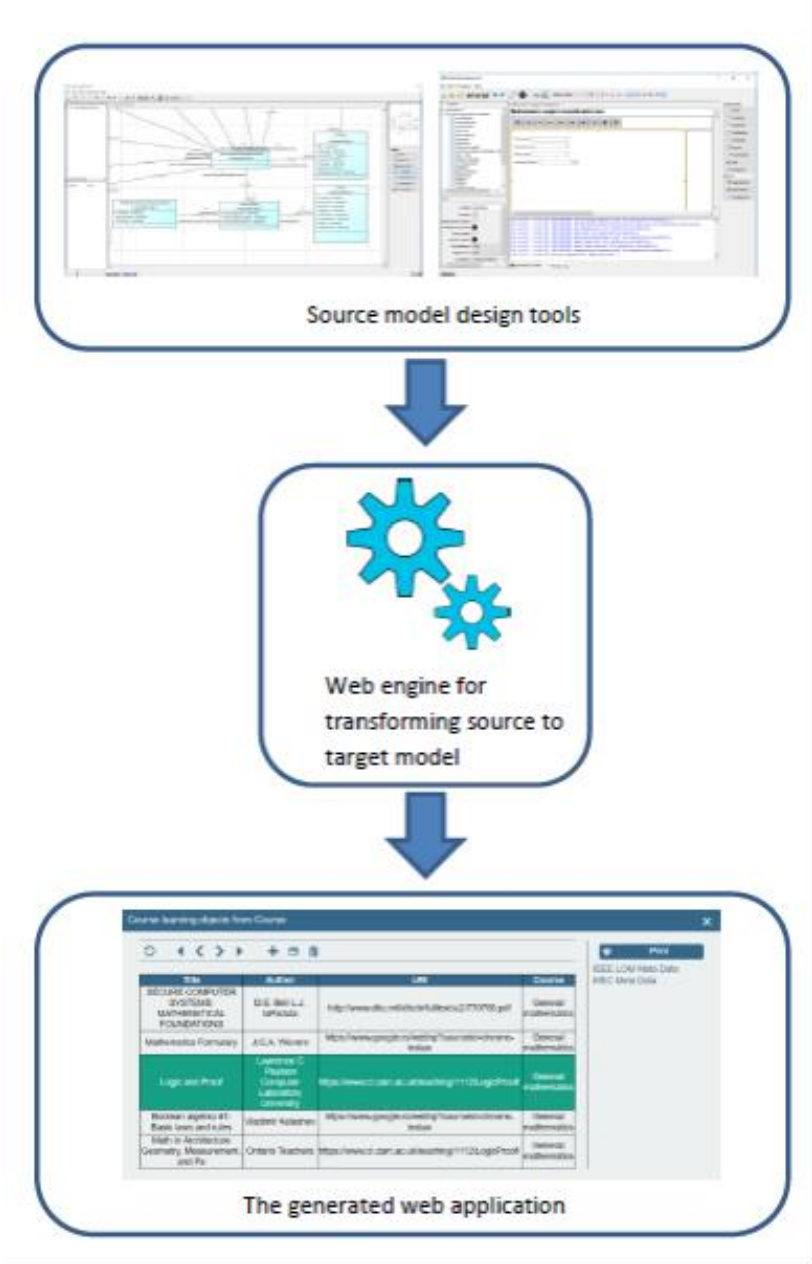


Figure 3.1. Platform overview

The first step in creating a new software application for managing educational resources using our platform is designing a model of educational resources and their metadata. It is done using the graphical tools described in the

previous section. We mentioned that our approach relies on the model-driven engineering techniques which imply generating the target model based on a source model. In this case, the source model is the model of educational resources and its metadata designed within the Kroki tool. The target model is the implementation of the final software application for the management of educational resources metadata.

Based on the designed metadata models and their visual representation, the proposed platform can generate the final software application. The platform supports both generating desktop and web application. Since we want to provide public access to our application for managing educational resources, we are focused on generating the web application.

The platform will generate a complete three-tier Java web application. The architecture of the generated application is illustrated in Figure 3.2

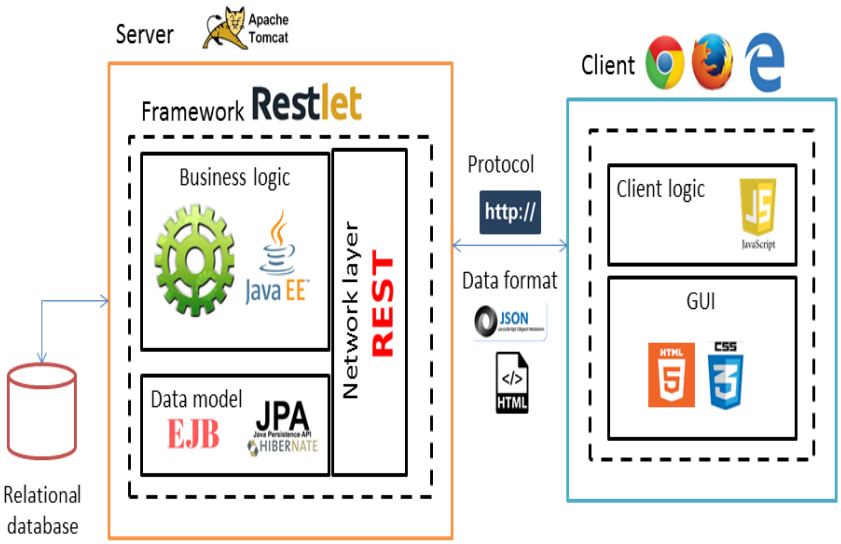


Figure 3.2. The architecture of the final application

The generated application has the client-server architecture with Apache Tomcat (The Apache Software Foundation, 1999) used as a web server. The data storage on the server side is implemented using a relational database. Underlying database management system (DBMS) is configurable enabling support for any DBMS which has appropriate JDBC driver implemented. The

server side is implemented using Java Enterprise Edition (Java EE) (Oracle) technologies. In particular, the in-memory data are represented using Enterprise Java Beans (EJB). The data are synchronized between memory and database semi automatically using the JPA specification (Oracle). As the concrete implementation of this specification, we use Hibernate library (Red Hat, 2017). The server-side code is designed in conformance with the Restlet framework (Restlet, 2017). This framework enables communication with the client side using REST web services (Fielding, 2000). Restlet is a framework for creating REST-compliant web services API in the Java web application. The Representational State Transfer (REST) software architecture models a system as a set of resources, where the predefined set of operations can be performed on each resource. The resource represents data or functionalities identified by a uniform resource identifier (URI). The resource is separated from the concrete format used to represent it, meaning that different formats can be used to represent the same resource. The most commonly used operations on REST resources are create, read, update and delete, which provide adding, retrieval, modification and removal of the entity, separately. The REST architecture is in software engineering commonly used for web services. A web service is a software system designed to support interoperable communication between heterogeneous software components by using the HTTP protocol, the service receives the request, processes it, and returns a response. Lately, the most commonly used solution for implementing web services is RESTful which is based on the REST architecture. RESTful services (Tyagi, 2006) use the HTTP protocol for network communication between software components. It is a stateless protocol which is in conformance with the stateless design of REST communication. The standard REST operations are provided using the corresponding HTTP methods (POST, GET, PUT, DELETE ...). The resources can be transferred through the HTTP protocol in different formats, mostly XML, JSON or HTML. The Restlet framework provides Java classes for implementing RESTful web services. The framework has been designed in conformance with the classic REST architecture providing the classes for each REST concept, such as resource, representation, component, etc.

With regard to the client side, the client application is the web browser. The commonly used browsers are supported. The business logic executed on the client side is implemented using the JavaScript programming language, while the visual representation of the web pages relies on HTML and CSS

technologies. The client and server communicate over the HTTP protocol by exchanging data in the JSON and HTML format.

3.2. Model transformation

At the process of getting the final software application, there are two main approaches. The first approach is to generate the complete source code of the final application. The main focus of this approach is saving developer hours by programmatically creating the code that would have to be implemented manually otherwise. The second one relies on using generic engines which generate application components on the fly. Such an approach is mainly aimed at designing generic solution that is more flexible and can be reused for different applications. The software platform proposed in this research uses the second approach.

The implementation has been done by using aspect-oriented programming. This programming paradigm helps to modularize program code by using aspects. Aspects are separate application components that can be externally applied to different parts of the application. The aspects usually implement functionalities that affect multiple parts of an application. By using aspects, we avoid to implement the same functionalities multiple times. Instead, an application code is enhanced with aspect, which externally adds specific functionality into the application code. An example on using aspects is a support for security or logging in an application. Such supporting functionalities are needed in most application components and it would be complicated if all the components should explicitly take a care on them.

The classic object-oriented approach on implementing reusable code components is placing the code within functions that are available in different application parts. The function requires to be explicitly called resulting in the application code whose main logic is interwoven with the support for the additional functionalities, such as security. Another classic object-oriented technique is to use inheritance. The common functionalities needed within different application components can be centralized within a base class. This can lead to a complicated object hierarchy since the same base class should probably be used across the whole application.

Aspect-oriented approach can provide the same level of reusability in a cleaner way. Instead of being explicitly part of the main application code, aspects are incorporated within that code in a declarative way. Such a way does not require modification of the class to which we are applying the additional feature. Rather, the supporting features are moved to aspects leaving the main application code much cleaner.

With regard to using aspects in our solution, aspect-oriented programming has been used for the transformation of the source model to the target web application. More precisely, it has been used by our generic engine to capture run-time points of interests within the application and apply the model-specific features to the generic application code (Filipovic et al., 2015). In the rest of this subsection we present more details on this topic.

The first step in the transformation process of the source model within our platform is to create a generic enterprise engine that provides general features that each enterprise information system should support. The features are later adapted to be in conformant with the previously designed source model of the application. For example, the generic engine provides the general functionality of the standard panel, which is then adapted to manage the model-specific data. The adaptation is done on the fly meaning that the engine does not generate the final application source code. The engine adapts the generic application using the data stored within the application repository. The repository stores configuration files that contain information on application source model (e.g. created panels, fields, associations, etc.). The repository files are programmatically generated within the process of creation of the target web application.

The repository structure is shown in Figure 3.3.

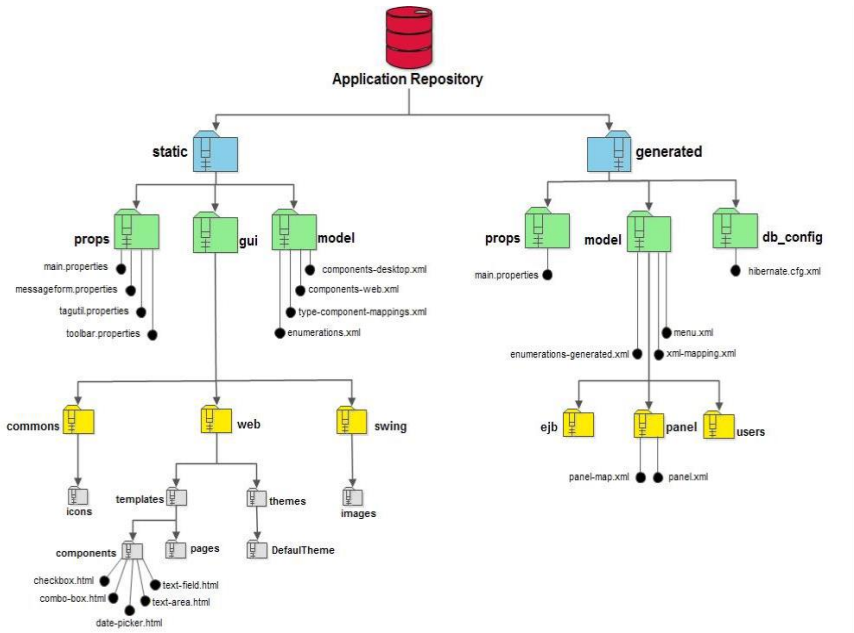


Figure 3.3. Application repository structure(Filipovic et al., 2015)

The repository structure contains two main parts. The static part of the repository stores files that are model-independent. These files are the same for each generated application and they are not subject of adaptation. This includes the implementation of the core engine functionalities as well as look-and-feel artifacts such as web pages layout, icons, images, etc.

There are three main subfolders within the *static* folder. The *props* folder contains property files with global application settings. *Model* folder stores XML files that configures generic application engine. In general, the whole configuration defined within the application repository mostly relies on XML files due to their machine readability. The model files mostly relate to the mapping of programming language types to the GUI elements of the web page. Folder *gui* holds data on the application graphical interface. There are separate subfolders for the desktop and web application. For the web application, HTML fragments that represent GUI components are stored. Also, the folder contains the HTML files that represent the application web pages. Given that this part of the engine is generic, only the templates of the web pages are stored. The final web pages are created by combining these HTML templates with data stored in the source application model. In addition, CSS files that specify the application design can be found in this folder. The data common

for the both web and desktop applications are contained within the *common* folder. Currently, those are only the application icons.

The generated part of the repository stores programmatically created files. These files are model-specific meaning that they hold information stored in the source model of the application. These are XML files that describe application model. The same information, but in the different format are already stored within the model created using the Kroki tool. Still, instead of directly reading Kroki files, we decided to create the XML files as an intermediate step. The reason is that our intention is to provide an independent application engine that does not rely on the specific tool used for the model specification.

This part of the repository has a similar structure as the one used for statically created files. Just like the folder of the same name in the static part, the *props* folder stores properties data that configures the application. In this case, the data are related to the particular source model. The *db_config* folder stores the configuration file that specifies the details on the database connection. It is the XML file used by the Hibernate library (Filipovic et al., 2015) which is in our application used for the communication with a database.

The most important programmatically generated files are those stored in the *model* folder. These files holds information on the components created within the source model.

The subdirectory *ejb* contains the specifications of the EJB beans used in our application for describing entities from the source model. The description is given as XML files. There is a separate file for each entity. The file specifies the entity attributes. Different fields are specified for an attribute, such as name, type, length, etc.

The *panel* subdirectory contains XML files that describe GUI panels in the generated web application. Each panel contains a mapping to a particular entity defined within the *ejb* folder. The panel will present data from the entities which it is mapped to.

User rights for the generated application will be stored within the *users* folder. Java enumerations used by the application code are listed in the *enumerations-generated.xml* file. The file *xml-mapping.xml* specifies the mapping between

the names of EJB classes and their corresponding XML files from the *ejb* folder. The specification of menus in the generated application is given in the *menu.xml* file. By default, there will be a menu item for each entity. The menu item directs a user to the standard panel presenting the corresponding entities.

The data stored in the application repository are used for configuring the web engine that generates the web application. The architecture of the engine is presented in Figure 3.4.

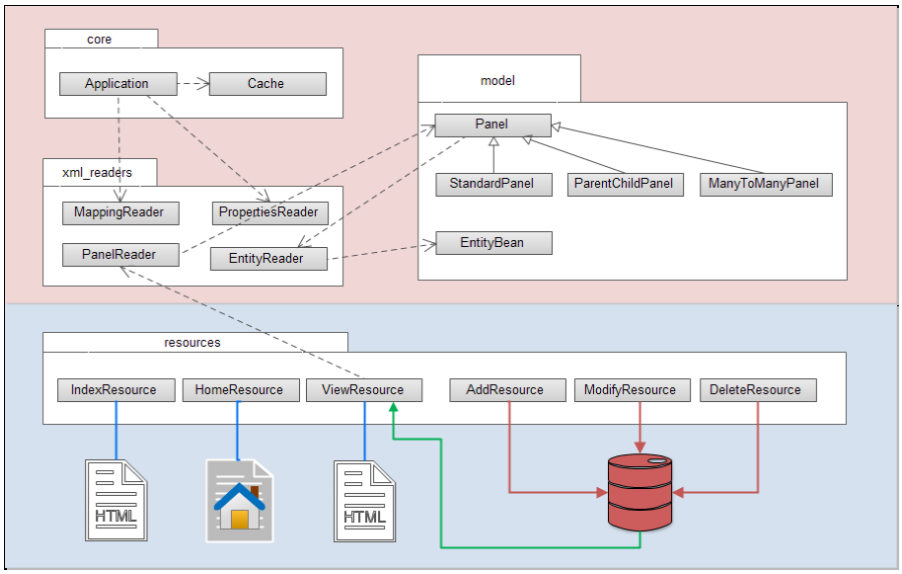


Figure 3.4. The architecture of the web engine (Filipovic et al., 2015)

The core of the engine, shown in the upper part of the figure, is used for generating both web and desktop applications. In the lower part of the figure, the components which are specific to the web engine are presented. The architecture sets core functionalities loosely coupled with a concrete GUI interface. This implies that only the view layer depends on the interface type.

The *core* package contains the main application class. This class is responsible for moderating all the engine actions. The data managed through the generation process are stored within the *Cache* class. The first step in the generation process is parsing the configuration files stored within the application repository. The parsers are implemented within the *xml_readers* package. We can notice that there are separate parsers for different repository

components. The parsers create the instances of the classes defined within the *model* package. The classes represent the elements from the application model described in the previous section. There are three types of panels for presenting application data, i.e. *standard panel*, *parent-child panel* and *many-to-many* panel. The panels are defined using the previously described mockup editor. The data shown on the panels are represented by the entities which are instances of the *EntityBean* class. It should be noted that not all model data are loaded by the parsers. Rather, only the mapping data are loaded by the parsers initially, while the actual model data is loaded on demand.

The *resources* package contains components responsible for presenting web content to the user. As explained, the web engine uses the Restlet framework. The engine generates the Resource Restlet components responsible for the RESTful communication between a web page and the back-end application. The *HomeResource* resource is responsible for handling user login and presenting the application main page. The content on the main page is provided by the *IndexResource* resource. The actions on the main page are handled by the *ViewResource* class. By communicating with the *PanelReader* class, it provides presenting application panels as the web pages. The *PanelReader* itself loads only the panel layout and standard controls. For displaying the panel data, the *PanelReader* component must call the *EntityReader* class which is associated with the corresponding *EntityBean* instance. By combining the data retrieved by the *PanelReader* and *EntityReader* components, the *ViewResource* component returns a web page that should be presented. More precisely, a Freemarker HTML template is returned. The template is later combined with the data stored in the database to get the final web page that presents the exact data that the generated application manages (e.g. the list of educational resources). On the presented data, the standard *Create*, *Update* and *Delete* operations are supported by the *AddResource*, *ModifyResource* and *DeleteResource* classes, respectively.

As mentioned, the default behavior provided by the engine can be adapted by aspects. The Restlet resource provides the method *prepareContent* that can be used for injecting additional aspect code. Using the *dataModel* resource attribute an arbitrary data can be passed to the HTML page. An aspect can add its specific data into this attribute. The whole process of generating the web application and applying custom aspects is presented in Figure 3.5.

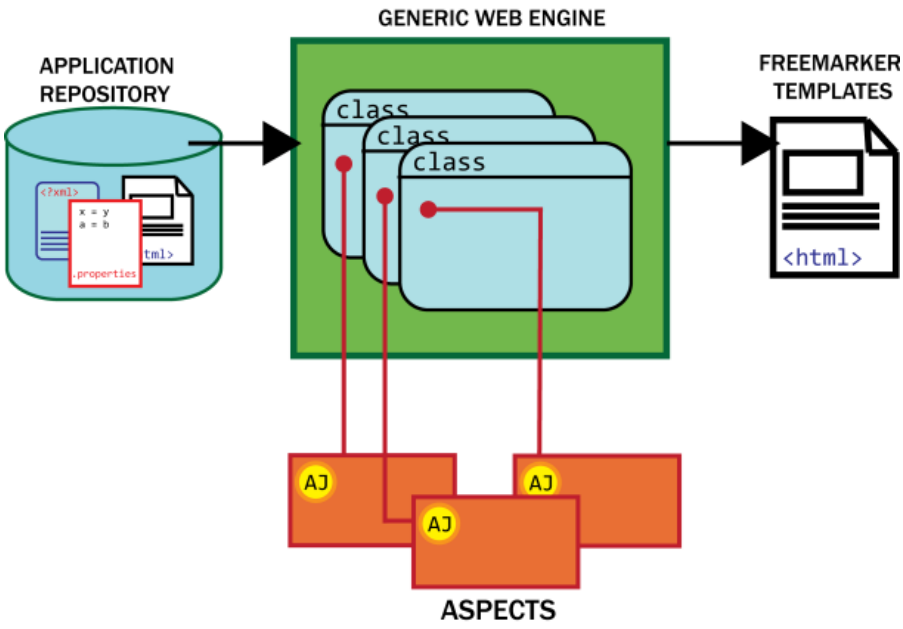


Figure 3.5. The process of generating the web application (Filipovic et al., 2015)

3.3. Model of educational resources metadata

This subsection presents created model of educational resources metadata. The model is created using the tools presented in Section 2. The next subsection describes the web application for the management of educational resources metadata which manages resources in accordance with the model presented in this section. Given that the proposed platform provides dynamic extension of supported metadata sets, the application for the management of educational resources is not limited to the modeled metadata sets.

Due to its size, we are going to present the model in multiple parts. The first part of the model is shown in Figure 3.6.

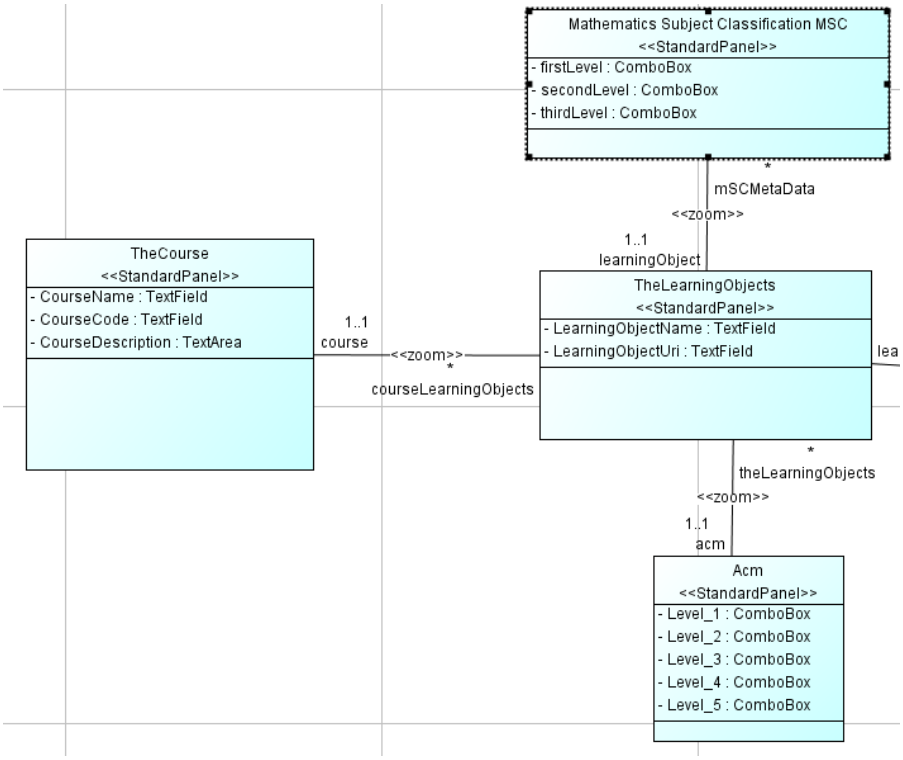


Figure 3.6. The model of educational resources metadata (part 1)

The model presents standard panels that will be displayed to users in the final application. As we can see, we organize educational resources into courses. Each educational resource can be described using different metadata sets. The presented part of the model contains ACM and MSC metadata sets. These metadata sets are modeled in accordance with their characteristics presented in Section 1. The combo boxes are chosen as graphical components for displaying metadata fields, since the value set for fields is predefined.

In addition to the presented ACM and MSC metadata models, we defined the application model for managing metadata according to the IEEE LOM model. As mentioned, the metadata fields from this scheme are organized into nine categories. Our model supports information from these categories. Figure 3.7 presents a part of the model for representing data from the categories *educational*, *general* and *rights.*

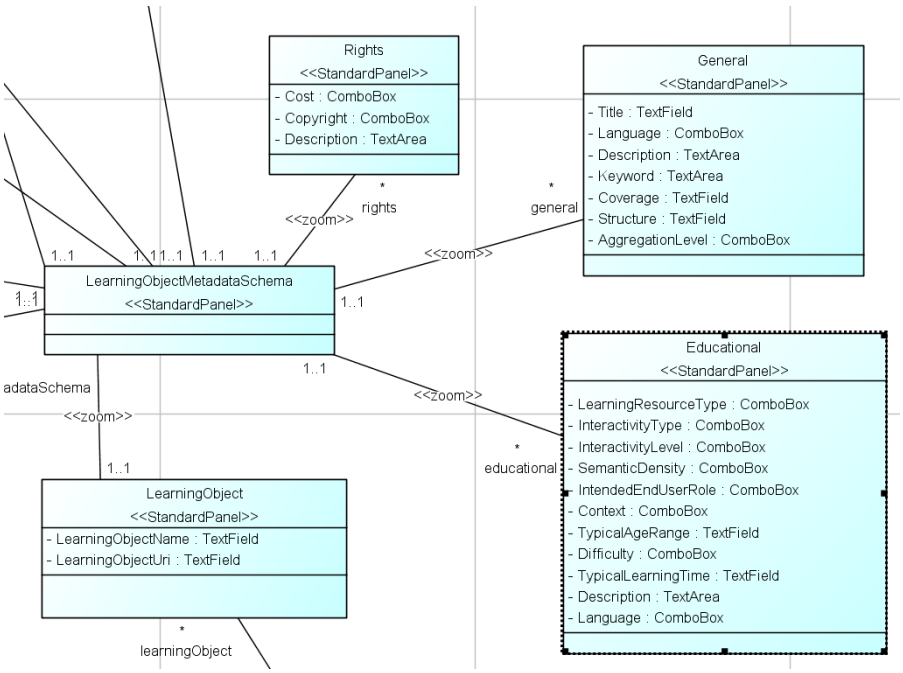


Figure 3.7. The model of educational resources metadata (part 2)

As we can see, data from each category will be presented on a separate standard panel. The fields on the panels correspond to the metadata elements from these three categories in the IEEE LOM metadata schema (see Section 1). The elements that have predefined values are presented using ComboBox panel fields. A separate panel named *LearningObjectMetadataSchema* connects all panels that present IEEE LOM data. Just like ACM and MSC panels, this panel is connected with the *LearningObject* panel, which allows user to open the panel for editing IEEE LOM data when displaying data on the specific educational resource.

The Figure 3.8 presents a part of the model that specifies IEEE LOM category *technical*.

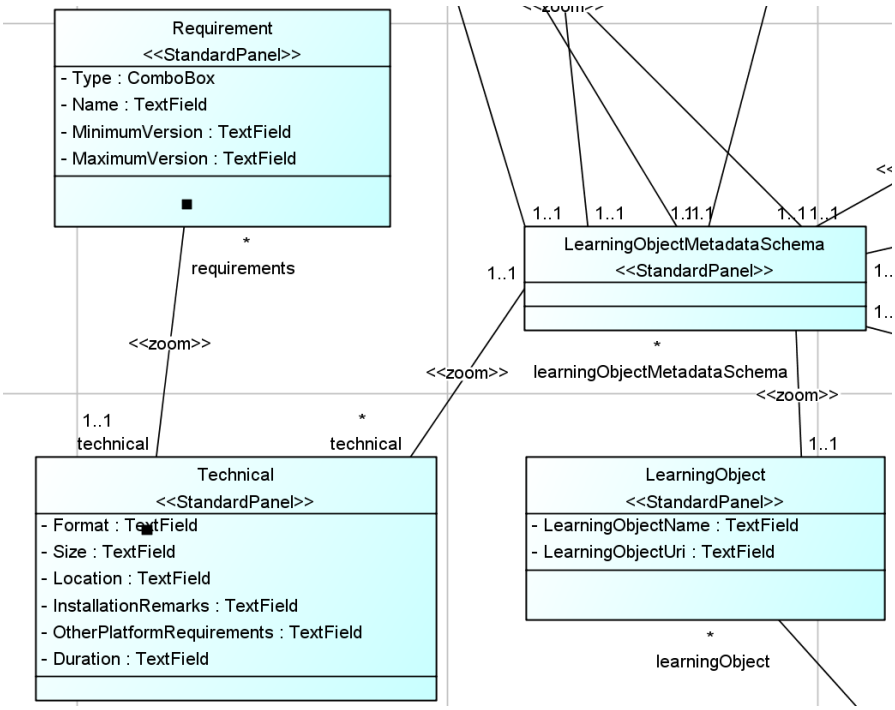


Figure 3.8. The model of educational resources metadata (part 3)

The category *technical* specifies technical characteristics of a learning object. The characteristics will be displayed in the application within the *Technical* standard panel. As specified by IEEE LOM, the category *technical* also specifies the requirements that must be fulfilled for using a learning object. The requirements are in our model defined within a separate standard panel. It should be noted that the IEEE LOM model allows specifying a composite requirement by grouping of multiple requirements using the *OrComposite* element. A composite requirement is fulfilled if at least one of the containing requirements is satisfied, meaning that the requirements form the *OR* logical relation. This is in contrast with adding multiple requirements without grouping, where the *AND* relation will be formed between them requiring that all the requirements must be fulfilled in order to use the learning object. Our model currently does not support such relations between standard panels to present the IEEE LOM *OrComposite* element. Hence, we only support defining multiple requirements using the AND logical operator.

Figure 3.9 presents *life cycle* and *annotation* categories in our model.

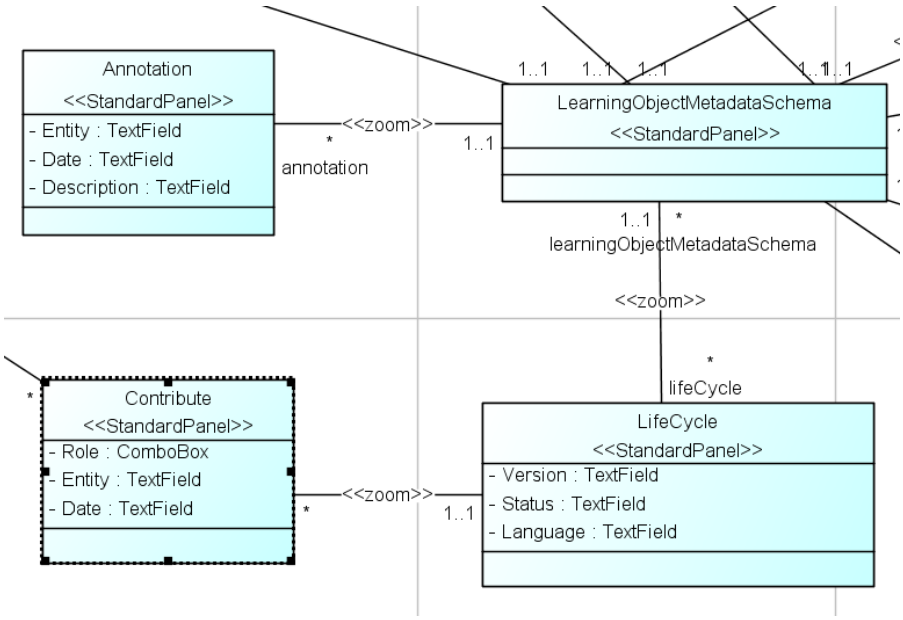


Figure 3.9. The model of educational resources metadata (part 4)

The information related to the life cycle of a learning object will be managed on the *LifeCycle* standard panel. The entities that contributed to the learning object within its life cycle are listed on the *Contribute* standard panel. The additional comments that will be recorded for a learning object will be displayed on the *Annotation* standard panel.

IEEE LOM enables storing additional information regarding the metadata record itself. This category is named *meta-metadata* and does not describe learning object, but the metadata used for describing it. The support for this category in our model is shown in Figure 3.10.

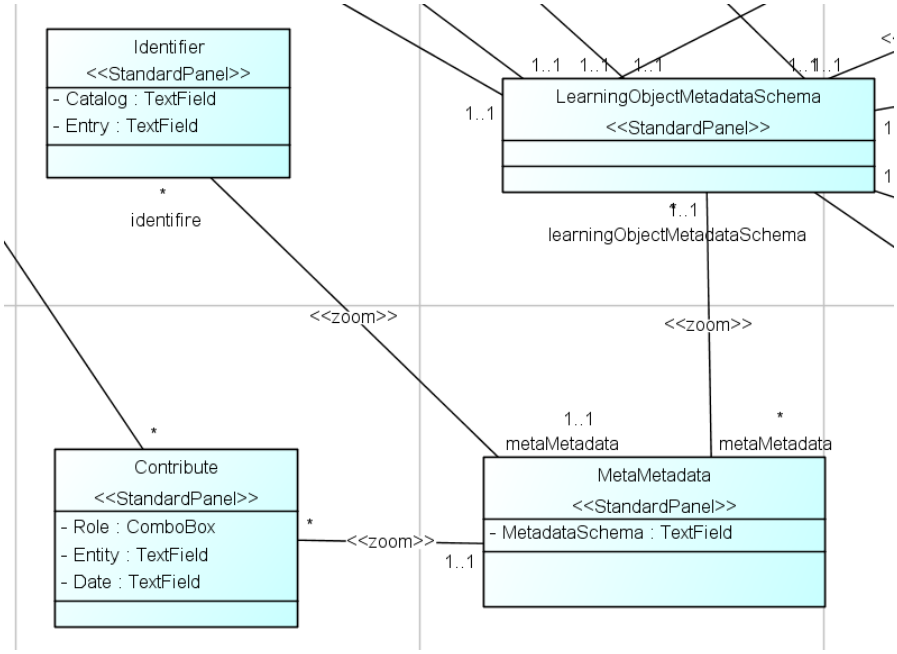


Figure 3.10. The model of educational resources metadata (part 5)

The *meta-metadata* category is presented by the corresponding standard panel. The only field presented on this panel stores information on the metadata schema used for describing learning the object. Similar to the *lifeCycle* element, a user can manage information on the entities that contributed to describing learning object with metadata. This information is displayed within the *Contribute* standard panel. The metadata record that describes the learning object is identified with information contained on the *Identifier* panel.

The relations among educational resources can be stored in accordance with the IEEE LOM metadata scheme using the part of our model presented in Figure 3.11.

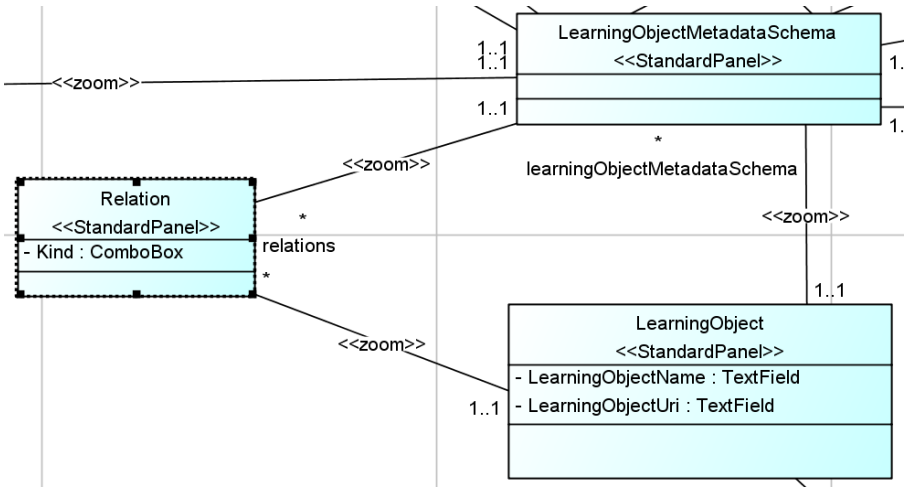


Figure 3.11. The model of educational resources metadata (part 6)

We can notice that the relation is formed with another learning object managed by our application. The *Relation* standard panel will have *zoom* link to the learning object. Using this link a user will select the related learning object. Different kinds of relations are supported by IEEE LOM. The kind of a relation is defined by the *Kind* field in the *Relation* standard panel.

With regard to the *classification* category from IEEE LOM model, its purpose is to classify learning object according to the specific taxonomy. IEEE LOM provides defining custom taxonomies. Given that our platform already provides creating custom metadata models that can be used for the classification of learning objects, we did not add support for this IEEE LOM category in the presented model. The IEEE LOM classification elements are less expressive than our application model. Hence, instead of following IEEE LOM model, it is more appropriate to create custom classifications by adding arbitrary entities in our model, just as we did with the ACM and MSC classifications (see Figure 3.6).

It should be noted that our platform still does not have a full support for displaying 1:1 relation between two entities. One way to present such relation in the application is to display data from both entities within the same standard panel. Still, there is a lack of support for presenting data on two separate panels. Dividing data into separated panels can be achieved by establishing the *zoom* association between two standard panels, but such association represents 1:N relation between entities. This means that the user will be

allowed to add more than one related entity, although the relation is defined as 1:1. Also, the standard panel will display a list of entities, but the list will always contain a single row.

A better support for 1:1 relations would improve the management of IEEE LOM metadata elements in the application. The IEEE LOM metadata model describes each learning object by 9 main categories of characteristics, where some of the categories are designed to appear only once in the object's description. Namely, those are categories *general, lifecycle, technical* and *rights*. These categories will be handled in the application in the same way like other categories that can be presented multiple times for the learning object. It means that the user will e.g. get a list of the categories *general*, although there should be only one such category for a learning object.

As described, besides the UML editor, the application model can be designed using the mockup editor. The model of the Mathematics Subject Classification displayed in the mockup editor is shown in Figure 3.12.

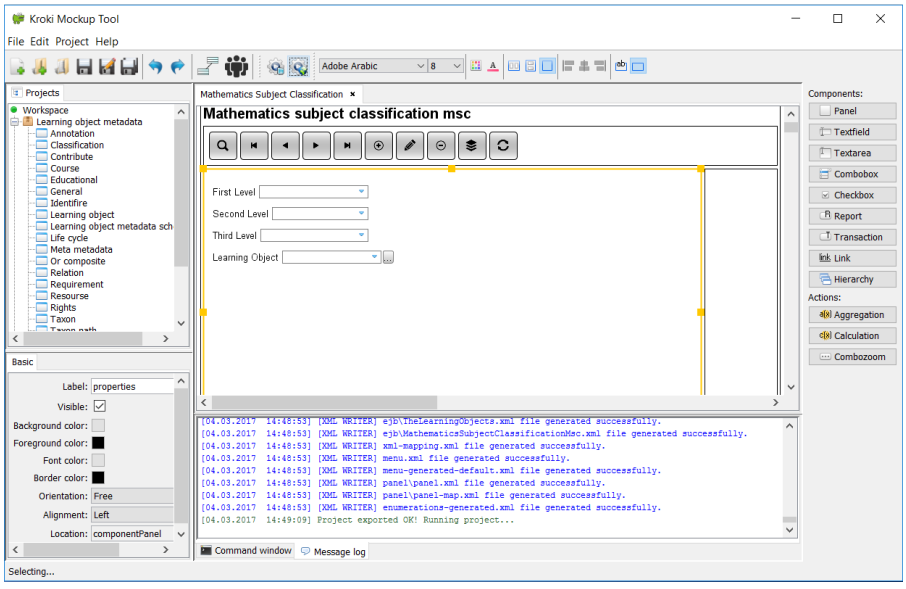


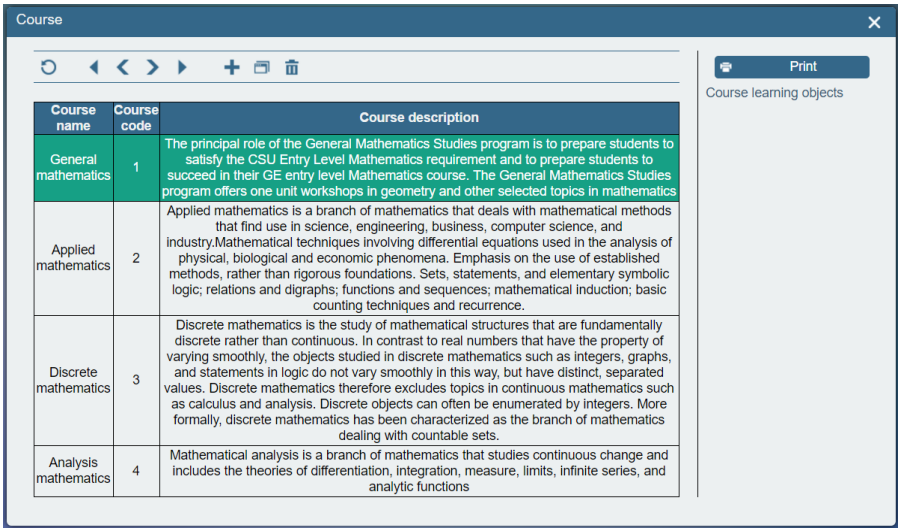
Figure 3.12. MSC model in the mockup editor

We can notice that the panel displays fields defined for the MSC entity in the UML editor (see Figure 3.6). In addition, there is the *Learning object* field. It is a *zoom* field that represents *zoom* association between MSC entity and

learning object. Since that MSC fields describe metadata related to the specific learning object, the object can be displayed using this *zoom* field.

3.4. Web application for the management of educational resources metadata

Using the application model presented in the previous subsection, the final web application for the management of educational resources metadata is generated. As explained, we organize educational resources into courses. Figure 3.13 presents the standard panel for displaying courses.



Course name	Course code	Course description
General mathematics	1	The principal role of the General Mathematics Studies program is to prepare students to satisfy the CSU Entry Level Mathematics requirement and to prepare students to succeed in their GE entry level Mathematics course. The General Mathematics Studies program offers one unit workshops in geometry and other selected topics in mathematics
Applied mathematics	2	Applied mathematics is a branch of mathematics that deals with mathematical methods that find use in science, engineering, business, computer science, and industry.Mathematical techniques involving differential equations used in the analysis of physical, biological and economic phenomena. Emphasis on the use of established methods, rather than rigorous foundations. Sets, statements, and elementary symbolic logic; relations and digraphs; functions and sequences; mathematical induction; basic counting techniques and recurrence.
Discrete mathematics	3	Discrete mathematics is the study of mathematical structures that are fundamentally discrete rather than continuous. In contrast to real numbers that have the property of varying smoothly, the objects studied in discrete mathematics such as integers, graphs, and statements in logic do not vary smoothly in this way, but have distinct, separated values. Discrete mathematics therefore excludes topics in continuous mathematics such as calculus and analysis. Discrete objects can often be enumerated by integers. More formally, discrete mathematics has been characterized as the branch of mathematics dealing with countable sets.
Analysis mathematics	4	Mathematical analysis is a branch of mathematics that studies continuous change and includes the theories of differentiation, integration, measure, limits, infinite series, and analytic functions

Figure 3.13. List of courses in the generated application

For each course, its educational resources can be displayed using the link on the right side of the panel.

Figure 3.14 shows the view mode of the standard panel which displays a list of the recorded educational resources for the *General mathematics* course.

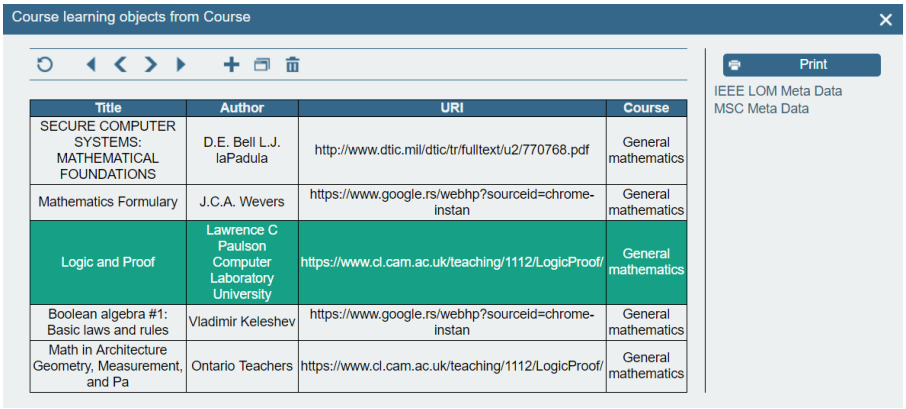


Figure 3.14. List of educational resources in the generated application

The panel presents the list of the educational resources in the general mathematics course. For each resource, its main data are presented in the table. Additionally, the resources can be described using different metadata sets. The defined metadata sets are presented in the list of the links in the upper right corner of the panel.

Given that an educational resource references its MSC metadata entity in Kroki model (see Figure 3.6), one of the links the generated application contains is a link from the educational resource to the window which displays its metadata according to MSC. The link is presented in the upper right corner in Figure 3.14. The window that displays MSC metadata and provides its editing is presented in Figure 3.15.

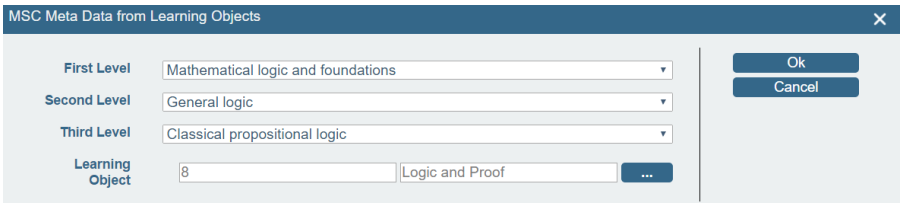


Figure 3.15. Editing MSC metadata in the generated application

1. The panel displays metadata fields in accordance with the model of MSC classification shown in Figure 3.6. We can notice the corresponding field for each level from the MSC classification. Additionally, the field *Learning object* displays the educational resource that is described by this metadata. Using the *zoom* button

(labeled with ...), a user can choose an appropriate learning object when specifying metadata. Click on the button will open a standard panel that presents the list of all learning objects.

4. Verification

The previous section presented how educational resources can be managed within our platform using various metadata sets. In order to verify our solution, we performed an experiment where independent participants evaluated the proposed software platform by performing different tasks within the platform. This chapter presents these two verification methods.

4.1. Experiment

In addition to the previously described internal case study, we have performed an experiment aimed to evaluate characteristics of the proposed platform.

4.1.1. Experiment goal

The experiment goal was to validate application features related to the model creation and application generation. We have evaluated characteristics specified by ISO/IEC 25022:2016 standard (ISO/IEC JTC 1, 2012), i.e. effectiveness, efficiency, satisfaction, usability, ease of use, clarity and attractiveness. In addition we have evaluated user's general impression of the application.

4.1.2. Data Collection Instruments

Data collection instruments are chosen in conformance with the ISO/IEC 25022:2016 standard. The characteristics effectiveness and efficiency are measured based on the results of a task given to the experiment participants and the time taken for the task completion. Other 5 characteristics are evaluated using a subjective questionnaire created according to the Common Industry Format for Usability Test Reports (CIF Questionnaire) (National Institute of Standards and Technology, 1999). This questionnaire contains 5 questions with 7-point Likert scale answers, where each question evaluates one of the characteristics.

With regard to general impressions, they are measured using a standard PSSUQ questionnaire (Lewis, 1995). The questionnaire contains 16 questions with 7-point Likert scale answers. The questions relate to user satisfaction, system usability, quality of information and quality of user interface.

In addition, the participants could leave a free comment about their impressions of the application.

4.1.3. Participants

We have mentioned that the proposed platform should allow users to modify model of educational resources metadata with no need to program support for this new model manually. Still, users must design the new model, which includes defining entities and forming relations between them. This implies that users must have some skills necessary for modeling, like knowing abstraction, association and some graphical modeling notation. For this reason, we set specific preconditions for participants in the experiment. The participants were required to be digitally literate with some experience in using information systems and developing data model in such systems. In addition, they had to know the computer science domain since the experiment task referred to the classification of educational resources in that domain. For all these reasons, we chose 16 students from the third year of professional studies of Software and information technologies from Faculty of Technical Sciences at University of Novi Sad to be the participants in our experiment.

4.1.4. Experiment procedure

The experiment is conducted in a computer laboratory at Faculty of Technical Sciences. Each participant got a separate computer with our Kroki-based executable platform preinstalled. These students have already used this laboratory during their studies, so they were familiar with the software environment. On this way we tried to minimize the effects of the software and hardware environment on the experiment results.

Within the Kroki platform, the participants got already created model of the application for managing educational resources in Kroki. The model contained the entity Course which had a collection of Learning resource entities. In the given model, the learning resource semantics was described by two metadata sets – IEEE LOM and MSC. Both metadata sets were represented in the model with the appropriate entities and their attributes. As well, we used Kroki to generate the application based on the given model. This application was installed on the computers in the laboratory and the application was initially filled with data describing 10 educational resources from the mathematics

field. In addition, we entered metadata for these resources according to IEEE LOM and MSC specification, respectively.

The main task given to the participants consisted of two subtasks. The first subtask was extending the given data model so that it supports describing educational resources using metadata from ACM classification scheme too. As their second subtask, they were asked to generate application in Kroki using the extended data model from the first subtask. In the generated application, they had to record another 5 educational resources from computer science field and to describe them using metadata set that conforms to ACM classification scheme.

In the beginning of the experiment, the participants were given the Background questionnaire to evaluate their profile and previous knowledge. On this way, we verified that chosen students satisfy the preconditions set for the participation in this experiment. The questionnaire consisted of 7 questions with answers that use 7-point Likert scale. The questions referred to their previous experience with computers and information systems, as well as the knowledge of the computer science domain, modeling and the purpose of metadata in describing educational resources.

After the Background questionnaire, we organized an oral presentation on using Kroki platform. As well, we explained the general concepts of ACM classification. Together with this oral presentation, the participants were given an electronic version of Kroki documentation and the ACM specification document. These two documents were available to the participants during the whole experiment.

Then, the participants started working on the main experiment task. The time for the task was unlimited, but we still recorded the start and time, since this parameter was used in calculating the platform efficiency.

When the participants finished the main task, they were asked to fill in the CIF and PSSUQ questionnaires in order to give feedback on using our platform.

4.1.5. Experiment results

The results of the background questionnaire are shown in Table 4.1. For each question we show per cent of the participants who gave specific answer and calculated average answer.

In accordance with the target group, we can see from the results that most of the participants have considerable experience with the topics that are preconditions for the participation in the experiment. Before the experiment, the participants were not familiar with the ACM classification. We assumed this, so we prepared ACM specification document to be available for the participants during the experiment.

The effectiveness and efficiency of the platform were evaluated through the above described main task. The task consisted of two subtasks, where each task maximum was 50 points. The points scored on the tasks are presented in Table 4.2.

In general, the participants mastered both subtasks well, while the second subtask appears to be slightly more challenging. This may indicate that usability of the generated final application, which is used to complete subtask 2, is not as high as for Kroki tool which served for completing subtask 1. This was expected since Kroki is a fully manually developed tool, while most parts of the final application are programmatically generated. Also, depending on the model complexity, the generated application will contain various panels and visual controls, so that it is inherently more complicated to use.

Question	Percent of participants							Avg answer
	Very low 1	2	3	4	5	6	Very high 7	
How would you rate your skills in using computer?	0%	0%	0%	6.3%	0%	50%	43.8%	6.31
How would you rate your skills in using information systems?	0%	0%	6.3%	12.5%	31.3%	25%	25%	5.50
How would you rate your knowledge of the computer science domain?	0%	6.3%	0%	6.3%	50%	25%	12.5%	5.25
How would you rate your skills in general modeling of problems and systems?	0%	6.3%	0%	25%	50%	12.5%	6.3%	4.81
How would you rate your experience in the e-learning domain?	0%	6.3%	0%	25%	62.5%	0%	6.3%	4.69

How would you rate your knowledge of the role of metadata in describing educational resources?	0%	6.3%	0%	37.5%	37.5%	12.5%	6.3%	4.69
How would you rate your knowledge of ACM Computing Classification System?	25%	12.5%	25%	25%	6.3%	6.3%	0%	2.94

Table 4.1 The results of the background questionnaire

Task	Average score	Standard Deviation
Subtask 1	47.85	1.41
Subtask 2	44.46	10.51
Main task total	92.31	11.61

Table 4.2. The results of the experiment main task

In accordance with the ISO/IEC 25022:2016, we measured efficiency as a ratio of the test score and the time spent on the test. In order to evaluate obtained efficiency, we defined a reference efficiency based upon estimated time for the expert who has a prior experience in using Kroki. This predicted time was 30 minutes. Setting the reference test score at 100% points gives us the reference efficiency of 3.33. Table 4.3 shows the results for the efficiency characteristic.

Reference efficiency (RE)	Average efficency (AE)	Standard deviation	Efficency ratio (AE/RE)
3.33	2.41	0.57	72.23%

Table 4.3. Task efficiency results

Although the achieved efficiency is less than the referent one, it is still at the satisfactory level. The reason for lower efficiency can be found in the fact that the participants used our application first time on the experiment. In addition, both Kroki and the generated application have some shortcomings that are discussed later and which we will try to correct in the further research. We assume that these problems also caused the obtained efficiency.

As explained, other platform characteristics are measured using the subjective CIF questionnaire. The questionnaire results are shown in Table 4.4.

In general, all the characteristics are rated with relatively high average points. The platform attractiveness was rated with the lowest average points, which is expected for the application that was developed mostly programmatically.

Question	Percent of participants							Avg answer
	Very low 1	**2**	**3**	**4**	**5**	**6**	**Very high 7**	
Satisfaction	6.3%	0%	0%	6.3%	18.8%	56.3%	12.5%	5.50
Usefulness	6.3%	0%	0%	0%	31.3%	37.5%	25%	5.63
Ease of use	0%	6.3%	0%	6.3%	31.3%	31.3%	25%	5.56
Clarity	6.3%	0%	0%	6.3%	6.3%	56.3%	25%	5.75
Attractiveness	0%	6.3%	12.5%	0%	25%	31.3%	25%	5.38

Table 4.4. The results of the CIF questionnaire

General impressions on the platform are evaluated using PSSUQ questionnaire. The questionnaire has 16 questions where each question can be related to some of the four general system characteristics, i.e. *overall satisfaction*, *system usefulness*, *quality of information*, and *interface quality*. The result for each characteristic is calculated as the average response for the corresponding group of questions.

The results are calculated according to the following scheme:

- all 16 questions refer to *overall satisfaction* characteristic
- the questions from 1 to 6 evaluate the characteristic *system usefulness*,
- the questions from 7 to 12 relate to the characteristic *quality of information*,
- the questions from 13 to 16 measure *interface quality* characteristic.

Table 4.5 presents the question-level results of the PSSUQ questionnaire, while Table 4.6 presents summed results with average responses for each characteristic.

It should be noticed that in the PSSUQ questionnaire the answer “Strongly agree” is marked with number 1, while number 7 indicates the answer “Strongly disagree”. This is opposite from the previously presented CIF and Background questionnaires where the “Strongly agree” answer was graded with number 7.

Similar to the results measured by the CIF questionnaire, the participants answered that they are generally satisfied with the platform. Just like with the CIF questionnaire, the programmatically produced user interface of the generated application got slightly lower grades.

Besides questionnaires, the participants were free to enter comments on using platform. 10 participants take advantage of this option. Generally, the comments expressed a positive impression on our system with few remarks that should be discussed. The remarks mostly relate to the quality of user

interface. Some users state that platform features would be more exploitable with the improved visual controls. Further, even 6 participants noted that they had problems with entering values into the combo box component. After the test we found a bug in this component. Besides the bug, the component was functional, still taking some time from the participant to make it work. This postponed the task completion causing lower efficiency and affecting general impressions on user interface.

Question	Percent of participants							Avg answer
	Very high 1	2	3	4	5	6	Very low 7	
1. Overall, I am satisfied with how easy it is to use this system	37.5%	37.5%	6.3%	12.5%	6.3%	0%	0%	2.15
2. It was simple to use this system	50%	25%	6.3%	18.8%	0%	0%	0%	1.92
3. I was able to complete the tasks and scenarios quickly using this system	37.5%	31.3%	18.8%	6.3%	6.3%	0%	0%	2.15
4. I felt comfortable using this system	37.5%	18.8%	18.8%	6.3%	6.3%	12.5%	0%	2.62
5. It was easy to learn to use this system	68.8%	12.5%	18.8%	0%	0%	0%	0%	1.46
6. I believe I could become productive quickly using this system	50%	18.8%	25%	6.3%	0%	0%	0%	1.85
7. The system gave error messages that clearly told me how to fix problems	31.3%	43.8%	0%	6.3%	0%	6.3%	6.3%	2.42

Question	Percent of participants							Avg answer
	Very high 1	2	3	4	5	6	Very low 7	
8. Whenever I made a mistake using the system, I could recover easily and quickly	31.3%	50%	0%	12.5%	0%	6.3%	0%	2.15
9. The information (such as on-line help, on-screen messages, and other documentation) provided with this system was clear	50%	31.3%	12.5%	0%	0%	0%	0%	1.58
10. It was easy to find the information I needed	50%	37.5%	6.3%	0%	6.3%	0%	0%	1.77
11. The information was effective in helping me complete the tasks and scenarios	43.8%	37.5%	12.5%	0%	0%	6.3%	0%	1.92
12. The organization of information on the system screens was clear	56.3%	37.5%	0%	0%	0%	0%	0%	1.42

Question	Percent of participants							Avg answer
	Very high 1	2	3	4	5	6	Very low 7	
13. The interface of this system was pleasant	50%	37.5%	12.5%	0%	0%	0%	0%	1.62
14. I liked using the interface of this system	37.5%	37.5%	25%	0%	0%	0%	0%	1.85
15. This system has all the functions and capabilities I expect it to have	25%	50%	25%	0%	0%	0%	0%	2.00
16. Overall, I am satisfied with this system	37.5%	50%	12.5%	0%	0%	0%	0%	1.77

Table 4.5. The results of the PSSUQ questionnaire

Characteristics	**PSSUQ Score**
Overall satisfaction	1.92
System usefulness	2.03
Information quality	1.88
Interface quality	1.81

Table 4.6. The results for the system characteristics based on PSSUQ questionnaire

5. Conclusion

This thesis is focused on the topic of the management of metadata on educational resources. The thesis analyses current techniques of representation and storage of such data.

The educational resources are commonly described using metadata. The thesis presents two types of these data. The first type refer to the metadata that are domain-neutral meaning that it represents general information that any educational resource contains, beside the domain that its content belongs to. IEEE LOM and Dublin Core metadata sets are described, as two most widely recognized metadata sets of this type. Handling an educational resource can be improved if its description, in addition to general information, contains some domain-specific information too. For example, there can be taxonomy for classifying educational content in the specific domain. Domain-specific metadata can describe the place in the taxonomy where an educational resource fits in. As the examples of such domain-specific metadata, we presented widely adopted taxonomies for the computer science and mathematics field.

The educational resources are commonly managed by digital repositories. We have identified the problem of supporting different metadata sets in such repositories. The number of possible metadata sets is to large making it impossible to statically predefine them. Instead, we propose a solution for the dynamic specification of metadata models.

We propose a solution on this problem using the techniques of the model-driven engineering. The basic idea of the research was to dynamically provide support for managing educational resources according to new metadata sets just by designing the models for these sets.

With regard to the general implication of the proposed solution, it presents a novel approach on managing educational resources. The current trend in software applications for the management of educational resources is implementing applications to support predefined metadata set that describes resources. In the modern digital world, diverse educational resources can be stored in centralized repositories which allow users to access a wide set of resources within a single application. Still, such an approach opens a problem of searching such diverse content. For that reason, different metadata sets should be developed to describe resources from different domains. It is difficult to initially embed all the necessary domain-specific metadata sets in

the software application, due to large number of possible sets. For the development of each domain-specific metadata set domain experts are needed who are difficult to assemble in a centralized way during the development of the software application. Instead, the approach presented in this paper suggests that domain experts could customize the initial metadata set for their specific domain after the application is developed. Such an approach ensures that the metadata set is adjustable to the need of a specific user. In this way, a user can model an appropriate metadata set during the usage of the application.

A practical implication of the proposed solution is that a single generic software application can be used for managing educational resources in various institutions. Each institution can additionally customize the initial metadata set to meet the specific domain. For example, when managing educational resources within some technical university, software administrators can adapt a metadata set to contain elements and vocabulary related to the domain of technical sciences.

The main result of the research is a platform which programmatically generates an application for managing metadata of educational resources. The platform allows dynamic modification of the underlying metadata model. The platform contains a special purpose tool for designing model of metadata which serves as a basis for further generation of the final application for managing educational resources. Such an approach ensures that the application is not limited to any predefined metadata set. A user can create its own model of metadata that is relevant in the particular domain. The programmatic generation of the final application will provide him/her recording educational resources which are described using the created metadata model with no need to modify application source code manually.

The solution is implemented using the concepts of the model-driven approach. In the terms of this approach, the metadata models represent a source model that is then programmatically transformed to a target model. The target model is the web application for the management of metadata of educational resources. Besides metadata models, the source model specifies the functionalities and visual appearance of the generated web application.

The proposed platform has been verified using the experiment where 16 students of software engineering evaluated the platform characteristics. They were asked to add new metadata model, generate the application based on this model and describe educational resources using the metadata fields contained in the newly created model.

Although the experiment proved the general usefulness of the platform, there is still dilemma how much a user must be skilled in modelling to use the platform. Further research should explore the possibility of training domain experts with no technical knowledge to use the platform and define new metadata model in their domain.

With regard to the platform current maturity, the experiment showed high level of satisfaction among users. Still, it is already evident that graphical interface must be improved. We are aware that the interface shortcomings are tightly related to the chosen approach. The fact that the final web application is programmatically generated implies generic and template-based graphical interface for each web page. In the current version of the application, the only way to customize the application is to manually implement specific support either by modifying application source code or applying Java aspects. Our goal in the future work on this project will be allowing users to specify more sophisticated visual characteristics of the application using a special purpose graphical editor which will reduce the need of adding source code for the interface customization.

Another important direction of the future work is migrating Kroki from the stand-alone application to cloud-based online service. The main idea is to provide online tool where users can design their own metadata models. Among many metadata models, each user could choose a specific personal view on metadata which includes only metadata that are of interest for the specific scenario. Finally, we are planning to provide the feature of storing and downloading educational resources themselves instead of recording metadata only as supported in the current version.

References

Advanced Distributed Learning (2015) SCORM [online] Available at: http://www.adlnet.org [Accessed 30 Mar. 2017].

Agostinho, S., Bennett, S., Lockyer, L. & Harper, B. (2004) Developing a learning object metadata application profile based on LOM suitable for Australian higher education context. Australasian Journal of Educational Technology, 20 (2), 191-208..

Alhaag, A. A., Savić, G., Milosavljević, G., Segedinac, M. T., Filipović, M. (2018) Executable platform for managing customizable metadata of educational resources, The Electronic Library, In press.

Alhir, S. (2003) Understanding the Model Driven Architecture (MDA). [online] Available at: http://www.methodsandtools.com/ archive /archive.php?id=5 [Accessed 1 Jun 2017].

Ariadne (2017), ARIADNE program. [online] Available at http://www.ariadne-eu.org.

Association for Computing Machinery (2018) How to Use the 1998 Computing Classification System. [online] Available at: http://www.acm.org/about-acm/class/how-to-use [Accessed 12 Feb. 2018].

Association for Computing Machinery. (2012) Computing Classification System - 2012 Revision. [online] Available at: http://dl.acm.org/ft_gateway.cfm?id=2371137&ftid=1290921&dwn=1 [accessed 4 March 2017].

Education Services Australia. (2011) ANZ-LOM Metadata Application Profile. [online] Available at: http://www.ndlrn.edu.au/verve/_resources/anz-lom_1_02_file.pdf [Accessed 1 Jun. 2017].

Barker, P. (2005) What is IEEE Learning Object Metadata / IMS Learning Resource Metadata? [online] Available at: http://publications.cetis.org.uk/wp-content/uploads/2011/02/WhatIs IEEELOM.pdf [Accessed 17 Mar. 2017].

Bizonova, Z. (2007) Model-driven LMS Platform Integration. Journal of Information, Control and Management Systems. 5(1), 3-12.

Bizonova, Z. & Ranc, D. (2007) Courseware Material Reuse via Model-driven LMS Platform Integration. Journal of E-learning and distance learning, University of Cyprus. 150-158.

Boyle, T. (2003) Design Principles for Authoring Dynamic, Reusable Learning. Australasian Journal of Educational Technology, Vol 19 (1). 46 – 58. doi: https://doi.org/10.14742/ajet.1690

Brambilla, M., Cabot, J. & Wimmer, M. (2012) Model-Driven Software Engineering in Practice, Synthesis Lectures on Software Engineering, vol. 1. doi: https://doi.org/10.2200/S00441ED1V 01Y201208SWE001

Brown, A. (2004). An introduction to Model Driven Architecture Part I: MDA and today's systems. IBM Developer Works, RationalEdge. [online] Available at: https://www.ibm.com/developerworks/ rational/library/3100-pdf.pdf [Accessed 24 Apr. 2017].

Brown, A. & Conallen, J. (2005) An introduction to model-driven architecture. IBM Developer Works, RationalEdge. [online] Available at: https://www.ibm.com/developerworks/rational/library/may05/ brown/brown.html [Accessed 11 Jul. 2017].

Buchmann, T. (2012) Towards tool support for agile modeling: sketching equals modeling. XM '12 Proceedings of the 2012 Extreme Modeling Workshop, Innsbruck, Austria. 9 – 14. doi: 10.1145/2467307.2467310

Campbell, L. (2007) Instalment on "Learning Object Metadata". - Digital Curation Centre. [online] Available at: https://www.era.lib.ed.ac.uk/bitstream/handle/1842/3334/Campbell%20learning-object-metadata.pdf?sequence=1 [Accessed 11 Jun. 2017].

Cebeci, Z. & Erdogan, Y. (2005) Tree View Editing Learning Object Metadata. Interdisciplinary Journal of Knowledge and Learning Objects 1(1), 100-108.

Cerny, T., Cemus, K., Donahoo, M. & Song, E. (2013) Aspect-driven, data-reflective and context-aware user interfaces design. ACM SIGAPP Applied Computing Review, 13(4), 53-66.

Laverde, A., Cifuentes, Y. & Rodriguez, H. (2007) Toward an instructional design model based on learning objects. Educational Technology Research and Development. 55(6), 671-681.

OpenStax CNX. (1999) Discover learning materials in an Open Space. [online] Available at: https://cnx.org [Accessed 10 Oct. 2017].

Cohen, E. & Nycz, M. (2006) Learning Objects and E-Learning: an Informing Science Perspective. Interdisciplinary Journal of Knowledge and Learning Objects 2(1). Informing Science Institute. ISSN 1552-2237.

Cook, S., Jones, G., Kent, S. & Wills, A.C. (2007) Domain-specific development with Visual Studio DSL tools. Upper Saddle River, NJ: Addison-Wesley.

Coursera Inc. (2017) Coursera. [online] Available at: https://www.coursera.org/ [Accessed 12 May. 2017].

Coyle, K. & Baker, T. (2009) Guidelines for Dublin Core Application Profiles (Working Draft). [online] Available at: http://dublincore.org/documents/profile-guidelines/ [Accessed 14 Aug. 2017].

Coyette, A., Schimke, S., Vanderdonckt, J. & Vielhauer, C. (2007) Trainable Sketch Recognizer for Graphical User Interface Design. Berlin Heidelberg: Human-Computer Interaction-INTERACT. 124-135.

Coyette, A., & Vanderdonckt, J. (2005) A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces. In Proceedings of Human-Computer Interaction-INTERACT. Berlin Heidelberg. 550-564.

Currier, S. (2008) Metadata for Learning Resources: An Update on Standards Activity for 2008. [online] Available at: http://www.ariadne.ac.uk/issue55/currier [Accessed 4 Oct. 2017].

Curriki. (2017) Curriki - a community for teaching or studying: Create, share, and explore high quality K-12 content. [online] Available at: http://www.curriki.org/ [Accessed 12 Mar. 2017].

Dahl, O. & Nygaard, K. (1966) SIMULA an ALGOL-Based Simulation Languag. Communications off the ACM. 9(9), 671-678.

De Robbio, A., Maguolo, D. & Marini, A. (2003) Mathematics Subject Classification and Related Schemes in the OAI Framework. 100-111.

Dejanovic, I., Tumbas, M., Milosavljevic, G. & Perisic, B. (2010) Comparison of Textual and Visual Notations of DOMMLite Domain-Specific Language. ADBIS (local proceedings) 131-136.

The Institute of Electrical and Electronics. (2002) Draft Standard for Learning Object Metadata. [online] Available at: https://biblio.educa.ch/sites/default/files/20130328/lom_1484_12_1_v1_final_draft_0.pdf [Accessed 11 Jun. 2017].

Downes, S. (2004) Learning objects, Resources for learning worldwide, In: Online Education Using Learning Objects, 2nd edition, ed. Mc Greal, R. 20-30. London: Routledge.

Duval, E. & Hodgins, W. (2003) A LOM Research Agenda. [online] Available at: http://www2003.org/cdrom/papers/alternate/P659/p659-duval.html.html [Accessed 12 Apr. 2017].

Duval, E., Hodgins, W., Sutton, S. & Weibel, S. (2002). Metadata Principles and Practicalities. D-Lib Magazine, 8(4).

EdX. (2012) edX. [online] Available at: https://www.edx.org/ [Accessed 21 Oct. 2017].

Narayan, P. (2010) Mathematics Genealogy Networks. [online] Available at: http://docplayer.net/57552146-Mathematics-genealogy-networks.html [Accessed 23 Nov. 2017].

Red Hat (2018), Hibernate. [online] available at: http://hibernate.org/orm/releases / [Accessed 22 Oct. 2017].

Filipovic, M., Kaplar, S., Vaderna, R., Ivković, Ž., Milosavljević, G., Dejanović, I. (2015) Aspect-Oriented Engines for Kroki Models Execution. ICIST 2015 Proceedings Vol. 2. Kopaonik, Serbia. 502-507

Filipovic, M., Vaderna, R., Ivković, Ž, Kaplar, S., Vuković, Ž., Dejanovic, I., Milosavljević, G. & Ivanovic, D. (2017) Application of Kroki Mockup Tool to Implementation of Executable CERIF Specification. Procedia Computer Science 106(1), Elsevier. 245-252.

Fowler, M. & Parsons, R. (2011) Domain-specific languages. Reading, MA: Addison-Wesley.

Friesen, N. (2004) Three Objections to Learning Objects and E-learning Standards. In: Online Education Using Learning Objects, ed. Rory McGreal, 59-70. London: Routledge.

Friesen, N. (2005). CanCore in Canada and Around the World. The International Review of Research in Open and Distributed Learning 6(1).

Goede, K. & Irizarry, J. (2008) Understanding tool requirements for Model Driven Architecture. [online] Available at: http://www.omg.org/mda/mda_files/Understanding_MDA_Tool_Requirements2.pdf [Accessed 28 Sep. 2017].

IEEE Learning Technology Standards Committee (LTSC) (2001) Draft Standard for Learning Object Metadata.

IEEE Xplore. (2002) IEEE 1484.12.1-2002 Standard for Learning Object Metadata. [online] Available at: http://ieeexplore.ieee.org/servlet/opac?punumber=8032[Accessed 2 Mar. 2017].

IMS Global Learning Consortium. (2002) IMS Reusable Definition of Competency or Educational Objective v1 - Best Practice and Implementation. [online] Available at: https://www.imsglobal.org/competencies/rdceov1p0/imsrdceo_bestv1p0.html [Accessed 24 Sep. 2017].

IMS Global Learning Consortium. (2003) IMS Learning Design Best Practice and Implementation. [online] Available at: https://www.imsglobal.org/learningdesign/ldv1p0/imsld_bestv1p0.html [Accessed 14 Nov. 2017].

IMS Global Learning Consortium. (2006) IMS Meta-data Best Practice Guide for IEEE 1484.12.1-2002 Standard for Learning Object. [online] Available at: https://www.imsglobal.org/metadata/ mdv1p3/imsmd_bestv1p3.html [Accessed 12 May. 2017].

IMS Global Learning Consortium. (2001) IMS Learning Resource Meta-Data Best Practice and Implementation Guide [online] Available at: http://www.imsglobal.org/metadata/imsmdv1p2p1/imsmd_bestv1p 2p1.html [Accessed 10 Jun. 2017].

IMS Global Learning Consortium. (2004) IMS Content Packaging Best Practice and Implementation Guide. [online] Available at: http://www.imsglobal.org/content/packaging/cpv1p1p4/imscp_bestv1p1p4.html [Accessed 14 Jun. 2017].

A. Ip, A. Young, I. Morrison (2002) Learning Objects - Whose are they? Proceedings of the 15th Annual Conference of the National Advisory Committee on Computing Qualifications ISBN 0-473-08747-2. 315-320.

JHipster. (2013). JHipster project. [online] Available at: http://www.jhipster.tech/ [Accessed 23 Jul. 2017].

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. PhD dissertation, University of California, Irvine.

Jovanovic, D., Zizovic, M. & Milosevic, D. (2012) ILOMPEL: Information and metadata modeling for personalized e-learning. Scientific Research and Essays Vol. 7(11), 1212-1229.

Kaplar, S., Filipovic, M., Milosavljevic, G. & Sladic, G. (2015) Kroki Administration Subsystem Based on RBAC Standard and Aspects. In proceedings of ICIST 2015 Conference, Kopaonik, Serbia. 61-66

Karakostas, B. & Zorgios, Y. (2008). Engineering Service Oriented Systems: A Model Driven Approach 1st Edition. IGI Global, ISBN: 978-1599049687.

Kleppe, A., Warmer, J. & Bast, W. (2003) The Model Driven Architecture : Practice and Promise. Addison Wesley.

Kroki team (2018a) Kroki source code. [online] Available at: https://github.com/KROKIteam/KROKI-mockup-tool [accessed 3. Mar. 2017)

Kroki team (2018b) Kroki MDE tool - A rapid prototyping tool for participatory development of business applications. [online] Available at: www.kroki-mde.net [accessed 3 Mar. 2017].

L'Allier, J. J. (1997) Frame of Reference: NETg's Map to the Products, Their Structure and Core Beliefs. [online] Available at: http://web.archive.org/web/20020615192443/www.netg.com/research/whitepapers/frameref.asp [accessed 30 Mar. 2017].

Lama, J. (2001) Conceptual Model of reusable Object-based e-Learning.

Lewis, J. R. (1995) IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. International Journal of Human-Computer Interaction. 7(1), 57-78.

SREB-SCORE (2007), Learning Object Metadata SREB-SCORE Initiative. [online] Available at: http://txlor.utsa.edu/docs/SREB_learning_object_metadata.pdf [Accessed 13 Oct. 2017].

Littlejohn, A. & Buckingham, S. (2003) Reusing Online Resources: A Sustainable Approach to E-Learning. Kogan Page: London. ISBN: 0 7494 3950 5

MERLOT. (1997) MERLOT. [online] Available at: https://www.merlot.org/merlot/index.htm [Accessed 11 Aug. 2017].

Milosavljevic, G., Ivanovic, D., Surla, D. & Milosavljevic, B. (2011) Automated construction of the user interface for a CERIF-compliant research management system. The Electronic Library 29 (5), 565-588.

Nguyen, V., Qafmolla, X. & Richta, K. (2014) Domain Specific Language Approach on Model-driven Development of Web Services. Acta Polytechnica Hungarica. 11(8), 121-138.

National Institute of Standards and Technology. (2001) Common Industry Format for Usability Test Reports [online] Available at: http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/common-industry-format.pdf [Accessed 24 Jun. 2017].

Nilsson, M., Baker, T. & Johnston, P. (2008) DCMI: The Singapore Framework for Dublin Core Application Profiles. [online] Available at: http://dublincore.org/documents/singapore-framework/ [Accessed 11 Apr. 2017].

Nilsson, M., Palmler, M. & Brase, J. (2003) The LOM RDF binding - principles and implementation. The 3rd annual ARIADNE conference Proceedings of the 3rd annual ARIADNE conference, Katholieke Universiteit Leuven, Belgium. ISSN 1403-0721.

Carnegie Mellon University. (2017) Open Learning Initiative. [online] Available at: http://oli.cmu.edu/ [Accessed 12 Oct. 2017].

Massachusetts Institute of Technology. (2001) MIT OpenCourseWare. [online] Available at: https://ocw.mit.edu/index.htm [Accessed 22 Oct. 2017].

The Open University. (1999) OpenLearn. [online] Available at: http://www.open.edu/openlearn/ [Accessed 15 Oct. 2017].

Pawlak, R., Seinturier, L. & Retaille, J. (2005) Foundations of AOP for J2EE Development. Apress. ISBN: 978-1-59059-507-7. doi: 10.1007/978-1-4302-0063-5

Paunovic, V. & Domazet, D. (2013) Set of Metadata Established for Application in Learning Materials Developed for BMU. The Fourth International Conference on e-Learning, 143-148.

Perisic, B., Milosavljevic, G., Dejanovic, I. & Milosavljevic, B. (2011) UML profile for specifying user interfaces of business applications. Computer Science and Information Systems 8(2), 405-426.

Plimmer, B., Apperley, M. (2004) INTERACTING with sketched interface designs: an evaluation study. In CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04). ACM, New York, NY, USA, 1337-1340. DOI: https://doi.org/10.1145/985921. 986058

Polsani, P. (2005) Use and Abuse of Reusable Learning Objects. Journal of Digital Information. 3(4), 10 p.

Poole, J. (2001) Model-Driven Architecture: Vision, Standards And Emerging Technologies. n In ECOOP 2001, Workshop on Metamodeling and Adaptive Object Models .

Powell, A., Nilsson, M., Naeve, A., Johnston, P. & Baker, T. (2007) DCMI: DCMI Abstract Model. [online] Available at: http://dublincore.org/documents/abstract-model/ [Accessed 14 Oct. 2017].

Restlet. (2017). Restlet Framework | Overview. [online] Available at: https://restlet.com/open-source/ [Accessed 24 Nov. 2017].

Riley, J. (2017) Understanding Metadata: What is Metadata, and What is it For? Baltimore. ISBN: 978-1-937522-72-8.

Rivero, J., Grigera, J., Rossi, G., Robles Luna, E., Montero, F. & Gaedke, M. (2014) Mockup-Driven Development: Providing agile support for Model-

Driven Web Engineering. Information and Software Technology 56(6). doi: 10.1016/j.infsof.2014.01.011

Ros, M. (2005) Sequencing of Contents of Learning Objects. Revista de Educación a Distancia, núm. 13, diciembre, 2005, pp. 1-14

Roy, D., Sarkar, S. and Ghose, S. (2010) A comparative study of learning object metadata, learning material repositories, metadata annotation & an automatic metadata annotation tool. In: Advances in Semantic Computing, 103-126.

Savić G., Segedinac M., Konjović Z. (2013) The Semantic Annotation of Digital Learning Content Using Competence-based Knowledge Space Theory. Transactions on Internet Research (ISSN: 1820-4503), Vol 9 (1), pp. 39 – 44

Savić, G., Segedinac, M., Konjović, Z. (2012), „Automatic Generation of E-Courses Based on Explicit Representation of Instructional Design“, Computer Science and Information Systems, Vol. 9, No. 2, pp. 839 – 869.

Stuempel, H., Salokhe, G., Aubert, A., Keizer, J., Nadeau, A., Katz, S. & Rudgard, S. (2007.) Metadata Application Profile for Agricultural Learning Resources. In Metadata and Semantics Research (MTSR) Conference, Corfu (Greece), 11-12 October 2007.

The Apache Software Foundation. (1999) Apache Tomcat. [online] Available at: http://tomcat.apache.org/ [Accessed 19Oct. 2017].

Todorova, M. & Petrova, V. (2003). Learning objects. International Conference on Computer Systems and Technologies – CompSysTech'2003.

Tyagi, S. (2006). RESTful Web Services. Oracle Technology Network [online] Available at: http://www.oracle.com/technetwork/articles/ javase/index137171.html [Accessed 14 Oct. 2017].

Udacity Inc. (2011) Udacity [online] Available at: https://www.udacity.com [Accessed 15 Oct. 2017].

Udemy Inc. (2010) Udemy [online] Available at: https://www.udemy.com [Accessed 15 Oct. 2017].

Boldt, N. & Steinberg, D. (2006) Introduction to the Eclipse Modeling Framework. EclipseCon 2006 [online] Available at:

https://www.eclipse.org/modeling/emf/docs/presentations/EclipseCon/EclipseCon2006_EMF_Intro.pdf [Accessed 14 Oct. 2017].

Weibel, S. & Koch, T. (2000). The Dublin Core Metadata Initiative. D-Lib Magazine, 6(12).

Wiley, D. (2002) The Reusability Paradox. [online] Available at: http://opencontent.org/docs/paradox.html [Accessed 14 Oct. 2017].

Yeoman (2017) Yeoman - the web's scaffolding tool for modern webapps. [online] Available at: http://yeoman.io/ [Accessed 23 Oct. 2017].

zbMATH. (2010). Mathematics Subject Classification – MSC2010. [online] Available at: https://zbmath.org/classification/ [Accessed 12 Oct. 2017].

Ključna dokumentacijska informacija

Redni broj, **RBR**:	
Identifikacioni broj, **IBR**:	
Tip dokumentacije, **TD**:	Monografska publikacija
Tip zapisa, **TZ**:	Tekstualni štampani dokument
Vrsta rada, **VR**:	Doktorska disertacija
Autor, **AU**:	Amel Abdyssalam Alhaag
Mentor, **MN**:	dr Goran Savić, redovni profesor, Fakultet tehničkih nauka, Novi Sad
Naslov rada, **NR**:	Model-Driven Software Architecture for the Management of Educational Resources Metadata
Jezik publikacije, **JP**:	engleski
Jezik izvoda, **JI**:	srpski
Zemlja publikovanja, **ZP**:	Srbija
Uže geografsko područje, **UGP**:	Vojvodina
Godina, **GO**:	2018.
Izdavač, **IZ**:	Autorski reprint

Mesto i adresa, **MA**:	Novi Sad, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6
Fizički opisa rada, **FO**: (broj poglavlja/strana/lit. citata/ tabela/slika/grafika/priloga)	5/138/102/11/37/0/0
Naučna oblast, **NO**:	Primenjene računarske nauke i informatika
Naučna disciplina, **ND**:	Elektronsko učenje
Predmetna odrednica/Ključne reči, **PO**:	elektronska nastava, upravljanje obrazovnim resursima, metapodaci, izvršiva platforma, pristup vođen modelom
UDK broj, **UDK:**	
Čuva se, **ČU:**	Biblioteka Fakulteta tehničkih nauka, Trg Dositeja Obradovića 6, Novi Sad
Važna napomena, **VN**:	Nema

Izvod,
IZ:

Cilj – Cilj disertacije je da se omogući dinamičko prilagođavanje metapodataka koji opisuju obrazovne resurse u digitalnim repozitorijumima.

Metodologija - Postoji potreba da se u digitalnim repozitorijumima obrazovni resursi opišu putem skupa metapodataka koji je specifičan za određenog korisnika ili domen. Obzirom da korisnici ne mogu samostalno da ručno vrše izmenu softverske aplikacije, pristup predložen u ovoj disertaciji se zasniva na tehnikama modelom vođenog razvoja softvera, koji treba da omogući prilagođavanje softverske aplikacije programski, bez potrebe za razvojem ili naručivanjem nove aplikacije. Da bi se predloženo rešenje verifikovalo, sproveden je eksperiment koji evaluira njegove karakteristike.

Rezultati - U disertaciji je predložena softverska platforma za upravljanje obrazovnim resursima opisanim dinamički proširivim skupom metapodataka. Platforma omogućuje kreiranje modela podataka koji se programski transformišu u veb aplikaciju za upravljanje obrazovnim resursima. Na ovaj način, korisnik može da kreira sopstveni model metapodataka koji je odgovarajući u određenom domenu.

Ograničenja istraživanja/implikacije – Rešenje verifikovano od strane korisnicima sa određenim tehničkim znanjem. Potrebno je istražiti prikladnost platforme za domenske eksperte sa ograničenim tehničkim znanjem, koji treba da definišu nove skupove metapodataka u svom domenu.

Praktične implikacije – Rešenje se može koristiti u digitalnim repozitorijuma koji skladište raznolike obrazovne resurse. Svaki resurs može biti opisan koristeći metapodatke iz domena kojem resurs pripada.

Originalnost/vrednost - Digitalni repozitorijumi standardno opisuju

obrazovne resurse koristeći neki generalni skup metapodataka, koji je više fokusiran na fizičke karakteristike resursa, umesto na njihovo značenje. Predloženo rešenje uvodi proizvoljnu domenski-zavisnu semantiku u opis resursa, čime se unapređuje njihovo dobavljanje.

Datum prihvatanja teme, **DP**:	31.05.2017.
Datum odbrane, **DO**:	
Članovi komisije, **KO**:	
Predsednik:	dr Milan Segedinac, docent, Fakultet tehničkih nauka, Novi Sad
Član:	dr Gordana Milosavljević, vanredni profesor, Fakultet tehničkih nauka, Novi Sad
Član:	dr Dušica Rodić, docent, Prirodno-matematički fakultet, Novi Sad
Član:	dr Vlado Simeunović, redovni profesor, Pedagoški fakultet, Bijeljina
Član, mentor:	dr Goran Savić, docent, Fakultet tehničkih nauka, Novi Sad

Key words documentation

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	Monograph publication
Type of record, **TR**:	Textual printed material
Content code, **CC**:	PhD thesis
Author, **AU**:	Amel Abdyssalam Alhaag
Mentor, **MN**:	Goran Savić, PhD, assistant professor, Faculty of Technical Sciences, Novi Sad
Title, **TI**:	Model-Driven Software Architecture for the Management of Educational Resources Metadata
Language of text, **LT**:	English
Language of abstract, **LA**:	Serbian
Country of publication, **CP**:	Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2018.
Publisher, **PB**:	Author reprint

Publication place, **PL**:	Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad, Serbia
Physical description, **PD**: (chapters/pages/ref./ tables/pictures/graphs/appendixes)	5/138/102/11/37/0/0
Scientific field, **SF**:	Applied computer science and informatics
Scientific discipline, **SD**:	E-learning
Subject/Key words, **SX**:	e-learning, educational resources, metadata, executable platform, model-driven approach
UC:	
Holding data, **HD:**	Library of Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad, Serbia
Note, **N**:	None

Abstract,
AB:

Purpose – The purpose of the research is to enable dynamic customization of metadata that describe educational resources in digital repositories.

Design/methodology/approach – Users need to describe educational resources in digital repositories according to the user-specific metadata set. Since users are mostly unskilled to customize the software application manually, our approach relies on the techniques of the model-driven software engineering, which should allow customization of the software application programmatically with no need to develop or order a new software application. In order to verify the proposed solution, we conducted an experiment which evaluated its characteristics.

Findings – A software platform for managing educational resources described by dynamically extendable metadata is proposed. The platform enables creating data models which are programmatically transformed to the web application for the management of educational resources. In this way a user can create their own model of metadata that is relevant in a particular domain.

Research limitations / implications – The solution has been verified by users with technical knowledge. We should still explore the appropriateness of the platform for domain experts with little technical knowledge who would define new metadata in their domain.

Practical implications – The solution can be used for digital repositories that store diverse educational resources. Each resource could be described using metadata that relates to the domain the resource belongs to.

Originality/value – Digital repositories standardly describe educational

resources using some general metadata, which are more focused on the physical characteristics of resources rather than their semantics. The proposed solution introduces custom domain-specific semantics into the resources' description, which improves their retrieval.

Accepted by te Scientific Board on, **ASB**:	31.05.2017.
Defended on, **DE**:	
Defended board, **DB**:	
President:	Milan Segedinac, PhD, assistant professor, Faculty of Sciences, Novi Sad
Member:	Gordana Milosavljević Segedinac, PhD, associate professor, Faculty of Sciences, Novi Sad
Member:	Dušica Rodić, PhD, assistant professor, Faculty of Sciences, Novi Sad
Member:	Vlado Simeunović, PhD, full professor, Faculty of Pedagogy, Bijeljina
Member, mentor:	Goran Savić, PhD, assistant professor, Faculty of Technical Sciences, Novi Sad