THE RESEARCH ON MODEL TRANSFORMATIONS, BASED ON DOMAIN METAMODEL, FOR DESIGNING REQUIREMENTS SPECIFICATIONS

Neringa Žemaitytė

October 2016

Technical Report MII-DS-07T-16-10
VU Institute of Mathematics and Informatics, Akademijos str. 4, Vilnius LT-08663, Lithuania

www.mii.lt
Abstract

In recent years, ontologies – shared conceptualizations of some domain – are increasingly seen as the key to further automation of information processing. There are many applications of such an approach, e.g. automated information processing, information integration or knowledge management, to name just a few.

Although many approaches for representing and applying ontologies have already been devised, they still haven’t found their way into enterprise applications.

Ontologies, for software design and development, can be used with the following objectives [29] [26]:

• Specification: ontologies are used to specify either the requirements and components definitions (informal use) or the system’s functionality.

• Confidence: ontologies are used to check the system’s design.

• Reusability: ontologies could be organized in modules to define domains, subdomains and their related tasks, which could be later reused and/or adapted to other problems.

• Search: ontologies are used as information repositories.

• Reliability: ontologies could be used in (semi)–automatic consistency checking.

• Maintenance: ontologies improve documentation use and storage for system’s maintenance.

• Knowledge acquisition: ontologies could be used as a guide for the knowledge acquisition process. Within Software Engineering, two main roles for ontologies have been considered [30]:

• Ontologies for the Software Engineering Process: the definition, re–use and integration of software components is aided by the use of ontologies as the conceptual basis.

• Ontologies for the Software Engineering Domain: the use of ontologies to describe the structure and terminology of the software engineering domain itself.

These are the key reasons why ontology is included in this research to represent knowledge base. The main considerations in the research will be requirements specification generation from ontology. Further work will include the research on ontologies, their languages and tools, ontology based model transformations and requirements engineering.
Preliminary objectives: a research on designing requirements specifications, based on domain metamodel.

Preliminary goals:
1) To make a literature research on designing requirements specifications, based on domain metamodel;
2) To make a literature research on using ontologies in requirements engineering;
3) To present a methodology on requirements specification generation from domain ontology.

Keywords: Knowledge base, MDA, Ontology Engineering, Model transformations, Software Requirements, meta-model
# Contents

1. Introduction .................................................................................................................. 6
2. Theory of ontologies ....................................................................................................... 6
3. Roles of ontologies in software engineering ................................................................. 8
4. Model Driven Architecture (MDA) to Ontologies ......................................................... 9
5. Benefits of using ontologies in software engineering .................................................... 10
6. Problems about ontologies ............................................................................................ 11
7. Languages .................................................................................................................... 13
8. Ontologies in Requirements Engineering ..................................................................... 13
9. Ontology based model transformations ........................................................................ 16
10. Other activities during 2015-2016 year of study .......................................................... 19
11. References .................................................................................................................. 20
12. Appendixes .................................................................................................................. 23
1 Introduction

Software development is more demanding nowadays as software applications have become increasingly more complex. New procedures and techniques are sought to simplify the software engineering processes with the aim of shortening development time and costs by re–using components. Therefore, different stakeholders, heterogeneity and new software features make software development a heavily knowledge–based process [27] [26].

To reduce this complexity, the use of ontologies might prove useful. Ontologies allow for the definition of a common vocabulary and framework among users (either human or machines). Software development has benefited from this conceptual modeling, allowing a common understanding of the concepts involved in the software process. Ontologies also ease the integration issues that usually appear while developing software applications [26].

Arising challenges, like to develop more and more complex systems efficiently, have lead to the trend of enterprise and application modeling becoming a provider of intelligent infrastructure services [18]. This drives the need to achieve interoperability in modeling enterprises and application systems by semantic enrichment, which can be met by integrating ontology technology into services of an intelligent infrastructure.

Due to the existence of a many different modeling languages and metamodels used for modeling the same semantic concepts, ontologies will help to facilitate the integration of the diversity of modeling approaches. The Model-Driven Architecture, as a framework for model-driven development, can be realized more efficient by extending specifications of models and model transformations with ontologies, since this semantic information can be used to automate the integration of various modeling languages. Ontology technology like ontology mappings or inference machines will provide new possibilities for analysis of metamodels and processing metamodels.

2 Theory of ontologies

In recent years ontologies – shared conceptualizations of some domain – are increasingly seen as the key to further automation of information processing. There are many application of such an approach, e.g. automated information processing, information integration or knowledge management, to name just a few. Especially after Tim Berners-Lee coined the vision of the Semantic Web, where Ewb pages are
annotated by ontology-based meta-data, the interest in ontology research increased, in hope of finding ways to off-load large-volume information processing from the human user to autonomous agents [2].

Although many approaches for representing and applying ontologies have already been devised, they haven’t found their way into enterprise applications [2].

The word ontology comes from the Greek ontos (being) and logos (word). It denotes the science of being and the descriptions for the organization, designation and categorization of existence [33]. Carried over to computer science in the field of artificial intelligence and information technologies, an ontology is understood as a representational artifact for specifying the semantics or meaning about the information or knowledge in a certain domain in a structured form [34] [37].

An ontology is an explicit specification of a conceptualization [12]. It is a designed artifact that formally represents agreed semantics of a domain interest in computer resources [12]. This enables the sharing and reuse of information and allows for the interoperability of information systems [13]. Although not a new field, ontology research has recently received renewed interest and attracted many other fields such as the semantic web, databases, electronic commerce, knowledge management, electronic learning, information retrieval, digital library, bioinformatics, geographical information systems, software engineering, intelligent systems and natural language processing. Thus, we can classify the ontology applications as reported in Jasper et al. [76], Pisanelli et al. [14], Fensel [15], Mizoguchi [1] and the most comprehensive survey by Hart et al. [16].

Mizoguchi [1] defines five typical types of ontology application including: (a) ontology as a common vocabulary, (b) ontology as assisting of information access, (c) ontology as the medium for mutual understanding (d) ontology as specification and (e) ontology as foundation of knowledge systematization.

Ontologies, which are formal, explicit specifications of shared conceptualisations, encourage collaborative development by different experts. Ontologies capture knowledge at the conceptual level, thus enabling ID experts to directly manipulate them without the involvement of a knowledge engineer. In its simplest form, an ontology is a taxonomy of terms (i.e. a “shared lexicon”) whereas more expressive approaches such as the Ontology Web Language OWL (W3C – OWL, 2003) encode knowledge in logical axioms. Ontologies support knowledge reuse by
allowing more specific concepts to inherit the properties of those concepts they specialise. This also allows the representation of knowledge at different abstraction levels. In this way, instructional theories at a high level of abstraction can be related to concrete teaching methods [4].

Ontology–driven software development or engineering has been defined as an approach that, based on ontologies, takes into account semantic constraints, adapting in a dynamic way to new constraints [28]. It could be considered a particular case of model–driven software, where models are based on ontologies at different levels of abstraction [26].

3 Roles of ontologies in software engineering

Ontologies, for software design and development, can be used with the following objectives [29] [26]:

- Specification: ontologies are used to specify either the requirements and components definitions (informal use) or the system’s functionality.
- Confidence: ontologies are used to check the system’s design.
- Reusability: ontologies could be organized in modules to define domains, subdomains and their related tasks, which could be later reused and/or adapted to other problems.
- Search: ontologies are used as information repositories.
- Reliability: ontologies could be used in (semi)–automatic consistency checking.
- Maintenance: ontologies improve documentation use and storage for system’s maintenance.
- Knowledge acquisition: ontologies could be used as a guide for the knowledge acquisition process. Within Software Engineering, two main roles for ontologies have been considered [30]:
  - Ontologies for the Software Engineering Process: the definition, re–use and integration of software components is aided by the use of ontologies as the conceptual basis.
  - Ontologies for the Software Engineering Domain: the use of ontologies to describe the structure and terminology of the software engineering domain itself.
4 Model Driven Architecture (MDA) to Ontologies

Ontologies and MDA are two technologies being developed in parallel, but by different communities [5]. They have common points and issues and can be brought closer together [8] [5] [7]. Therefore, to bring software engineering practitioners and ontologies closer, many researchers suggest the use of Unified Modeling Languages (UML) in ontology development (e.g., [9] [10] [8] [7]). The main question they want to answer is how to use UML as well-accepted modeling languages for developing and using ontologies in real world applications. Although the ontology concepts are coincidently similar to object-oriented paradigms, it has some limitations mainly regarding the concept of property. Because of these discrepancies, initially, we could only use UML in the beginning of ontology development. However, there is a significant movement in this research to overcome these limitations using UML extensions (i.e. UML profiles) as implemented in [8] [7]. As a result, the Object Management Group (OMG) has established Ontology Definition Metamodel (ODM) as a MDA standard metamodel for modeling ontology [8]. The ODM defines concrete abstract syntaxes (i.e. OWLDataTypeProperty, OWLClass) for modeling ontology that can be represented by using UML profiles [8]. The ODM is centrally based on UML and the W3C Web Ontology Language (OWL) recommendation [11]. In terms of ontology modeling, on one hand, the UML provides powerful graphical modeling capabilities and widely supported tools (i.e. Rational Rose, Poseidon, Magic Draw, ArgoUML, etc). In addition, since the UML and ODM are MOF-compliant languages, it is possible to store ontologies in MOF-based repositories, to store ontologies diagrams in a standard way (UML2 XMI), as well as to share and interchange ontologies using XMI [8] [7]. However, on the other hand, we note that not all OWL features could be represented by UML. We will use ODM and UML profiles defined in [8] for representing ontologies and designing the server. In addition, UML is currently a de facto standard modeling language. There is a growing interest in its adoption as a language for conceptual modeling and ontological representation (e.g., [9] [10] [8][7]).

Like described in [19] the Model-driven Architecture (MDA) provides a framework for software development focusing on models in all phases of development. Models are more than abstract descriptions of systems, as they are used for model- and codegeneration – they are the key part of the definition of a software system. Since in
MDA abstract models are refined to more concrete models, (automated) model transformations are very important [21]. For MDA methodologies we can distinguish two kinds of model transformations. In *vertical model transformations* models from higher level of abstraction are transformed to models of lower level of abstraction, e.g. platform independent models to platform specific models. There, knowledge of platforms is encoded into transformations, reused for many systems rather than redesigned for each new system. *Horizontal model transformations* are used for describing mappings between models of the same level of abstraction. By relating concepts of various model types, knowledge of modelling domains is encoded into transformations, enabling the integrated use of different models without having to specify interrelationships between each set models manually.

In MDA a model is a representation of a part of the functionality, structure and behaviour of a system. A specification is said to be *formal* when it is based on a language with well-defined structure (‘syntax’) and meaning (‘semantics’). Thus MDA models must be paired unambiguously with a definition of the modeling language syntax and semantics [22]. Most metamodels have, despite of well-defined syntax, descriptions of their semantic concepts and dynamic semantics, which is neither formal nor machine understandable. Taking the idea of the semantic web [23], where the word semantic means machine understandable to modeling, metamodels have to be grounded using ontology meta data. This enables machines to understand the meaning of metamodels’ concepts. In our approach we lift the syntactical (meta-)model description by semantic enrichment into ontologies describing the concepts of the model in a machine understandable form. Model transformations are defined on top of those ontologies [24].

By enriching model-driven development with ontologies a mutual understanding for conceptual integration can be achieved [25] [24].

5 **Benefits of using ontologies in software engineering**

Ontologies provide benefits regarding the process of software development, which could be summarized as follows, [31] [29] [26]:

- Ontologies provide a representation vocabulary specialized for the software process, eliminating conceptual and terminological mismatches.
• The use of ontologies and alignment techniques allow to solve compatibility problems without having to change existing models.
  • Ontologies might help to develop benchmarks of software process by collecting data on the Internet and the use of the Semantic Web.
  • Ontologies allow both to transfer knowledge and to simplify the development cycle from project to project.
  • Ontologies promote common understanding among software developers, as well as being used as domain models.
  • Ontologies allow for an easier knowledge acquisition process, by sharing a same conceptualization for different software applications.
  • Ontologies allow to reduce terminological and conceptual mismatches, by forcing to share understanding and communications among different users during the ontological analysis.
  • Ontologies also provide for a refined communication between tools forming part of an environment.
  • Ontologies, when as machine–understandable representations, help in the development of tools for software engineering activities.

6 Problems about ontologies
Although ontologies are considered a useful element within software engineering activities, some issues should still be born in mind when developing ontology–based software development projects [30] [26]:
  • The ontology–based approach is adequate for those software development projects that belong to a set of projects within the same domain.
  • The ontology–based approach allow to extend the notion of reusability to the modeling phase, not only the usual implementation one. Therefore, ontologies could be considered reusable model components in the system.
  • Model–Driven Developments can benefit from the use of ontologies as model re–use mechanisms.
  • The ontology–based approach affects all the software development process phases, from requirement analysis and domain analysis to integration, deployment and use of the developed software.
• The ontology–based approach allow ontologies to be used to facilitate software development in the long term, as well as addressing interoperability and re–use issues.

Furthermore, ontologies should exhibit some specific properties to facilitate their use within the software engineering community [González–Pérez and Henderson–Sellers, • Completeness: to assure that all areas of software development are covered. It could be achieved by paying attention to the different activities carried out by software development enterprises.

• Unambiguity: to avoid misinterpretations. Ambiguity could be avoided by using both concise definitions of concepts and semi–formal models.

• Intuitive: to specify concepts familiar to users’ domain.

• Genericity: to allow the ontology to be used in different contexts. It could be done by keeping the ontology as small as possible, to achieve maximum expressiveness while being minimal.

• Extendability: to facilitate the addition of new concepts. It could be achieved by providing appropriate mechanisms defining how to extend the ontology.

The relevant problems identified by Mizoguchi and Bourdeau (2000) include [1]:

1. The “conceptual gap” between authoring systems and authors. In Artificial intelligence, the “knowledge level” is explicitly distinguished from the “symbol level” in which knowledge is encoded (Newell, 1982). The traditional approach to the development of rule–based systems involves a knowledge engineer, who encodes the knowledge elicited from a domain expert. Unfortunately, this results in a gap between the conceptualisation of the ID expert and the corresponding computer representation. Consequently, (i) the development, (ii) the verification & validation, and (iii) the maintenance of rule bases can become rather difficult.

2. The lack of theory awareness of systems. Heuristic rules cannot explicitly represent the theories they commit to.

3. The difficulty to integrate the latest research results. The lack of theory awareness prevents the adaptation of rule bases in order to accommodate subsequent results of ID research [4].
7 Languages

For ontology representation in a machine-interpretable way, different languages exist. Ontology languages are usually declarative languages commonly based on either first-order logic or on description logic. Ontology languages based on first-order logic have high expressive power, but computational properties such as decidability are not always achieved due to the complexity of reasoning [35]. The most popular language based on description logic is OWL DL, which have attractive and well-understood computational properties [36]. Another relevant language in Ontological Engineering is the Resource Description Framework (RDF). RDF was originally meant to represent metadata about web resources, but it can also be used to link information stored in any information source with semantics defined in an ontology.

The basic construction in RDF is an (Object, Attribute, Value) triplet: an object O has an attribute A with value V. A RDF-triplet corresponds to the relation that could be written as \((O, A, V)\), such as for example \((\text{Professor};\text{teaches};\text{ArtificialIntelligence})\)[37].

Many ontology languages have been developed, each aimed at solving particular aspects of conceptual modeling. Some of the, such as RDS(S) are simple languages offering elementary support for ontology modeling for the Semantic Web. There are other, more complex languages with roots in formal logic, focused around inference – ways to automatically infer facts not explicitly present in the model.

8 Ontologies in Requirements Engineering

The study of an information system requirements should result in the establishment of well-defined functionalities and attributes agreed by the stakeholders. If the functionalities are defined as incomplete or incorrect, the software may not meet the expectations of users. Factors that could lead to an inadequate process of requirements elicitation can be [37]:

Ambiguous Requirements: which produce lost of time and repeated work. Their origin resides in the diverse stakeholders, who produce different interpretations of the same requirement. Moreover, one stakeholder can interpret the same requirement in diverse ways. The ambiguity conduces to mistaken product tests.
Insufficient Specifications: they produce the absence of key requirements. This conduces to developers’ frustration, because they base their work in incorrect suppositions and, so, the required product is not developed, which displeases the clients.

Requirements not completely defined: they make impossible the project secure planning and its monitoring. The poor requirements understanding leads to optimistic estimations, which return against when the agreed limits are surpassed.

Dynamic and changing requirements: which require constant requirements revision in order to help to understand new clients needs and to identify how they can be satisfied.

In order to reduce the negative effects of the previous factors on the RE processes, the ontologies can be used. The potential uses of ontologies in RE include the representation of: (i) The requirements model, imposing and enabling a particular paradigmatic way of structuring requirements, (ii) Acquisition structures for domain knowledge, and (iii) The knowledge of the application domain. Figure 2 shows a framework that depicts the interrelations between the ontologies previously described and a requirement specification document [37].

An ontology-based requirements specification tool may help to reduce misunderstanding, missed information, and help to overcome some of the barriers that make successful acquisition of requirements so difficult [38].

Simplified, ontologies are structured vocabularies having the possibility of reasoning. It includes definitions of basic concepts in the domain and relations among them. It is important that the definitions are machine-interpretable and can be processed by algorithms.

Why would someone want to develop an ontology?

Some of the reasons are [38]:

- To share common understanding of the structure of information among people or software
- agents.
- To enable reuse of domain knowledge.
- To make domain assumptions explicit.
- To separate domain knowledge from the operational knowledge.
- To analyze domain knowledge.
For an ontology being successfully used in requirements checking, it has to have the following properties: completeness, correctness, consistency, and unambiguity.

The intuitive meaning is:

- **correctness** means that the knowledge in the ontology does not violate the domain rules that correctly represent the reality;
- **consistency** means that there are no contradictory definitions in ontology;
- **completeness** means that the knowledge in ontology describes all aspects of the domain;
- **unambiguity** means that the ontology has defined a unique or unambiguous terminology.

There are not obscure definitions of ontology concepts, i.e. each entity is denoted by only one, unique name, all names are clearly defined and have the same meaning for the analyst and all stakeholders [38].

The architecture of ODRE (Ontology-Driven Requirement Engineering)

In Figure 1 it is presented the architecture of ODRE. A so-called Requirement Meta Model is generated from the knowledge of the requirements analysis and builds the TBox of the Requirements Ontology. It formalizes the RE concepts, as well as relationships between requirement artefacts. This domain independent ontology TBox can be instantiated with the requirement artefacts for a particular project. This way, the
ABox is generated from the results of requirement analysis of an application project. It builds the requirement Specification Base [39].

9 Ontology based model transformations

As a realization of semantic-based model transformations, ontology-based model transformation needs the following parts to achieve an increased level of abstraction [24]:

• **Semantic Transformation:** A *semantic transformation* is a transformation specification describing a transformation between two ontologies. A semantic transformation is specified between a source ontology and a target ontology (see figure 1), but it can also be bidirectional. For horizontal transformations the semantic transformation normally is the ID.

• **Syntax-semantic Binding:** The *syntax-semantic binding* specifies the connection between syntax (metamodels) and semantics (ontologies).

• **MO-Binding:** *(Metamodel-ontology) MO-Bindings* specify how semantic information can be derived model elements.

• **OM-Binding:** *(Ontology-metamodel) OM-Bindings* specify how ontology elements are expressed in models.

\[\text{Figure 1: Concept of ontology-based model transformation}\]
In figure 1 we can see concepts and design of ontology-based model transformation. A transformation is specified on the basis of ontologies, called semantic transformation. The transformation between the two ontologies, a source ontology and a target ontology, is described by the means of this semantic transformation. Elements of the source ontology are transformed to elements of the target ontology. The connection between syntax defined in metamodels and the semantics of the ontology elements has to be defined by a syntax-semantic binding, done with a MO-Binding and an OM-Binding. In a mid-term perspective these bindings have to be derived semiautomatically from already existing transformations and bindings in combination with metamodel analysis [24].

Figure 2 shows the overall approach of ontology-based model transformation. A combination of one semantic transformation, one MO-Binding and one OM-Binding form a transformation configuration. A transformation configuration is the basis for an automated generation of common model transformations. A generator for model transformations takes a transformation configuration as well as appropriate metamodel- and ontology-definitions as input and outputs a model transformation specified in an MII-DS-07T-16-10
intermediate model transformation language. Introducing an intermediate transformation language aims to obtain a common representation of model transformations independent to specific transformation languages, maybe on the basis of a QVT common language and comparable to the platform independent model in the MDA approach. The generated model transformation is input to arbitrary MDA-tools performing model transformations [24].

Model transformations specified between ontologies, will lead to interoperable model transformations independent of methodologies’ tailoring to specific projects. The specification of multiple model transformations will be reduced to few or even one ontology-based model transformation. Furthermore, one specification of an ontology-based model transformation can be used to generate multiple transformations for specific modeling environments (and their transformation languages) automatically [24].
10 Other activities during 2015-2016 year of study

Exams taken

<table>
<thead>
<tr>
<th>Title of the lecture</th>
<th>Credits</th>
<th>Planned date</th>
<th>Exact date</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informatikos ir informatikos inžinerijos tyrimo metodai ir metodika</td>
<td>9</td>
<td>2016.05.20.</td>
<td>2016.06.09</td>
<td>6</td>
</tr>
<tr>
<td>Informacijos poreikių specifikavimas</td>
<td>7</td>
<td>2017.03.</td>
<td>2016.09.27</td>
<td>9</td>
</tr>
<tr>
<td>Žiniomis grindžiama kompiuterizuota informacijos sistemų inžinerija</td>
<td>7</td>
<td>2016.10.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sistemų analizės technologijos</td>
<td>7</td>
<td>2017.09.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Conferences

- DATAMSS2015 - 7th International Workshop Data Analysis Methods for Software. Date: gruodžio 3-5 d., 2015 m., Druskininkai.

Publications


The text of the publication will be presented in the Appendix No 1.
11 References


[38] Petr Kroha, Jose Emilio Labra Gayo (2008) Using Semantic Web Technology in Requirements Specifications
12 Appendixes

Appendix No 1

Ontologies and Enterprise Modelling based UML Models Generation Process (UML Interaction Overview Case)

Audrius Lopata, Ilona Veitaitė, Neringa Žemaitytė

Vilnius University, Kaunas Faculty of Humanities, Department of Informatics
Muitinės g. 8, LT-44280, Kaunas, Lithuania

Audrius.Lopata@khf.vu.lt, Ilona.Veitaitė@khf.vu.lt, Neringa.Zemaitytė@khf.vu.lt

Abstract. The main scope of the research is to analyse Unified Modelling Language (UML) models generation process from Enterprise Model (EM) in Information Systems (IS) development process by using knowledge-based subsystem. The knowledge-based subsystem is proposed as an additional computer aided software engineering (CASE) tool component to avoid IS development process based on empirics. For comprehensible perception there is also presented relation between EM and ontologies and its use in generation process.

As the result of this part of research transformation algorithms are presented and described. These algorithms are capable of whole UML models elements generation from Enterprise Model. Example of UML Interaction Overview model generation illustrates full process.

Keywords: Enterprise Modelling, Knowledge-based, IS engineering, UML, CASE, Ontology, Interaction Overview model.

1 Introduction

In a modern world software development and software applications are becoming more complex and demanding. Developers, analysts, engineers, researchers are creating and seeking for new techniques and procedures to streamline software engineering processes to ensure shorter development time and reduce costs by re-using different components. The development of software systems is a complex activity which may imply the participation of people and machines (distributed or not). Therefore, different
stakeholders, heterogeneity and new software features make software development a heavily knowledge-based process [1, 11].

In a modern day enterprise engineering, it is paramount that Enterprise Models are grounded in a well-defined, agreed-upon Enterprise Architecture that captures the essentials of the business, IT, and its evolution. Enterprise architectures typically contain different views (e.g. Business, Information, Process, Application, Technical) on the enterprise that are developed by distinct stakeholders with a different background and knowledge of the business. Consequently, the developed Enterprise Models that populate these views are hard to integrate. A possible solution for this integration problem is using a shared terminology during the development of these different views [2]. Such explicit formal representations, often materialized in the form of ontology – in a business context called an enterprise-specific ontology – provide a myriad of advantages. Ontologies are shared views of domains. They provide conceptualizations that are agreed upon by participants in collaborative action and decision making. The explicit existence of such shared perspectives makes it possible for both people and programs to collaborate by ensuring that everybody makes the same distinctions and uses the same terms with the same meaning [19]. On an intra-organizational level, they ensure model re-usability, compatibility and interoperability, and form an excellent basis for enterprise-supporting IT tools, such as Enterprise Resource Planning (ERP) systems, business intelligence (BI) tools or information systems (IS), for which they serve as common terminology. On an inter-organizational level, they facilitate interoperability, cooperation and integration by allowing formal mappings between, and alignment of separately developed Enterprise Models [12].

2 Enterprise Modelling and Ontologies relation

An Enterprise Model is a computational representation of the structure, activities, processes, information, resources, people, behaviour, goals and constraints of a business, government, or other enterprise. It can be both descriptive and definitional - spanning what is and what should be. The role of an Enterprise Model is to achieve model-driven enterprise design, analysis and operation [6, 19]. Enterprise Modelling is an activity where an integrated and commonly shared model of an enterprise is created [7, 12, 28]. The resulting Enterprise Model comprises several sub-models, each representing one specific aspect of the enterprise, and each modelled using an appropriate modelling language for the task at hand. For example, the Enterprise Model may contain processes modelled in BPMN, data modelled in ER and goals modelled in n*. The Enterprise Model is thus developed by several enterprise engineers, and aggregates all information about the enterprise. As a result, Enterprise Models without homogenized underlying vocabulary suffer interoperability and integration problems [12, 25]. An Enterprise Model can be developed for single or more different purposes. Few Enterprise Modelling formal purposes are presented [3, 21]:

1. To capitalize enterprise knowledge and know how.
2. To illustrate relations and dependencies within the enterprise and with other enterprises, to achieve better control and management over all aspects.
3. To provide support to business process re-engineering.
4. To get a common and complete understanding of the enterprise.
5. To improve information management across organizational and application system boundaries and provide a common means for communication throughout the organization. Rationalize and secure information flows.
6. To provide operative support for daily work at all levels in the enterprise from top to bottom.
7. To control, co-ordinate and monitor some parts of the enterprise.
8. To provide support for decision making.
9. To provide support the design of new parts of the enterprise.
10. To simulate processes.
Ontology is a discipline rooted in philosophy and formal logic, introduced by the Artificial Intelligence community in the 1980s to describe real world concepts that are independent of specific applications. Over the past two decades, knowledge representation methodologies and technologies have subsequently been used in other branches of computing where there is a need to represent and share contextual knowledge independently of applications [23].

Ontology engineering is a filiation of knowledge engineering that studies the methods and methodologies for building ontologies. In the domain of enterprise architecture, ontology is an outline or a schema used to structure objects, their attributes and relationships in a consistent manner. As in Enterprise Modelling, ontology can be composed of other ontologies. The purpose of ontologies in Enterprise Modelling is to formalize and establish the shared understanding, reuse, assimilation and dissemination of information across all organizations and departments within an enterprise. Also, an ontology enables integration of the various functions and processes which take place in an enterprise [10].

Using ontologies in Enterprise Modelling offers several advantages. Ontologies ensure clarity, consistency, and structure to a model. They promote efficient model definition and analysis. Generic enterprise ontologies allow for reusability and automation of components. A common ontology allows to ensure shared understanding, clearer communication, and more effective coordination among the various divisions of an enterprise. These lead to more efficient production and flexibility within the enterprise [24].

3 Transformation Algorithm

The computerized IS engineering specific methods are developed based on common requirements, which systematize the selected methodology. Computerized knowledge-based IS engineering project management basis is CASE system knowledge-based subsystem. CASE system’s knowledge-based subsystem’s core component is knowledge base, which essential elements are enterprise meta–model specification and Enterprise Model for certain problem domain [4, 8, 25]. Knowledge-based subsystem is one more active participant of IS engineering process beside analyst, whose purpose is to verify results of IS life cycle phases [5, 9].

Knowledge-based CASE systems holding substantial components, which organize knowledge: knowledge-based subsystem’s knowledge base, which essential elements are enterprise meta–model specification and Enterprise Model for certain problem domain [7, 13, 16].
Information system design methods indicate the continuance of systems engineering actions, i.e. how, in what order and what UML models to use in the design process and how to fulfil the process. Association between UML models and Enterprise Model is realized through the transformation algorithms [14, 15].

Enterprise Model as organization’s knowledge repository enables generate UML models with the help of transformation algorithms. Enterprise meta-model holds essential elements of business modelling methodologies and techniques, which ensures a proper UML models generation process [17, 18, 20]. Presently, used CASE system’s Enterprise Models constitution is not verified by formalized criteria. Enterprise Models have been formed in compliance with the notations. However, their composition has not been proved by the characteristics of the specific domain area [27, 28].

In IS engineering all design models are fulfilled on the basis of the empirical expert experience. Experts, who participate in the IS development process, do not gain enough knowledge, and process implementation in requirements analysis and specification phases can take a too long time. Enterprise meta–model contains essential elements of business modelling methodologies and techniques, which insures a suitable UML diagrams generation process [27, 28].
Figure 3 presents top level transformation algorithm for Enterprise meta-model based UML models generating process. Main steps for generating process are identifying and selecting UML model for generating process, identifying starting elements for the selected UML model and selecting all related elements, reflecting Enterprise Model elements to UML model elements and generating the selected UML model [22, 27, 28].

4 UML Interaction Overview Model Transformation

UML Interaction Overview diagram determines interactions through a variant of activity diagrams in a manner that maintains overview of the control flow. Interaction Overview model concentrate on the overview of the flow of control where the nodes are interactions or interaction uses. The lifelines and the messages do not perform at this overview level. UML Interaction Overview model combines elements from activity and interaction diagrams [22]:

- the following elements of the activity diagrams could be used on the Interaction Overview diagrams: initial node, flow final node, activity final node, decision node, merge node, fork node, join node;
- the following elements of the interaction diagrams could be used on the Interaction Overview diagrams: interaction, interaction use, duration constraint, time constraint.
Main steps of UML Interaction Overview model generation from Enterprise Model transformation algorithm are: selecting Interaction Overview model for generating process, identifying initial element, selecting element’s type for chosen model, selecting related model elements and generating model.

Table 1 presents UML Interaction Overview model elements generated from Enterprise Model. Frame as Interaction model element is generated from EM Actor element, Interaction Use as Interaction model element is generated from EM Information Activity, Initial Node, Decision Node, Merge Node, Final Node as Activity model elements are generated from EM Business Rules elements and Decision Guard as Activity model element is generated from EM Information Flow element.

**Table 1.** EM and Online Service Ordering UML Interaction Overview model elements

<table>
<thead>
<tr>
<th>UML Interaction Overview model</th>
<th>EM</th>
<th>Frame</th>
<th>Interaction Use</th>
<th>Initial Node</th>
<th>Decision Node</th>
<th>Merge Node</th>
<th>Final Node</th>
<th>Decision Guard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>Material Input</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Material</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Business Rules</td>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Information</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure (Fig. 5) presents an example of UML Interaction Overview model. The necessary elements through transformation algorithms were received from CASE tool’s knowledge-based subsystem’s Enterprise Model, where all knowledge of subject area is stored. In this figure it is clearly seen all necessary UML Interaction Overview model elements generated from Enterprise Model.

5 Conclusions

Computer aided IS engineering is based on empiric and IS development life cycle stages are fulfilled on the basis of the expert’s experience. A large part of the CASE tools design models are generated only partially, and complete realization is possible only non-automatic and with experts participation. Today IS engineering should be based on knowledge. In this way, knowledge-based IS engineering computerized IS development activities are executed using the subject area knowledge, which is stored in the knowledge base of CASE tool repository.

In order to decrease the influence of empirical factors on IS development process, the decision was made to use knowledge-based IS engineering approach. The main advantage of this approach is the possibility to validate specified data stored in EM against formal criteria, in that way decreasing the possible issues and ensuring more effective IS development process compared to classical IS development methods.

Using ontologies in Enterprise Modelling offers several advantages. Ontologies ensure clarity, consistency, and structure to a model. They promote efficient model definition and analysis. Generic enterprise ontologies allow for reusability of and automation of components. Because ontologies are schemes or outlines, the use of ontologies does not insure proper Enterprise Model definition and analysis. Ontologies are limited by how they are defined and fulfilled. Ontology not always includes ability to cover all of the aspects of what is being modelled.

The paper deals with the generation process of UML models from EM options. Every element of UML model can be generated from the EM using CASE Tool knowledge base’s subsystem and transformation algorithms. Method of UML model generation process from EM could implement full knowledge-based IS development cycle design stage. This is partially established by the example of online service ordering presented as UML Interaction Overview model elements generation.
6 References