Using UML class diagrams for content ontology design patterns engineering

N O Dorodnykh\(^1\), O A Nikolaychuk\(^1\) and A Y Yurin\(^1\)

\(^1\) Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of the Russian Academy of Sciences (ISDCT SB RAS), 134, Lermontov str., Irkutsk, 664033, Russia

E-mail: iskander@icc.ru

Abstract. The paper presents a method for Ontology Design Patterns (ODP) engineering based on the transformation of conceptual models serialized in the XML-like formats. In this paper, we use UML class diagrams as the source of conceptual models. The method includes the following activities: design and serialization of a source conceptual domain model; analysis and transformation of an XML structure of a source conceptual model to an ontological schema model; transformation of an ontological schema model to a content ODP code in the OWL 2 DL format; verification and modification of obtained content ODPs by domain experts. All transformations are implemented with the use of original domain-specific language, namely, Transformation Model Representation Language (TMRL). The proposed method is used for ODPs engineering in tasks of industrial safety inspection of technical systems. Obtained ODPs can be used for knowledge bases and intelligent systems engineering.

1. Introduction

Ontologies [1] remain one of the most powerful means for knowledge representation. Ontology engineering is a rather difficult and time-consuming task. The various approaches and tools are developed to reduce the complexity of this task, and Ontology Design Patterns (ODPs) is one of them. This approach eliminates some frequently repeated errors through the use of the typical solutions represented in the form of patterns. These patterns are collected in catalogs that in most cases are focused on a specific domain, therefore, they cannot have completeness and universality but can be useful for developers as drafts or templates. ODPs can be separated into six types in accordance with their purpose: structural, correspondence (re-engineering and alignment), content, reasoning, presentation (naming and annotating), and lexico-syntactic. Content ODPs are the most popular type of ODPs. Content ODPs encode conceptual design patterns and provide solutions for domain modeling problems. Thus, the use of common ontology fragments in the form of content ODPs simplifies and speeds up further refinement (modification) of the ontologies obtained through the use of standard solutions.

Various specialized editors (e.g., Protégé, ONTOedit, Menthor Editor, etc.) are used for ontologies and ODPs engineering. In most cases, these tools support a direct manipulation approach and don’t integrate with other data and knowledge sources for the automated formation of ODPs. One of these sources is conceptual domain models represented in the forms of flowcharts, event and fault trees, concept and mind maps, UML models, etc. Domain experts prefer to use these models for knowledge
systematization and formalization. The analysis of conceptual modeling tools (including editors: IHMC CmapTools, FreeMind, Coggle, Mindjet MindManager, ProTheBrain, XMind, etc.) showed the lack of ability to the synthesis of source codes and specifications for ontologies or knowledge bases. Most of these tools use XML-like formats that are incompatible with each other. These formats can be used for automatic retrieving information about concepts and relationships, and therefore for ontologies and knowledge bases engineering [2-4].

This paper proposes a method for content ODPs engineering based on automated analysis and transformation of conceptual models, in particular, XML-based UML (Unified Modeling Language) class diagrams [5]. This method is used for ODPs engineering in tasks of industrial safety inspection (ISI) of technical systems, in particular, connected with technical condition assessment and degradation processes modeling.

2. Related works

There are examples of transformations of conceptual models for ontology engineering at the level of schemas (T-Box) and instances (A-Box). Some of them use model transformation languages and UML class diagrams as a source of conceptual models. For example, a metamodel-driven model transformation approach for interchanging rules between OWL along with the Semantic Web Rule Language (SWRL) and OCL (Object Constraint Language) with UML is proposed in [6]. Herewith, the REWERSE Rule Markup Language (R2ML) was used as an intermediate representation of knowledge (rules), and ATL (ATLAS Transformation Language) is used to describe the transformation rules. In [7] researchers present an approach to transforming UML class diagrams into OWL ontologies using the QVT (Query/View/Transformation) standard. In [8], an approach for transforming UML class diagrams into OWL ontology based on graph transformations and using AToM3 tool is proposed. An integrated method is proposed in [9], it provides transformation of domain models in the Object Role Modeling (ORM) format into OWL ontologies. eXtensible Stylesheet Language Transformations (XSLT) is used in the first stage of converting conceptual models from the XML format to the ORM model. At the second stage, a specially developed ORM2OWL tool is used to convert the ORM model into OWL ontology. Other successful examples of ontology engineering are based on the transformation of conceptual UML models presented in [10, 11].

There are examples of transforming different conceptual models, for example, Entity-Relationship schemas [12], and SBVR statements [13] into OWL ontologies.

Most of the considered works are “ad-hoc” solutions which suitable for the specific case studies only. The main disadvantages of the mentioned studies are the followings:

- A focus on programmers, but not domain experts.
- A rigid algorithmic definition of correspondences between the elements of the source conceptual model and the target ontology. As a result, there is no possibility to change the interpretation of the mappings of elements without making changes in the program source codes of corresponding software.
- A rather complicated implementation that, in turn, causes a variety of software used by researchers within each transformation (sometimes a separate specialized tool is used at each transformation stage).

So, the problem of developing more reliable and efficient means for ontology and ODPs engineering remains urgent. In this paper, we propose a method for the automated development of ontological schemes in the form of content ODPs to address these disadvantages. Our method is based on the transformation of UML class diagrams using an author’s domain-specific language and software for describing transformation models.

3. Proposed method

Our method for ODPs engineering is based on the transformation of conceptual models serialized in the XML-like formats. This transformation can be formally described as follows: $T: CM^{XML} \rightarrow ODP^{OWL}$
when \( CM^{XML} \) is a source conceptual model in the XML-like format; \( ODP^{OWL} \) is OWL 2 DL ODP code, in our case \( CM^{XML} = CM^{UML,XML} \), \( CM^{UML,XML} \) is a conceptual model in the form of a UML class diagram serialized with the use of the XML Metadata Interchange (XMI) format.

\( CM^{UML,XML} \) can be represented as follows: \( CM^{UML,XML} = <C, DT, RL> \) where \( C \) is a set of classes; \( DT \) is a set of datatypes; \( RL \) is a set of relationships. \( C = \{c_1 \ldots c_n\}, c_i = <c\_name_i, AT_i>, i = \{1 \ldots n\}, c\_name_i \) is a class name; \( AT_i \) is a set of class attributes, \( AT_i = \{a_{i,1} \ldots a_{i,k}\}, a_{i,j} = <a\_name_j, type_j, value_j> \), \( a\_name_j \) is a attribute name; \( type_j \) is a attribute datatype, \( value_j \) is a possible attribute value.

\( RL = \{rl_1 \ldots rl_d\}, rl_i = <rl\_name_i, rl\_type_i, lhs_i, rhs_i> \), when \( rl\_type_i \) is a relationship type (inheritance, dependency, association, aggregation, composition, realization); \( rl\_name_i \) is a relationship name; \( lhs_i \) is a left side of a relationship, \( rhs_i = <name_{lhs_i} cd_{lhs_i} c_{j,i}> \), when \( name_{lhs_i} \) is a name of a class role at the left relationship side, \( cd_{lhs_i} \) is a cardinality of the left relationship side, \( c_{j,i} \) is a class at the left relationship side, \( rhs_i \) is a right side of a relationship, \( rhs_j = <name_{rhs_j} cd_{rhs_j} c_{k,j}> \), when \( name_{rhs_j} \) is a name of a class role at the right relationship side, \( cd_{rhs_j} \) is a cardinality of the right relationship side, \( c_{k,j} \) is a class at the right relationship side; \( T = \{T_{CM \rightarrow OSM}, T_{OSM \rightarrow ODP} \} \) when \( T_{CM \rightarrow OSM} \) is a set of rules for transforming \( CM^{XML} \) to \( OSM \); \( T_{OSM \rightarrow ODP} \) is a set of rules for transforming \( OSM \) to \( ODP^{OWL} \). \( OSM \) is an ontological schema model at the T-Box level. This model is used for unified representation and storage of knowledge extracted from conceptual models; it abstracts from features of various knowledge representation languages and some dialects used on the serialization of ontologies (e.g., OWL, RDFS, etc.).

\( OSM = <C, OP, DP, TP, DTS> \), when \( C \) is a set of classes; \( OP \) is a set of object properties; \( DP \) is a set of datatype properties; \( TP \) is a set of transitive properties; \( DTS \) is a set of XML Schema datatypes.

The proposed method for Content ODPs engineering based on transforming conceptual models includes four main activities:

- **Design and serialization of a source conceptual model:** a domain expert represents knowledge in the form of a visual schema using domain-specific notations; the resultant conceptual model is serialized in the XML-like format, for example, XMI for further transformation. These activities can be implemented using various software (editors) for example, StarUML or IBM Rational Rose Enterprise.

- **Analysis and transformation of an XML structure:** a source conceptual model to an ontological schema model (it is the implementation of a \( T_{CM \rightarrow OSM} \) operator). This activity involves extracting elements, their attributes, and relationships from an XML tree and also forming an ontological schema model (\( OSM \)). The main results of this activity are ontological fragments in the form of a set of classes and their relationships (including objects and datatype properties).

- **A model transformation is one of the major concepts in Model-Driven Engineering (MDE) [14]. In our case, we use a declarative domain-specific language – Transformation Model Representation Language (TMLR) [15] to describe a set of transformation rules (\( T_{CM \rightarrow OSM} \)), and special software – Knowledge Base Development System (KBDS) to support this transformation. The transformation is a set of rules for mapping XMI elements of a conceptual model to ontological constructs. Each rule contains correspondence between elements of the source and target metamodels. An XML Schema is used as a source metamodel for the XML-based conceptual model. TMLR defines three types of elements matching: equivalence (one-to-one matching), synonymy (ambiguous matching), and homonymy (indistinguishable matching).

- **Transformation of an ontological schema model:** to content ODP code in the OWL 2 DL format (it is the implementation of a \( T_{OSM \rightarrow ODP} \) operator). TMLR is also used to describe a scenario of model transformation. The description of OWL 2 DL specification is used as the target metamodel.

  The main correspondences between the elements of a source XMI UML class diagram, an ontological schema model, and target OWL ODP code are shown in Table 1.

- **Verification and modification of resulted content ODPs:** by domain experts. This stage is optional. The content ODP code can be refined using various ontological modeling editors, for example, Protégé and others.
The proposed method is implemented in the form of software modules (converters) for KBDS. KBDS is research software that provides the use of conceptual model transformations and TMRL to generate ontologies and rule-based knowledge bases.

### Table 1. Main correspondences between elements of XMI UML class diagrams, OSM, and OWL ODPs.

| XTI UML Class diagram | OSM                 | OWL ODP            |
|------------------------|---------------------|--------------------|
| UML:Model              | Ontology            | owl:Ontology       |
| UML:Class              | Class               | owl:Class          |
| UML:Class (generalization) | Class (superclass) | rdfs:subClassOf    |
| UML:Class (name)       | Class (name)        | rdf:about          |
| UML:Association        | Relationship        | owl:ObjectProperty |
| UML:AssociationEnd (type) | Relationship (lhs) | owl:ObjectProperty / rdfs:domain |
|                        |                     |                    |
|                         |                     |                    |
| UML:Attribute          | Property            | owl:DatatypeProperty |
| UML:Attribute / UML:Classifier.feature / | Property (class) | owl:DatatypeProperty / rdfs:domain |
| UML:Class              |                     |                    |
| UML:Attribute          | Property (value)    | owl:DatatypeProperty / rdfs:range |

### 4. Case Study

Let’s illustrate our method in the case of content ODPs engineering for tasks of industrial safety inspection (ISI) of petrochemical systems.

An ISI procedure consists of confirming the compliance of a technical object state with the requirements of safety in a particular industry. The specific content of ISI tasks depends on the organization conducting the ISI. In this case study, we use the experience of Irkutsk Research and Design Institute of Chemical and Petrochemical Engineering (IrkutskNIIhimmash).

Analysis of ISI tasks showed that the implementation of a part of them (development of the ISI program, analysis of diagnostics results including interpretation, etc.) requires the processing of a large volume of weakly formalized information and actual domain models, in particular, in the form of ontologies. We simplified ontology engineering by semi-automatic creation of content ODPs on the basis of the transformation of conceptual domain models and spreadsheets from ISI reports.

**Design and serialization of a source conceptual model.** ISI-models dataset¹ is used as the source of domain models. This dataset contains 26 conceptual models designed in IBM Rational Rose and IHMC CmapTools, and describing the basic 300 entities and 209 relationships in the field of ISI. This dataset was used for the automated formation of content ODPs. Figure 1 shows a fragment of the UML class diagram with a description of a technical object.

**Analysis and transformation of an XML structure.** We obtain an ontological schema model (OSM) on the basis of serialized UML class diagrams form the ISI models dataset with the use of the KBDS module (converter). Mappings between source and target elements of metamodel were described by TMRL in the accordance with Table 1.

**Transformation of an ontological schema model.** TMRL is also used to describe a scenario of model transformation in the accordance with Table 1. The description of OWL 2 DL specification is used as the target metamodel. The obtained content ODP fragment is presented in Figure 2.

Concepts and relationships from fragments of a class diagram (Figure 1) were mapped into a set of

---

¹ http://dx.doi.org/10.17632/f9h2t766tk.1
ontological classes and their properties. In the case study 24 classes, 18 object properties, and 48 data properties were obtained. In general, the obtained ontology scheme is presented at the terminological level (T-Box).

**Figure 1.** A fragment of the UML class diagram from the ISI models dataset.

**Figure 2.** The obtained content ODP fragment with a description of a technical object.
Verification and modification of resulted content ODPs. At the moment, the resulting content ODPs for ISI tasks are being verified by domain experts. These ODPs were refined and verified in Protégé editor.

5. Conclusion
This paper presents a method for ODPs engineering based on the analysis and transformation of conceptual domain models. Our method involves the use of a specialized domain-specific language (TMRL) and software (KBDS). The method is tested in the case of ontology engineering for technical systems in the context of ISI. The main result is a set of ontology schemas in the form of content ODPs, which are used as prototypes (drafts) of domain ontologies for the intelligent decision support system [16]. This case study and experiments show the applicability of this method for the creation of content ODPs at the terminological (T-Box) level. In the future, we plan to extend our method and to create content ODPs with instances and their concrete values based on the transformation of spreadsheets.

6. Acknowledgments
The reported study was supported by RFBR (research project No. 19-07-00927) and the Council for Grants of the President of Russia (grant No. MK-1647.2020.9).

References
[1] Guarino N 1998 First International Conference on Formal Ontology in Information Systems (FOIS’98) 46 3-15
[2] Iqbal R, Azrifah M, Murad A, Sliman L and da Silva C P 2018 Measurement 118 73-82
[3] Herrero-Zazo M, Segura-Bedmar I and Martínez P 2016 Knowledge-Based Systems 114 99-107
[4] Starr R R and de Oliveira P J M 2013 Information Systems 38 771-783
[5] Fowler M 2003 Addison-Wesley
[6] Milanović M, Gašević D, Giurca A, Wagner G and Devedžić V 2009 Software: Practice and Experience 39(16) 1313-1346
[7] Zedlitz J and Luttenberger N 2014 Int. J. Advances in Software 7(1) 182-196.
[8] Belghiat A and Bourahla M 2012 Int. J. Computer Applications 41(3) 41-48.
[9] Alberts R and Franconi E 2012 OWL: Experiences and Directions Workshop (OWLED) 849 1-12
[10] Xu Z, Ni Y, He W, Lin L and Yan Q 2012 World Wide Web 15(5) 517-545
[11] Parreiras F S and Staab S 2010 Data & Knowledge Engineering 69(11) 1194-1207
[12] Myroshnichenko I and Murphy M C 2009 IEEE International Conference on Semantic Computing 324-329
[13] Reynares E, Caliusco M L and Galli M R 2015 Expert Systems with Applications 42(5) 2680-2690
[14] Da Silva A R 2015 Computer Languages, Systems & Structures 43 139-155
[15] Dorodnykh N O and Yu rin A Y 2018 CEUR Workshop Proceedings of Information Technologies: Algorithms, Models, Systems (ITAMS’18) 2221 70-75
[16] Yu rin A Y, Berman A F, Nikolaychuk O A and Dorodnykh N O 2019 Studies in Systems, Decision and Control 199 112-124