УДК 502:004.45(075.8)

N. Sydorov, I. Mendzebrovsky, N. Sydorova

ONTOLOGIES IN SOFTWARE ENGINEERING

Software engineering is a separate scientific and practical area with its own structure, terminology, processes, and resources. The software product is a knowledge-based product, and it is the result of knowledge-based actions.

Purpose: *This research presents utilizing ontologies in software engineering. The focus with using ontologies is to represent the world of software development in the form of domains. The ontologies are involved in representing knowledge of three types of the software engineering domains. In the first one, the application domain, the focus is on understanding the customer needs and what the software product must do. In the second one, the implementation domain, the focus is on understanding how the software product must behave and respond to the customer needs. In the third one, the problem domain, the attention is paid to understanding their own software development problems that may occur when creating and maintaining a software product.*

The research goal *is, on the base of domain types, to develop categorization of the software engineering ontologies for supporting software processes.*

Methods: *ontological representation of software engineering knowledge; domain analysis; object-oriented programming; ontology-driven utilizing of programming styles.*

Results: *categorization of the software engineering ontologies and software development approaches. The paper presents the results of the case study, using ontologies by categorization.*

Discussion: *by means of development of categorization of ontologies it is possible to exactly define the types of software engineering ontologies and its place into software processes. This is demonstrated on the examples of the case studies.*

Keywords: ontology, ontology-based development, domain, domain analysis, programming style, coding standard, programming, software engineering.

Introduction

Software engineering is a separate scientific and practical area with its own structure, terminology, processes, and resources. The software product is a knowledge-based product which is the result of the knowledge-based actions. Therefore, the knowledge is the main component of software engineering, and representing, proceeding, and using different knowledge play a great role in software engineering. There are three types of domains in the software engineering: the application domain, the problem domain, and the implementation domain. The knowledge from these domains is used in software engineering during the software product life cycle. Nowadays the ontologies are the best means for representation and proceeding of the software engineering knowledge.

1. Analysis of the latest research and publications

Ontology is a model of a part of the world, which is known in software engineering as a domain [12]. Typically, the model is represented by a set of objects, properties that are associated with objects, relations between objects, and regulations that describe management. Nowadays ontologies are widely used in software engineering for two reasons. Firstly, the ontology is a means of representing the knowledge that is used both in the development and maintenance processes of the software, as well as in its use [11]. Secondly, one can automate the use of knowledge in software by representing the ontology formally, with the help of languages or descriptive logic [4; 6]. In software engineering, the application of ontologies was first classified in 10 directions, in "Software Engineering Body Knowledge". Understanding the role of ontologies in the context of software engineering, development environments and technologies, as well as cases of specific application are given in [11]. The UML extension and its application for graphic representation of ontologies in software engineering are offered in [16]. In view of only one type of the software engineering domain (the application domain) and the temporal dimension (the development time and the run time [8]), one categorization of ontologies in software engineering was developed and established by utilizing the ontologies in software lifecycle processes [8]. In this research, two additional types

of the software engineering domains (the implementation domain and the problem domain) are added, and the additional temporal dimension – pre-development time – is introduced.

2. Purpose and objectives of the research

This research presents utilizing ontologies in software engineering. The focus in utilizing ontologies is made on the representation of the software engineering world in the form of domains. The ontologies are involved in representing knowledge of three types of domains. In the first one, the application domain, the focus is on understanding the customer needs and what the software product must do. In the second one, the implementation domain, the focus is on understanding how the software product must behave and respond to the customer needs. In the third one, the problem domain, the attention is paid to understanding their own software development problems that may occur when creating and maintaining a software product during the software product life cycle. The research hypothesis is that the domain view can assist in understanding the role of ontology in the software engineering. The research goal is to present utilizing ontologies in software engineering in the whole and on the examples of case studies of the authors.

3. Ontologies in the software engineering. Categorization of ontologies and software development approaches

The categorization of ontologies is introduced on the base of two categories [8]: a domain and a software process. In this research its categories are also used. But our categorization is built on the connection terms domain and the software process, as in [5]: the essence of the software process is the progression from the identification of the need in some application domain to the creation of a software product in the implementation domain that responds to that need. Thus, the software process involves two domains: the application domain where the task is to be solved, and the implementation domain where a software-based solution to that task is to be executed. In this research, the third domain is used. It is termed the problem domain, where the software engineering problems are to be solved. For example, the new method or (and) technology is (are) the need for solving tasks from application or (and) implementation domains. Considering the added temporal dimension [7] and pre-development time dimension, in this research the three temporal dimensions are regarded: pre-development time, development time, and run time. The main actions during the pre-development time are the actions of the domain analysis [9]. For the implementation domain and the application domain, these actions are fulfilled for the legacy software products. In view of approaches of using ontologies in the software engineering [8] and processes of the software life cycle [12], the following categorization of ontologies was proposed in this research. The software engineering ontologies are divided into the software engineering domain ontologies (application ontologies, implementation ontologies, problem ontologies) and the software engineering processes ontologies (pre-development time ontologies, development time ontologies, run time ontologies). The software engineering domain ontologies that are created during the pre-development time are called pre-development-time ontologies and consist of reusable components, and they are used in the development-time and run-time software engineering processes of the software life cycle.

4. Case study

This part of the article presents the results of a case study, using ontologies by the introduced categorization. In section 4.1, the examples of pre-development time ontologies are presented. Section 4.2 presents the examples of run time ontologies for the developer and user.

4.1. Domain analysis ontologies

Software reuse can be improved by identifying objects and operations for a class of similar software products, i.e., for a certain domain. In the context of software engineering, domains are application, implementation and problem areas. Airline reservation (application domain), education software tools (implementation domain), green software problems (problem domain) are examples of such domains. The domain engineer captures and organizes this information in a set of domain models with the end of making it reusable when creating a new software product. For the formal representation of the results of the domain analysis, ontologies can be utilized. In the case study, the domain model is a description of objects, properties, and relations in the domain, and it consists of the following [3]: the domain language, the competencies and skills repository, a software engineering education template. The main problem of the domain analysis is creating a set of tools for automatic utilizing of the concrete domain analysis method [9]. The method of domain analysis depends on the domain characteristics and the domain analysis goals. Paper [9] proposes an approach for automation creating domain analysis tools on the base of the MS Office

platform. Provision of domain analysis by using the developed tools is considered on the example of an educational application domain for specialty "Software engineering". The competencies of a specialist are considered as reusable components. The application domain includes, but is not limited to, existing knowledge recommendations in the field under consideration [1; 2], the existing education system, and the legislation. The result of the domain analysis is a list of competencies and disciplines, as well as a reusable template for the "Software engineering" education standard in Ukraine.

4.2. Ontology-driven using of programming styles

Activities of a programmer will be more effective, and the software will be more understandable if within the process of software development, the programming styles (standards) are used, providing clarity of software texts. Programming stylistics problems arose before the structured programming period, but nowadays they remain relevant [13]. Paper [15] proposes a new method of programming styles application based on the ontology. To apply the style, a programmer should solve two tasks: study the description of the style and use and control the style during the coding stage. Thus, it requires two tools: one for studying the style and the other one to control the use of this style. Both tools are based on the presentation of the style. That is why the form of this presentation affects the efficiency of processes performed by a programmer and the efficiency of the tools. It is proposed to use the ontology as a form of knowledge representation about a programming style [14]. Using an appropriate tool (e.g. Protégé [10]), a formal representation of programming style – an ontology is developed. A programmer uses the ontology for coding. Therefore, two tools are required: one for creating an ontology and assisting the programmer, the second one to control the implementation of the style during the coding stage. For these tools two categories of ontologies are needed. The first one, the run time ontology for the ontology-enabled architecture is the result of the ontology-driven pre-development and from this position, it is the application domain ontology. The second one, the run time ontology for the ontology-based architecture is the result of combined utilizing both the ontology-driven pre-development and the ontology-based architecture, and from this position, it is the application domain ontology. The first tool creates the run time ontology template (a reusable asset), which is defining general programming standards properties. A style analyst uses Protégé setup template on a particular programming standard. Then the programmer uses an ontology in the run time to study the programming standard (the ontology-enabled architecture). The second tool is the reasoner [6; 14]. In terms of descriptive logic, the reasoner solves one major problem: verifies the consistency of the ontology [4]. This problem has certain features for the task of programming style implementation [14].

Conclusion

This research proposes categorization of the software engineering ontologies for supporting software processes. Solutions on categorization scheme are presented. Implementation details of categorization are given on the examples of case studies of the domain analysis and naming the styles for Java convention.

References

1. Бондаренко М. Модель випускника бакалаврату «Програмна інженерія» / М. Бондаренко, М. Сидоров, Т. Морозова, І. Мендзебровський // Вища школа. – 2009. – № 4. – С. 50–61.
2. Сидорова Н. М. Формування готовності бакалаврів з інженерії програмного забезпечення до професійної комунікації / Н. М. Сидорова // Вісник НАУ. – 2012. – № 3. – С. 94–100.
3. Сидоров М. О. Доменний аналіз – шлях доказової побудови галузевих освітніх стандартів / М. О. Сидоров, І. Б. Мендзебровський, І. В. Малін // Наукоємні технології. – Київ : НАУ, 2009. – Т. 4, № 4. – С. 59–63.
4. Baader F. The Description Logic Handbook: Theory, implementation, and applications / F. Baader, D. Calvanese, D. McGuinness [et al.]. – Cambridge University Press, 2003. – 320 p.
5. Blum B. I. A taxonomy of Software Development Methods / B. I. Blum // Communication of the ACM. – 1994. – Vol. 37, no. 11. – P. 82–94.
6. Dentler K. Comparison of Reasoners for large Ontologies in the OWL 2 EL Profile [Electronic resource] / K. Dentler, R. Cornet, A. Teije, N. Keizer. – Mode of access: http://www.semantic-web-journal.net/sites/default/files/swj120_2.pdf. – Title from the screen.
7. Guarino N. Formal ontology in information systems / N. Guarino // Proceedings of FOIS'98, Trento, Italy, 6–8 June 1998. – Amsterdam : IOS Press, 1998. – P. 3–15.
8. Happel H. Applications of ontologies in software engineering / H. Happel, S. Seedorf // Proceedings of 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006). – Athens, GA, U.S.A., 2006. – P. 1–14.
9. Medzebrovskiy I. B. Domain analysis tool / I. B. Medzebrovskiy // Матеріали Міжнародної конференції «Інженерія програмного забезпечення 2017». – Київ : НАУ, 2017. – P. 30.
10. Protégé [Електронний ресурс]. – Режим доступу: http://protege.stanford.edu. – Назва з екрана.
11. Ruiz F. Chapter 2. Using Ontologies in Software Engineering and Technology / F. Ruiz, J. Hilera // Ontologies for Software Engineering and Software Technology / Coral Calero, Francisco Ruiz, Mario Piattini. – Berlin, Heidelberg : Springer, 2006. – P. 62–102.

12. Sidorov M. O. Software engineering / M. O. Sidorov. – Kyiv : NAU, 2007. – 135 p.
13. Sidorov N. A. Software stylistics / N. A. Sidorov // Proc. of the National Aviation University. – 2005. – No. 2. – P. 98–103. – doi: 10.18372/2306-1472.24.1152.
14. Sidorov N. A. Ontology-driven tool for utilizing programming styles / N. A. Sidorov, N. N. Sidorova, A. I. Pirog // Proc. of the National Aviation University. – 2017. – No. 2. – P. 98–103. – doi: 10.18372/2306-1472.24.1152.
15. Sidorova N. N. Ontology-driven method using programming styles / N. N. Sidorova // Software engineering. – 2015. – No. 2. – P. 19–29.
16. Wongthongtham P. Development of a Software Engineering Ontology for Multi-site Software Development / P. Wongthongtham, E. Chang, T. Dillon, I. Sommerville // IEEE Transactions on knowledge and data engineering. – 2009. – Vol. 21 (8). – P. 1205–1217.

Сидоров М. О., Сидорова Н. М., Мендзебровський І. Б.

ОНТОЛОГІЇ В ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення – це окрема науково-практична сфера з власною структурою, термінологією, процесами та ресурсами. Програмний продукт є продуктом, що ґрунтується на знаннях, і це результат дій, базованих на знаннях.

Призначення: *у цій роботі представлено використання онтологій в інженерії програмного забезпечення. Фокус з використанням онтологій робиться на представленні світу інженерії програмного забезпечення у вигляді доменів. Онтології беруть участь у представленні знань трьох типів доменів. У першому, домені застосунків, основна увага приділяється розумінню потреб клієнта та того, що має виконувати програмний продукт. У другому, домені реалізації, основну увагу зосереджено на розумінні того, як програмний продукт повинен поводитися і реагувати на потреби клієнтів. У третьому, проблемному, домені основна увага приділяється розумінню власних проблем інженерії програмного забезпечення, які можуть виникнути при створенні та супроводженні програмного продукту.*

Мета дослідження *– розробка класифікації онтологій інженерії програмного забезпечення для підтримки програмних процесів.*

Методи: *онтологічне подання знань з програмного забезпечення; доменний аналіз; об'єктно-орієнтоване програмування; онтологічне використання стилів програмування.*

Результати: *категоризація онтологій та підходів інженерії програмного забезпечення. Представлено результати тематичних досліджень з використанням онтологій шляхом категоризації.*

Обговорення: *застосування типів доменів для класифікації онтологій інженерії програмного забезпечення, застосування демонструється на прикладах архітектури інструментарію доменного аналізу та засобів для контролю правил іменування стандарту Java.*

Ключові слова: онтологія, розробка, базована на онтології, домен, доменний аналіз, стиль програмування, стандарт кодування, програмування, інженерія програмного забезпечення.

Матеріал надійшов 28.11.2017