UDC 502:004.45 (075.8)

N.A. Sydorov, N.N. Sydorova, I.B. Mendzebrovsky

SOFTWARE ENGINEERING ONTOLOGIES CATEGORIZATION

Software engineering is an own scientific and practice aria with an own structure, terminology, products, processes and resources. The software product is a knowledge-based product and it is the result of the knowledge-based actions. In this research, categorization of ontologies in software engineering are presented. The known criterion (process, domain, structure) are used for the categorization of ontologies but two criterion (process and domain) was modified. The process criterions is looked in the connection whit time period and the predevelopment time added to time period. The domain criterions is making whit help of the representation of software engineering world in the form of domains. The ontologies are involved in representing knowledge of three types of the software engineering domains. In the first, the application domain, the focus is on understanding the customer needs and what the software product must do. In the second, the implementation domain, the focus is on understanding how the software product must behave and respond to the customer needs. In the third, the problem domain, the focus is on understanding the software engineering problems, which can be during software life cycle processes of the software product. Our research goal is to develop categorization of the software engineering ontologies on the base of adding known criterion. Ontological representation of software engineering knowledge; categorization; domain analysis; object-oriented programming; ontology-driven utilizing of programming styles. Categorization of the software engineering ontologies. The results of case study, using ontologies by categorization are presented. Had developed categorization of ontologies, it is possible exactly to define types of software engineering ontologies and its places into software processes. This is demonstrating on the examples of the case studies.

Key words: software engineering, programming, ontology, categorization, domain analysis, programming style.

1. Introduction

The software engineering is an own scientific and practice aria with an own structure, terminology, products, processes and resources. The software product is a knowledge-based product and it is the result of the knowledge-based actions. Therefore, the knowledge is main component of software engineering, and representing, proceeding and using of differently knowledge play 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 are used in the software engineering during the software processes of the software product life cycle. Nowadays the ontologies are the best means for representation and proceeding of the software engineering knowledge.

2. Analysis of latest research and publications

Ontology is a model of the world part, which is known in software engineering as a domain (Sidorov, 2007). 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, an 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 utilizing (Ruiz et. al. 2006). Secondly, one can automate the utilizing of the knowledge in software by representing an ontology formally, with the help of languages or descriptive logic (Dentler 2011, Baader, Calvanese,&Guinness, 2003). 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 (Ruiz et. al. 2006). The UML extension and its application for graphic representation of ontologies in software engineering are given in (Wongthongtham, et. al. 2009).

In view of only one type software engineering domain (the application domain) and two temporal dimensions (development time and run time (Guarino 1998)), the one categorization of ontologies in software engineering was developed and established on the utilizing of ontologies in software life cycle processes (Happel, Seedorf 2006). The logically categorization of software engineering ontologies presented on this results in (Dentler 2011, Baader, Calvanese,& Guinness, 2003). But the categorization is bulky and inaccurate as it uses only one type of the software engineering domains and two temporal dimensions. In our research, two types of the software engineering domains (implementation domain, problem domain) are added, and additional the temporal dimension – pre-development time is introducing.

3. Purpose and objectives of the research

In this research, the ontologies categorization in software engineering are presented. The first focus categorizing ontologies is making on the representation of software engineering world in the form of domains. The ontologies are involved in representing knowledge of three types of domains. Firstly, the application domain, the focus is on understanding the customer needs and what the software product must do. Secondly, the implementation domain, the focus is on understanding how the software product must behave and respond to the customer needs. Thirdly, the problem domain, the focus is on understanding the of software engineering problems, that can be during the software life cycle. The second focus categorizing ontologies is making on the software life cycle time periods. The software life cycle includes the three time periods – predevelopment, development and run. The research hypothesis is as the domain view can assistance in understanding the role of ontology in the software engineering. The research goal is to present utilizing ontologies in software engineering in whole and on the examples of the case studies of authors.

4. Categorization of ontologies

The categorization of ontologies was introduced on the base of two categories (Happel, Seedorf 2006): a domain and software process time period. In our research also, its categories are using. But our categorization is built on the connection terms time period, domain, and software process as in (Blum 1994): the essence of the software process is the progression from identification of the need in some application domain to the creation of a software product in implementation domain that responds to that need. Thus, the software process involves two domains: the application domain, where a task is to be solved, and the implementation domain, where software-based solution to that task is to be executed (application software is created). In our research, the third domain is using. It is called the problem domain, where the software engineering problems are to be solved. For example, the new method or (and) technology is (are) need for solving of tasks from application or (and) implementation domains. Considering the temporal dimension (Guarino 1998) and pre-development time dimension added, in our research, three the temporal dimensions are looking – pre-development time, development time, and run time. The main actions during pre-development time are actions of domain analysis (Prieto-Diaz 1990, Mendzebrovskiy 2017). For the implementation domain and the application domain, these actions are fulfilled on the legacy software products. To finish the categorization of ontologies will use the structure dimension category (Guarino 1998), when the ontology can be used as part of software environment or part of software product (software artifact/information resource). In view of approaches of using ontologies in the software engineering (Happel, Seedorf 2006) and processes of the software life cycle (Sidorov, 2007), the following categorization of ontologies was proposed in this research (Fig. 1).

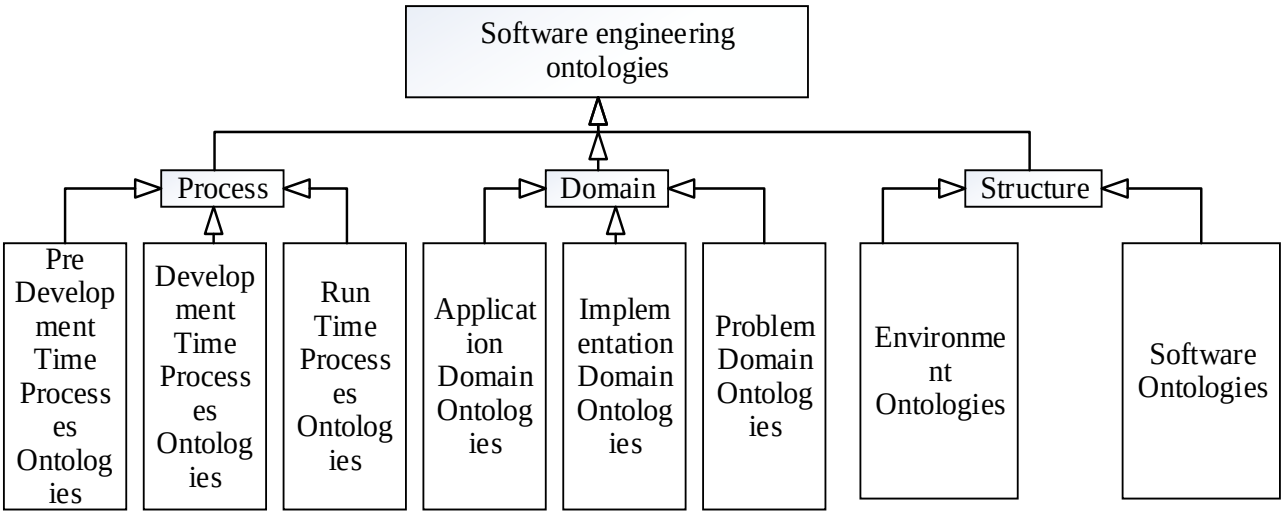

Figure 1. The categorization of the software engineering ontologies

The proposed categorization uses of three categories – the software process (it is in time periods), the software engineering domain, and the software structure (Fig. 1). Thus, in that research, the software engineering ontologies are divided on the software engineering processes ontologies (pre-development time processes ontologies, development time processes ontologies, run time processes ontologies), on the software engineering domain ontologies (application domain ontologies, implementation domain ontologies, problem domain ontologies), and software structure ontologies (environment ontologies, software product ontologies). The software engineering domains ontologies can be created during pre-development time, they are called pre-development time ontologies, consist of reusable components and they can be used into development-time and run time software engineering processes of the software life cycle. The software development approaches was described in (Happel, Seedorf 2006). In (Sydorov, Mendzebrovsky&Sydorova 2017), the ontology-driven pre-development approach is introducing. Ontology-driven pre-development subsumes the usage of ontologies at pre-development time (during domain analysis) that describe the software engineering domains.

5. Case study

In this part of the article will present the results of case study, using ontologies by introduced categorization. Into 4.1 section, the examples of pre-development time processes ontologies are presented. Into 4.2 section, the examples of run time processes ontologies for developer and user are presented.

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. Examples of domains are airline reservation (application domain), the airline reservation software system (implementation domain), and green software problems of the airline reservation software system (problem domain). The scope of a domain can be chosen arbitrarily, either broad, e.g., banking, or as narrow as simple text editing. Usually broad domains are built on top of several narrow domains. Domain analysis is the activity that discovers and formally describes the commonalities and variability within a domain (Sydorov, Mendzebroskiy&Malin 2009). The domain engineer captures and organizes this information in a set of domain models with the end of making it reusable when creating new software product. The output of domain analysis is a domain model: an explicit representation of knowledge about the domain. For the formal representations of the domain analysis results can be utilized ontologies. In case study domain model is a description of objects, properties and relations in domain and consists of the following (Bondarenko. et. al. 2009): domain language, competencies and skills repository, software

engineering education template. The main problem of domain analysis is creating the set of tools for automation utilizing of the concrete domain analysis method (Mendzebrovskiy 2017). The method of domain analysis is depending from domain characteristics and domain analysis goals. In (Mendzebrovskiy 2017) was proposed approach for automation creating domain analysis tools on the base of the MS Office platform. In context of approach two methods was proposed. The first method is called “in small”, when the separate process of domain analysis is automated with help of MS Office tool, for example, MS Visio is using for representing ontologies diagrams whit the help of UML (Wongthongtham et. al. 2009). The second method is called “in large”, when the all processes of domain analysis are automated with help of all tools of the MS Office. Provision of domain analysis using the developed tools is considered on the example of educational application domain for the specialty "Software engineering". The competences 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 (Bondarenko. et. al. 2009, Sydorova 2012), existing education system, and the legislation. The result of domain analysis is a list of competencies and disciplines, as well as a reusable template for the "Software engineering" education standard in Ukraine. Considering the activity of a specialist (bachelor) and the domain view in the context of software engineering, a general ontology is considered in three aspects (Fig. 2). The bachelor (domain expert) learns the task from an application domain and creates the pre-development time processes ontologies for software product.

He (her), having studied the application domain and interacting with the customer, implements software processes that are aimed at the development of a software product in the implementation domain. The software product will be used in the application domain. If a bachelor has problems related to the implementation of software processes, he (her) solves them within the problem domain. A bachelor implements software processes, creating a software product for the application domain. That is why, he should have knowledge of the application domain (to be the application domain expert) (Fig. 3), and interacting with the subjects (customers) of the application domain. That he should have the appropriate communication knowledge and skills (Fig. 4).

The pre-development time processes ontologies (Fig. 2, 3, 4) can be used during the development time or (and) the run time periods.

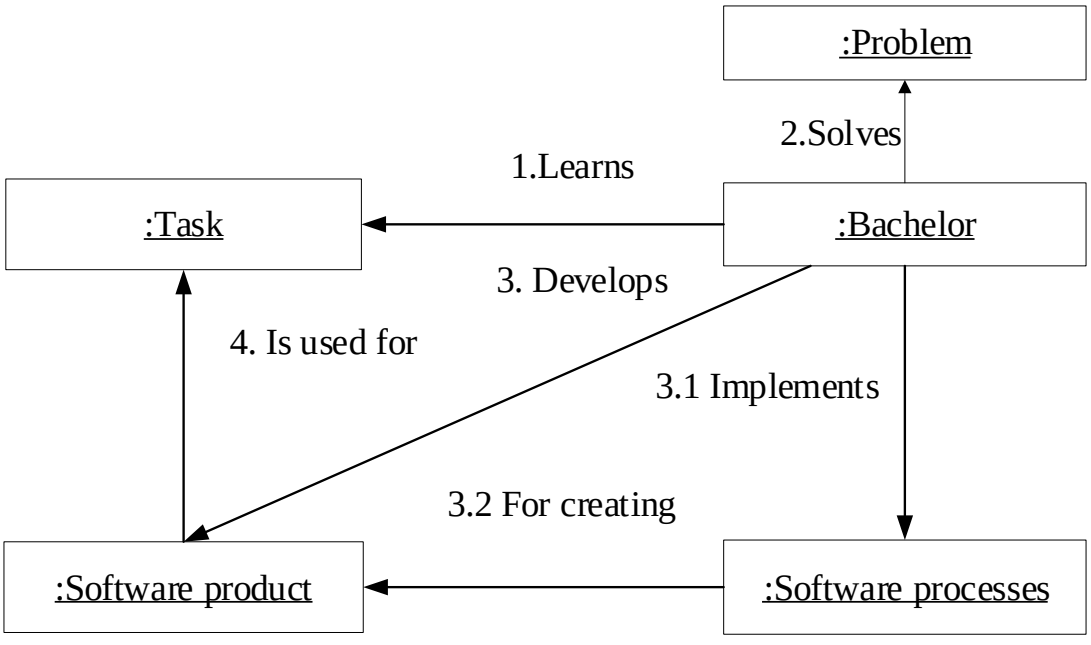

Figure 2. General ontology

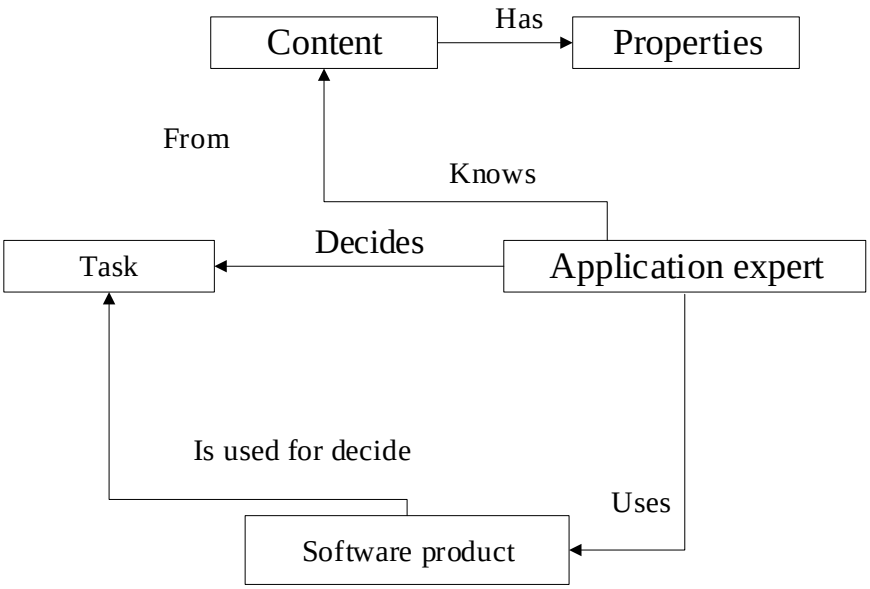

Figure 3. Application domain ontology

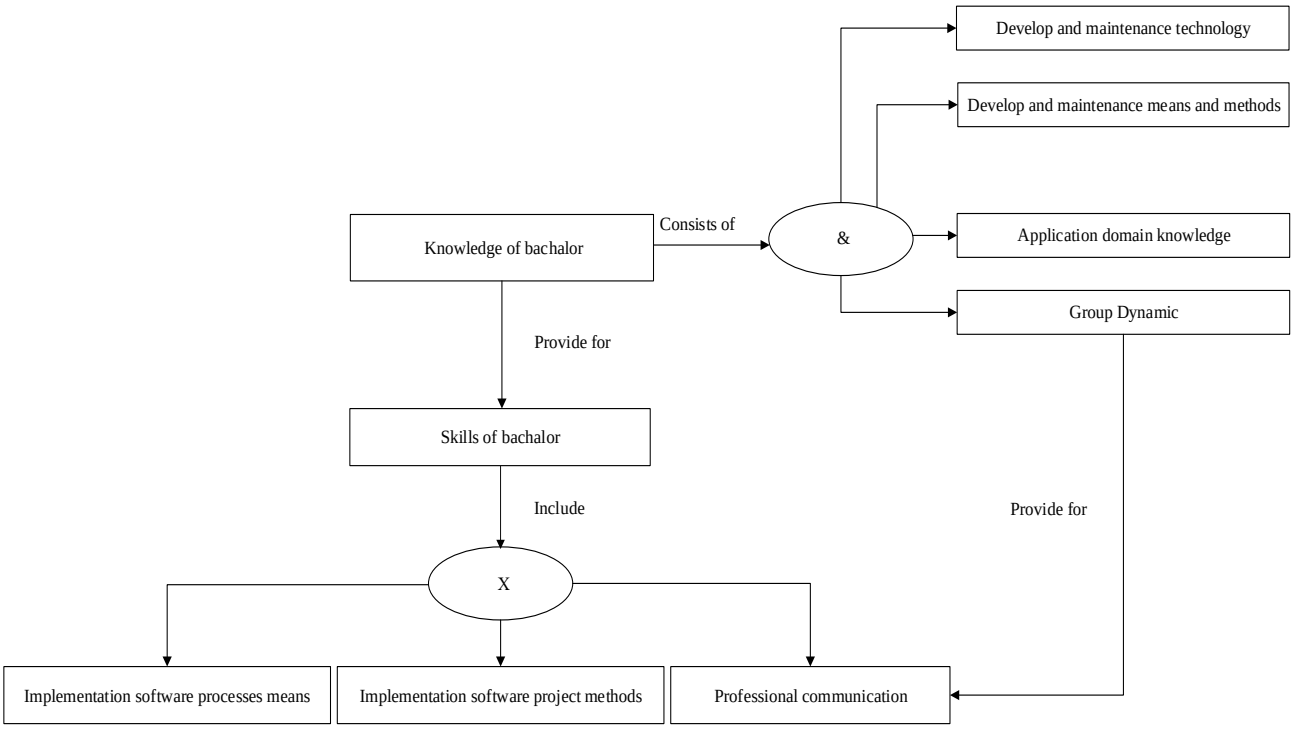

Figure 4. Knowledge and skills of bachelor ontology

Ontology-driven using of programming styles. Activities of a programmer will be more effective, and the software will be more understandable when within the process of software development, the programming styles (standards) will be used, providing clarity of software texts. Programming stylistics problems arose in the period before the structured programming, but nowadays they remain relevant (Sidorov, Sidorova&Pirog 2017). The existing problems of using the standards such as (Sidorova 2015): opposition of development team to use standards; developers "forgetting" to use standards; management thinking that the implementation of standards is too

expensive, are resolved by developing and using tools that automate the corresponding processes. In the paper (Sidorova 2015), new method of programming styles application based on the ontology has been proposed. To apply the style, a programmer should decide two tasks: study the description of the style; use and control the style during the coding. 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 tools. It is proposed to use the ontology as a form of knowledge representation about programming style (Sidorova Kramar 2014) (Fig. 5).

Using appropriate tool (e.g. Protégé (Protégé), a formal representation of programming style – an ontology is developed. A programmer for coding uses ontology as information resource. Therefore, two tools are required – one for creating an ontology and assisting the programmer, the second one to control the implementation of style during the coding (Fig. 6). For these tools two categories ontologies are needed. The first, the run time processes ontology for ontology-enabled architecture is the result ontology-driven pre-development. The second, the run time processes ontology for ontology-based architecture.

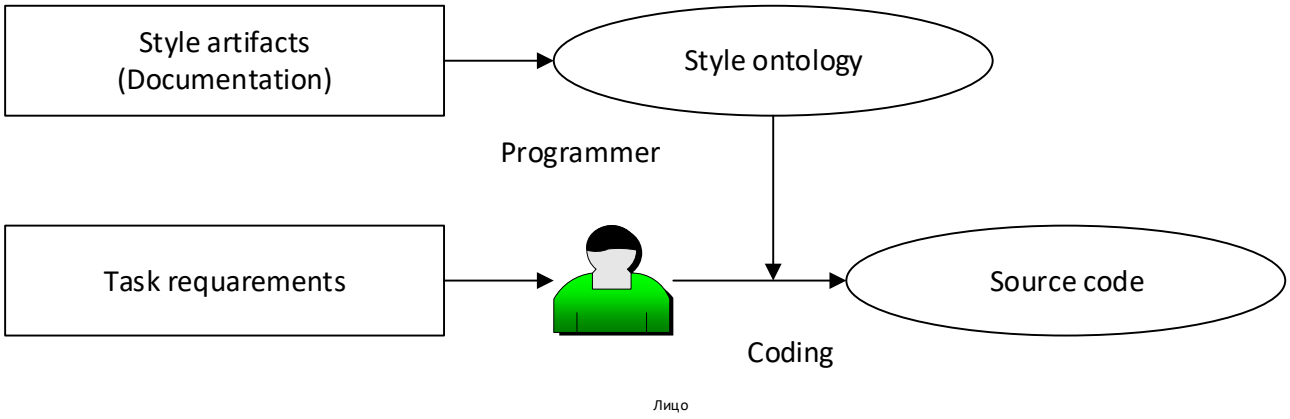

Figure 5. Ontology of style in programming

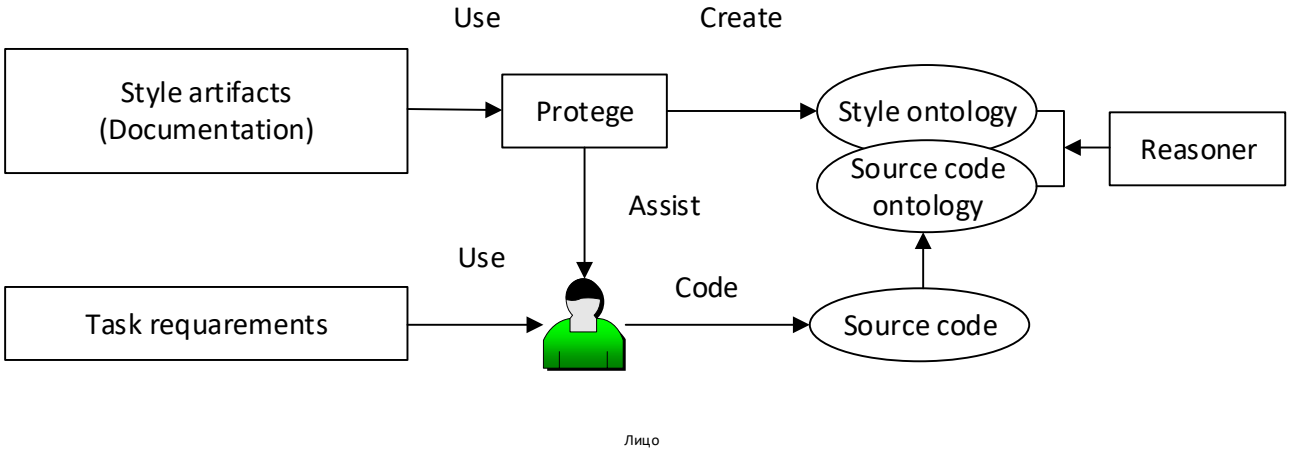

Figure 6. Ontology tools

The first tool, during predevelopment time processes in application domain creates the run time processes ontology template (reusable asset), which is defining, general programming standards properties. Style analyst using Protégé setup template on particular programming standard. After then the programmer uses ontology like software product ontology – information resource (Fig. 1) in run time to study programming standard (ontology-enabled architecture (Happel, Seedorf 2006). Examples of template ontologies on the fig.7, fig. 8 are presented.

The second tool is a reasoner [3]. In terms of descriptive logic, the reasoner solves one major problem – verifies consistency of the ontology (Dentler 2011). This problem has certain features for the task of programming style implementation (Fig. 9) (Sidorov, Sidorova&Pirog 2017).

Protege is used to create TBox, which includes terms describing programming style (Style Ontology, Fig. 7, 8, 9). The assertions about the source code – ABox (source code ontology, Fig. 9) are created according to the source code that is written by a programmer. Reasoner provides appropriate service based on TBox and ABox (Sidorov, Sidorova&Pirog 2017). But it should be not only assertion about knowledge base consistence, i.e. compliance of ABox assertions regarding TBox, but also indications of specific stylistic errors in the source code in case of inconsistency of the style knowledge base (Fig. 9). Thus, "regular" reasoner will not fully satisfy this service. Therefore, the implementation of corresponding Style Ontology Reasoner (SOReasoner) was implemented (Sidorov, Sidorova&Pirog 2017).

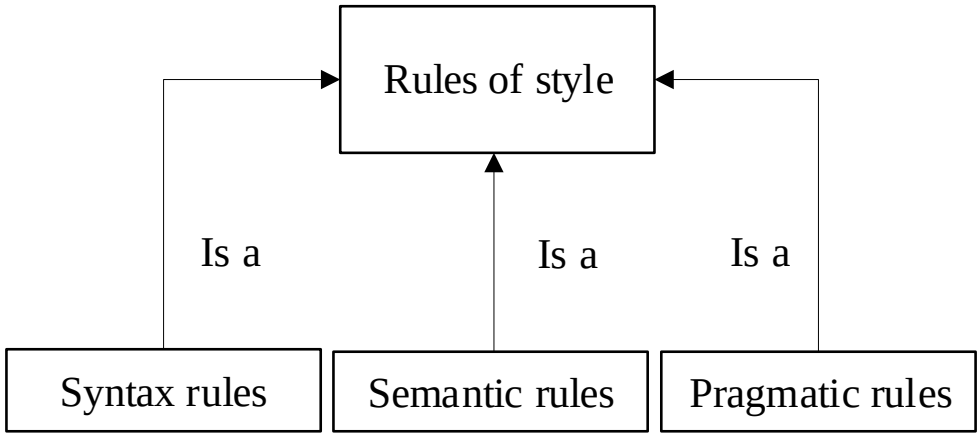

Figure 7. Style rules ontology

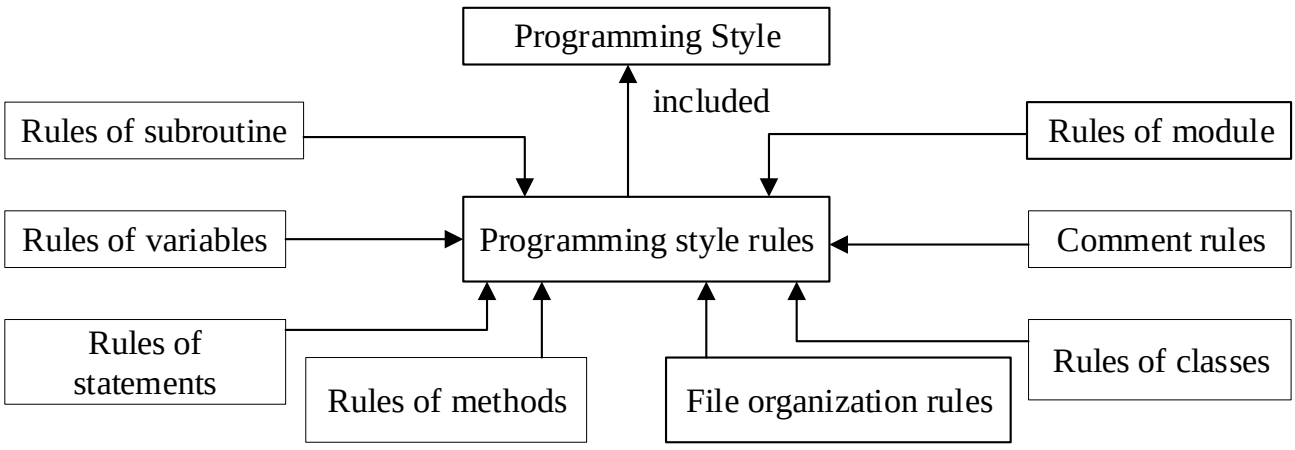

Figure 8. Programming style rules ontology

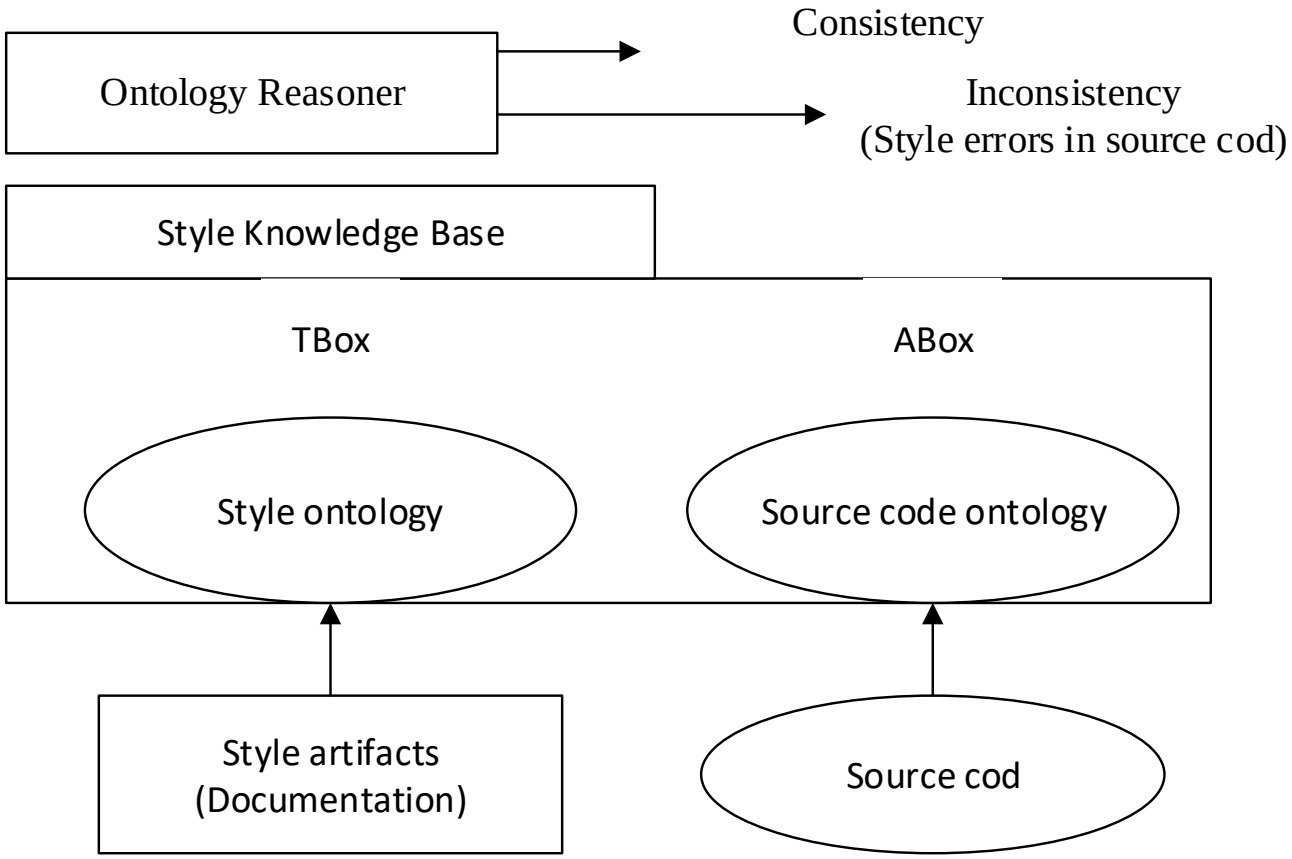

Figure 9. Model of style knowledge Base

As far as the ABox assertions for the operation of SOReasoner are not generated to TBox, the style ontology should be implemented in a format for SOReasoner. This format is recorded in a pattern on OWL, which is used for creating style ontology (Fig. 6). As far as the ABox assertions for the operation of SOReasoner are not generated to TBox, the style ontology should be implemented in a format for SOReasoner. This format is recorded in an ontology pattern on OWL, which is used for creating style ontology. Means for the creation of OWL template (OWLParser) and ABox (SourceCodeParser) are united in Style Ontology Reasoner (SOReasoner) that is the ontology-based architecture (Happel, Seedorf 2006).

Conclusion

In this research, categorization of the software engineering ontologies for supporting software life cycle processes is proposed. The categorization scheme is presented. Implementation details of categorization are given on the examples case studies of domain analysis and naming styles for the Java convention.

References

1. Baader F.D., Calvanese, D., Guinness Mc. The Description Logic Handbook: Theory, implementation, and applications. Cambridge University Press. 2003. 320 p.
2. Blum B. A taxonomy of Software Development Methods. *Communication of the ACM*. 1994. Vol. 37, N 11. P. 82–94.
3. Bondarenko M., Sydorov M., Morozova T., Mendzebrovskiy I. Model of a graduate of Bachelor's degree "Software engineering". Higher education, 2009. N 4. P. 50–61.
4. Guarino N. Formal ontology in information systems. *Proceedings of FOIS'98*, Trento, Italy, Amsterdam, IOS Press. 1998, P. 3–15.

5. Happel H., Seedorf S. Applications of ontologies in software engineering, *Proceedings of 2nd International Workshop on Semantic Web Enabled Software Engineering* (*SWESE 2006*), Athens, GA, U.S.A. 2006. P. 1–14.
6. Dentler K., Cornet R., Teije A., Keizer N. 2011, *Comparison of Rea*soners for large Ontologies in the OWL 2 EL Profile. Available from: http://www.semantic-web-journal.net/ sites/default/files/swj120_2.pdf. – Title from the screen.
7. Mendzebrovskiy I. Domain analysis tool, *Materials of the International Conference "Software Engineering 2017"*. Kyiv. 2017. P. 30.
8. Prieto-Diaz R. Domain analysis: Introduction, ACM SIGSOFT, Software engineering notes. 1990. Vol. 15, N 2. P. 47–54.
9. Protégé, Available from: http://protege.stanford.edu. Назва з екрана (*Protégé*, Режим доступу: http://protege.stanford.edu. Назва з екрана).
10. Ruiz F., Hilera J., Calero C., Ruiz F., Piattini M. Chapter 2. Using Ontologies in Software Engineering and Technology. *Ontologies for Software Engineering and Software Technology*, Berlin, Heidelberg, Springer, 2006. P. 62–102.
11. Sidorov M. *Software engineering*, NAU, Kyiv, 2007. P. 135.
12. Sidorov N. Software stylistics, *Proc. of the National Aviation University*, 2005. N 2. P. 98–103. Available from: 10.18372/2306-1472.24.1152.
13. Sidorov N., Sidorova N., Pirog A. Ontology-driven tool for utilizing programming styles, *Proc. of the National Aviation University*, 2017. N 2, P. 98–103. Available from: 10.18372/2306-1472.24.1152.
14. Sidorova N. Ontology-driven method using programming styles. *Software engineering.* 2015. N 2. P. 19–29.
15. Sydorov M., Mendzebroskiy I., Malin I. Domain analysis – way of proving development of branch educational standards. *Knowledge-based technologies*, NAU, Kyiv. 2009. B. 4, N 4. P. 59–63.
16. Sydorov N., Mendzebrovsky I., Sydorova N. Ontologies in software engineering. Kyiv. 2017. т. 198, P. 68–71.)
17. Sydorova N. Formation of preparedness of software engineering bachelors for professional communication, Proceedings of the National Aviation University. 2012. N 3. P. 94–100.
18. Sidorova N., Kramar Y. Ontology of programming style, Proc. the sixth world longest "Aviation in the XXI-st Century, 2014. Vol. 1. P.1.13.28 – 1.13.36.
19. Wongthongtham P., Chang E., Dillon T., Sommerville I. Development of a Software Engineering Ontology for Multi-site Software Development, *IEEE Transactions on knowledge and data engineering*. 2009. Vol. 21 (8). P. 1205–1217.

Література

1. Baader F.D., Calvanese, D., Guinness Mc. The Description Logic Handbook: Theory, implementation, and applications. Cambridge University Press. 2003. 320 p.
2. Blum B. A taxonomy of Software Development Methods. *Communication of the ACM*. 1994. Vol. 37, N 11. P. 82–94.
3. Бондаренко М., Сидоров М., Морозова Т., Мендзебровський І. Модель випускника бакалаврату «Програмна інженерія», *Вища школа*. 2009. № 4. С. 50–61.
4. Guarino N. Formal ontology in information systems. *Proceedings of FOIS'98*, Trento, Italy, Amsterdam, IOS Press. 1998, P. 3–15.
5. Happel H., Seedorf S. Applications of ontologies in software engineering, *Proceedings of 2nd International Workshop on Semantic Web Enabled Software Engineering* (*SWESE 2006*), Athens, GA, U.S.A. 2006. P. 1–14.
6. Dentler K., Cornet R., Teije A., Keizer N. 2011, *Comparison of Rea*soners for large Ontologies in the OWL 2 EL Profile. Available from: http://www.semantic-web-journal.net/ sites/default/files/swj120_2.pdf. – Title from the screen.
7. Мендзебровський І. Domain analysis tool. *Матеріали Міжнародної конференції «Інженерія програмного забезпечення 2017»,* Київ, 2017, С. 30.
8. Prieto-Diaz R. Domain analysis: Introduction, ACM SIGSOFT, Software engineering notes. 1990. Vol. 15, N 2. P. 47–54.
9. Protégé, Available from: http://protege.stanford.edu. – Назва з екрана (*Protégé*, Режим доступу: http://protege.stanford.edu. – Назва з екрана).

10. Ruiz F., Hilera J., Calero C., Ruiz F., Piattini M. Chapter 2. Using Ontologies in Software Engineering and Technology. *Ontologies for Software Engineering and Software Technology*, Berlin, Heidelberg, Springer, 2006. P. 62–102.
11. Sidorov M. *Software engineering*, NAU, Kyiv, 2007. P. 135.
12. Sidorov N. Software stylistics, *Proc. of the National Aviation University*, 2005. N 2. P. 98–103. Available from: 10.18372/2306-1472.24.1152.
13. Sidorov N., Sidorova N., Pirog A. Ontology-driven tool for utilizing programming styles, *Proc. of the National Aviation University*, 2017. N 2. P. 98–103. Available from: 10.18372/2306-1472.24.1152.
14. Sidorova N. Ontology-driven method using programming styles. *Software engineering.* 2015. N 2. P. 19–29.
15. Сидоров М.О., Мендзебровський І.Б., Малін І.В. Доменний аналіз – шлях доказової побудови галузевих освітніх стандартів, *Наукоємні технології,* НАУ, Київ, 2009. Т. 4, № 4. С. 59–63.
16. Сидоров М., Мендзебровський І., Сидорова Н. Наукові записки НаУКМА, Т. 198, Комп'ютерні науки, Київ. 2017. С. 68–71.
17. Сидорова Н.М. Формування готовності бакалаврів з інженерії програмного забезпечення до професійної комунікації, *Вісник НАУ*. 2012. № 3. С. 94–100.
18. Sidorova N., Kramar Y. Ontology of programming style, Proc. the sixth world longest "Aviation in the XXI-st Century, 2014. Vol. 1. P.1.13.28 – 1.13.36.
19. Wongthongtham P., Chang E., Dillon T., Sommerville I. Development of a Software Engineering Ontology for Multi-site Software Development, *IEEE Transactions on knowledge and data engineering*. 2009. Vol. 21 (8). P. 1205–1217.

Data received 11.02.2018

About authors:

Sidorov Nikolay,
Doctor of Engineering Sciences.
Professor.
Publications: 130.
Ukrainian: 118, Foreign: 12
http://orcid.org/0000-0002-3794-780X,

Sidorova Nika,
Postgraduate student
Department of Software Engineering.
National Aviation University, Kyiv, Ukraine.
Publications: 14.
Ukrainian: 14.
http://orcid.org/0000-0002-2989-3637,

Menzebrovski Igor,
Postgraduate student
Department of Software Engineering.
National Aviation University, Kyiv, Ukraine.
Publications: 10.
Ukrainian: 9, Foreign: 1.
http://orcid.org/0000-0001-9473-6876.

Location:

E-mail: nikolay.sidorov@livenau.net,
sna@nau.edu.ua
Tel.: 067 798 0361, 2343600.

Department of Software Engineering.
National Aviation University, Kyiv, Ukraine.

E-mail: nika.sidorova@gmail.com,
igor.menzebrovski@iteraconsulting.com