International Journal of Computer Engineering and Technology (IJCET)
Volume 11, Issue 5, September – October 2020, pp. 29-38, Article ID: IJCET_11_05_004
Available online at http://www.iaeme.com/ijcet/issues.asp?JType=IJCET&VType=11&IType=5
Journal Impact Factor (2020): 10.8630 (Calculated by GISI) www.jifactor.com
ISSN Print: 0976-6367 and ISSN Online: 0976–6375

MULTI-SITE SOFTWARE DEVELOPMENT WITH ONTOLOGY

Hemant H. Patel
Research Scholar, Dept. of IT,
Rai University, Ahmedabad, Gujarat, India

Dr. Nitesh M. Sureja
Department of Computer Science Engineering, Babaria Institute of Technology
Vadodara, Gujarat, India

ABSTRACT

One of the main goals of software engineering (SE) discipline is a way to find a higher level of abstraction and reuse software improve productivity and quality. The body is usually considered to be one or more of the technologies or artifacts used software life cycle stages can be used to achieve this goal. This article provides an overview, discussion, and analysis of the literature and implement a new solutions for ontology in se. We chose some examples of software development (including software products line, component development, generative programming and model-driven engineering) for our classification and discussion various methods are proposed in the literature. Confirmed the ontology is suitable for providing general vocabulary that should be avoided misunderstanding between the parties in the se, request specification, functional specification, variability management, component specifications, component conformity, model conversion and code generation. According to the review provides further research.

Key words: Software Engineering, Ontology lifecycle for Software Engineering, Ontology Development, Roles of Ontologies, Benefits and Issues on Ontology, Research Areas, Multi-site Software Development

Cite this Article: Hemant H. Patel and Dr. Nitesh M. Sureja, Multi-site Software Development with Ontology, *International Journal of Computer Engineering and Technology* 11(5), 2020, pp. 29-38.
http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=11&IType=5

1. INTRODUCTION

Today, with the growing demand for software applications for software development. It's getting more and more complicated. New procedures and technologies it seeks to simplify the software engineering process by, the goal is to reduce development time and costs by reusing components. Software system development is a complex activity that can mean participation of

people and machines (distributed and non-distributed). So different stakeholders, heterogeneity and new software features support software development the knowledge-based process [Force, 2001] [1].

To reduce this complexity, it may be useful to use an ontology. Ontology enable the definition of a common dictionary and frame user (human or machine). Benefits of software development from this conceptual model allows people to understand these concepts together participate in the software process. The ontology also alleviates integration problems it usually occurs when developing software applications. With the advent of the internet, software development has increasingly focused on the internet, enabling a multi-site environment that allows multiple teams across cities, regions, or countries to develop software distributed throughout the network. The benefits of multi-site software development have led large companies to shift software development to countries with relatively low wages.

1.1. Ontology Vs. Knowledge Representation

The term "ontology" is derived from his philosophy, which means the study of being or beginners and basic categories [2]. Therefore in this area used to indicate content that exists in the system model. In the field of informatics, ontology is an effort create a detailed and rigorous conceptual scheme within a given domain, usually a hierarchical data structure include all relevant elements and their relationships and rules (regulations) in the domain [3]. In the field of artificial intelligence, the ontology is clear conceptual specifications [4, 5]. In this ontology the definition combines the name of a concept with dissertation with description (e.g. class, relation, function) the meaning of the term and the formal axiom of the limiting axiom the interpretation and format of these terms are correct. For example, by default, all computer programmers have and basic ontology, including a standard library programming language or accessible files in the file system or list of other "things that exist". Representation, however sometimes it's bad for some problem domains, so a more professional architecture is required for authorization useful information for which we use ontology. Summary the aspect that represents knowledge of software engineering is as shown in Figure 1. Complete software engineering introduces the concept of software engineering domain knowledge be captured in ontology. Project or specific software development can only use the whole software engineering concept. Specific software engineering concepts for specific software development projects representing software engineering subdomains knowledge is also captured in ontology.

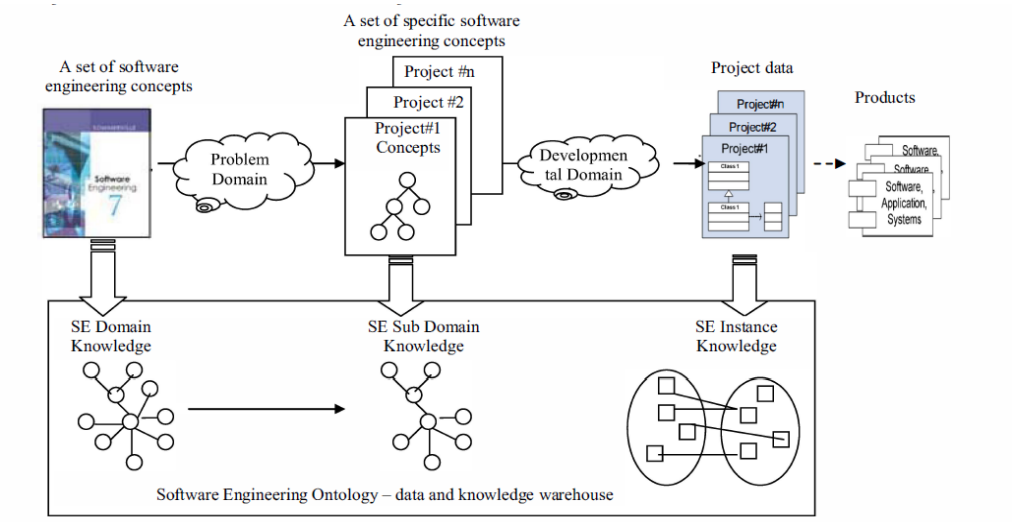

Figure 1 software engineering ontology as data and knowledge warehousing.

1.2. Fundamentals For Modeling Software Engineering Domain

Software engineering ontology is like other ontologies in other domain examples like, attributes and class. Software engineering ontology consists of examples represents specific project data, characteristics represent binary relationship between software engineering concepts / examples and classes representing software the engineering concept is interpreted as a collection containing specific content project data. The software engineering ontology class is created a description of software engineering concepts, indicate in the succeeding items the conditions that the project data must meet make him a member of the class. Representation of relationships between classes or instances according to the data type attribute and the object attribute, they come from two various sources of software engineering ontology. Data type attributes associate a class or instance with an XML schema value of data type or RDF text. Attributes of an object associated with one from class to class or from instance to instance. Association it is not always generated between classes and attributes express the class as a set of owned properties. In other words, the software engineering ontology class does not owned attributes of software engineering. They are independent of each other. Attributes of software engineering ontology are possible it has sub-attributes and can form a hierarchy of the ensuing attributes properties such as classes. A subclass specializes in its superclass the way they are specialized in sub-attributes super attributes. The relationship between classes is binary and it has a unique start and end point. Object attributes from the domain ontology class of the domain to and from range. The reference to the data type attribute comes from the value of the XML schema data type from domain to scope. It you can specify multiple classes as domains or scope of real estate. If more than one category is specified, then the domain or attribute scope is to be understood as unification of classes. Software engineering ontology support a fixed range of ontology classes has been defined. Used to define description of the enumeration class type. Software the project ontology also supports fixed data values data range of a data type attribute. Specifies the dataset data range value. In the ontology of software engineering properties are defined by default as scope and domain and both scope and domain can be used in any software category engineering ontology. Unlimited ownership and join the class that defines it.

2. SOFTWARE ENGINEERING DEFINITIONS

Software engineering or software process is a series of activities, methods, procedures, and conversions for software development and maintenance and related products, usually performed by a group of organized people according to the structure with the support of technical tools [Acuna Etc.] [6]

Software process modeling has been defined as a language, diagram, or numerical representation of the mode of operation (process) [Menzel and Gruninger, 2001] [7]. The result of software process modeling is a software process model, called process model. The process is represented by a model, in an abstract way and with different levels of detail, different processes completed, in progress, or involved in the proposed software process [Acuna and Ferre, 2001] [8].

A solid foundation for process modeling at the same time, describe the general structure of the process described by the model as a category of possible instances of the structure. The software process model can be a descriptive model [Acuna Etc.] [6], the descriptive model focuses on internal software development organization. On the other hand, the descriptive model focuses on criteria required to perform the software process, method of software it must be developed. In the future, there may be manual and automatic models consider. Manually explain that the model is a standard and a method software lifecycle guide. The automated model of the specification is computer processed specifying software process standards can

help participating agents explain the software process model in the software process in an automatic way.

Use process models for the ensuing purposes: to better understand processes and communication, process management, designed process development opinion or opinion [Acuna and Juristo, 2005] [9]. In addition, they can be analyzed, verified, simulated and performed. Therefore, it is a process model provide a better way to define development, maintenance, and development software process, including processing of one activity or process as one whole [Acuna and Sanchez-segura, 2006] [10].

Model driven software development (MDD) is based on models, modeling and model conversion. This model is used to consider the problem domain and the solution domain. Modeling methods have evolved from coexistence convert model and code to model only it is used as a tool for discussion in the software process. Between the two is a model-oriented approach if the definition is sufficiently detailed, consider the model as code. Transformation rules and patterns provide tools for generating code from such models [Brown et al., 2005] [11].

Domain-driven development refers to the development of internal software applications specific field or application field [Hruby, 2005] [12]. Domain engineering the goal is to identify, model and implement requirements it is used in specific application areas [Falbo et al., 2002] [13].

Ontology-based software development or engineering has been defined as a method based on ontology, with respect to semantic constraints, adapt to new constraints in a dynamic way [Tanasescu, 2005] [14]. It could considered a special case of a managed software model, where the model is ontologies based on different levels of abstraction.

2.1. Roles of Ontologies for Software Development

Ontologies, for software design and development, can be used with the following objectives [Ruiz and Hilera, 2006] [15]:

- Specification: the ontology is used to specify the requirements and definition of a component (informal use) or system function.
- Trust: ontology is used to control system design.
- Reusability: ontologies can be organized into modules to define domains, subdomains and related tasks can be reused later and / or adapt to other problems.
- Search: the ontology is used as a repository of information.
- Reliability: ontologies can be used in (semi) automatic consistency exam.
- Maintenance: the ontology can improve the use and storage of documents system maintenance.
- Knowledge acquisition: ontology can be used as a guide the process of acquiring knowledge.

Within Software Engineering, two main roles for ontologies have been considered [Hesse, 2005] [16]:

- Software engineering process ontology: definition, reuse the use of ontology facilitates the integration of software components as a conceptual basis.
- Ontology in software engineering: use of ontology describe the structure and terminology of software engineering the domain itself.

2.2. Benefits on the Use of Ontologies

Ontologies provide benefits regarding the process of software development, which could be summarized as follows, [Falbo et al., 1998] [17], [Falbo et al., 2002] [13], [Liao et al., 2005] [18], [Ruy et al., 2004] [19], [Abran et al., 2006] [20], [Ruiz and Hilera, 2006] [21]:

- Ontology provides vocabulary specifically for software process, elimination of inconsistencies between concepts and terminology.
- Use ontology and alignment technology to address compatibility problems without having to change an existing model.
- Ontology can help develop the scale of the software collection process data on the internet and the use of the semantic web.
- Ontology can not only transfer knowledge but also simplify development cycle from project to project.
- Ontology supports agreement between software developers, and it is used as a domain model.
- The ontology simplifies the knowledge acquisition process by sharing the ensuing items of the same concept of different software applications.
- Ontology allows you to reduce the mismatch between terminology and conception in the resulting ways forcing different people to share understanding and communication when the user performs an ontological analysis.
- The ontology also ensures fine communication between tools shaping part of the environment.
- When the body is used as a machine-readable image, it helps develop tools for software engineering activities.

2.3. Issues on Ontology–based Software Process

Although ontologies are considered a useful element within software engineering activities, some issues should still be congenital in mind when developing ontology–based software development projects [Hesse, 2005] [16]:

- Ontology-based methods are suitable for software development items that belong to a group of items in the same domain.
- The ontological approach makes it possible to extend the concept of re-use in the modeling phase, this is not just one of the usual implementations.
- Ontologies can be considered as reusable components of a model system.
- Model-based development can benefit, for example, from the use of ontology model reuse mechanism.
- Ontology-based methods affect all software development processes from requirements analysis and domain analysis to all phases of integration, deploy and use developed software.
- An ontology-based approach allows the use of ontologies for promotion long-term software development and interoperability solutions and reuse the issues.

3. RESEARCH AREAS

Software Product Line (SPL), sometimes called software series, which means a group of software products consisting of common architecture and sets the number of reusable assets used in the manufacture of the system (Asikainen et al., 2007) [22]. SPL courses focus on reuse

basic assets systematically, planned and strategically in order to produce various products that meet the succeeding conditions specific market segments (Duran-limon et al., 2015) [23]. SPL tends to use general system functions increase productivity and quality, reduce development time, cost and complexity (Strmecki etc., 2015) [24]. The main goal of SPL is to avoid development create software from scratch through reconfiguration and reuse existing SPL in various projects. Sound pressure level development consists of two processes: Domain Engineering (DE) and Application Engineering (AE) (Magdalenic et al., 2013) [25]. The aim of de is to develop a general systems family architecture and design and production plans of family members (Strmecki, etc., 2015) [24]. In the de process universality and it has determined the variability of the product line and can be reused assets that achieve the required variability are mechanisms of construction and solution of variability definition. The AE process is related to derivation and develop specific products. Product special requirements and use the defined variability mechanism k- develop products on time and at the lowest price, but (Nguyen et al., 2015) [26].

Component - Based Development (CBD OR CBE) is a clause-based software development paradigm plug and play reusable software components development style. This is a reuse-based method definition, implementation and composition of a free bond independent components. A software dream component integration should be able to compose a complex system consisting of common components (Ringert et al., 2014) [27]. SE consists of components suitable for development in distributed systems such as internet. The main goal of a distributed CBD is to increase reliability and maintainability of the software system by reusing components (Pahl, 2007a) [28]. Component it also plays an important role in today's popular Service - Oriented Architecture (SOA), because components can converted to web services that they can provide services of other components.

Model driven engineering (MDE) is a paradigm emphasize the model based on abstraction and automation code generation. In MDE it is the basic function of the system take pictures with the appropriate models and code the generator is used to automatically generate the source codes of various modeling entities (Magdalenić etc., 2013; Lilis et al., 2014) [25] [29]. Model representative part or simplified view of the system. Is it right the system is used to replace the studied system. Inside the MDE paradigm, the model is more than just document artifacts, but reusable artifacts used throughout the SE life cycle (Rodrigues da silva, 2015) [30]. MDE combines layering modeling technology for automatic conversion and code generation where the generated code can contains special tags with model information. It is based on three layers: (1) Computational Independence Model (CIM) solved system when viewed from a non-computational perspective; (2) The Platform Independent Model (PIM) defines a system as virtual machines or non - technology calculations abstract (3) includes a Platform Specific Model (PSM) the platform model that makes up the platform implementation-specific models for specific implementations. By definition, MDE has nothing to do with the platform, but the MDE prototype is based on unified modeling language (UML) (Pahl 2007b; Katasonov and Palviainen, 2010) [31] [32]. The MDE provides a way to: (1) specify the system is platform independent; (2) choice specific platform; (3) change specifications the platform of choice (Bartolo Espiritu et al., 2014) [33]. MDE developers first use UML to model the entire system, then perform iterative steps to refine the model. Final the model is specific enough to be executable code from this. Due to the high operating conditions of the developer efficiency can also be achieved on an abstract level. However, it is difficult to use MDE for general purposes programming and a lot of complexity can be hidden generator (Zimmer and Rauschmayer, 2004) [34].

Generative programming (GP) is a field use the generator for Automatic Programming (AP) facilitate the application development process (Magdalenic et al., 2013) [25]. The generator is

defined as programs that require higher technical parameters software and its implementation (Czarnecki and Eisenecker, 2000) [35]. Generated Programming using de technology can be used SPL AE. In addition to UML modeling, GP also uses functions function-oriented model method of analysis (FODA). Function modeling can defined as a creative activity to model a common model variable properties of terms and their interdependence. The term function refers to attributes stakeholder - related system. Features for capture common features and variability between products. Domain model generation focuses on mapping between the problem space and the solution space. Problem space indicates a set of product line functions, the representation of the solution space is based on implementation abstraction contained in the specification. Generator mapping two spaces using specifications appropriate implementation (Magdalenic et al., 2013) [25]. GP uses the meta-programming technology it refers to the goal of development is to read, generate, analyze or convert other programs or even modify them when running alone (Strmecki et al., 2015) [24].

4. SOFTWARE ENGINEERING ONTOLOGY IN PRACTICE

The actual use of software engineering ontology examples are given in this section. We start from examples of problems encountered by several sites surroundings. Then use software engineering there is an ontology to solve these special problems explanation.

Rewriting text is difficult because it is a task in a development team environment team members, geographically widely distributed participate in many projects simultaneously. It is a typical means of global communication is however, for multisite, this is neither effective nor sufficient surroundings. Below you have shown an example of such text the effect of transcription in a multi-site environment is not clear.

The Insurance Registered Driver is a subtype of the Customer. This means that for every Insurance Registered Driver object there must be a corresponding Customer object. However, in the Customer Object we store values like customer type, Insurance history values and rental history value. It does not make sense to have this values for the Insurance Registered Driver.

Now base on above example we have perform below pseudo code.

1. Splitting paragraph into sentences.
2. Chunking of sentences into words.
3. Performing POS (Parts of Speech) Tagging.
4. Performing xyz for finding Subject Predicate Object triplets.
5. Processing using Annotations to relate and use the Subject and Object.
6. Storing info on RDF database.

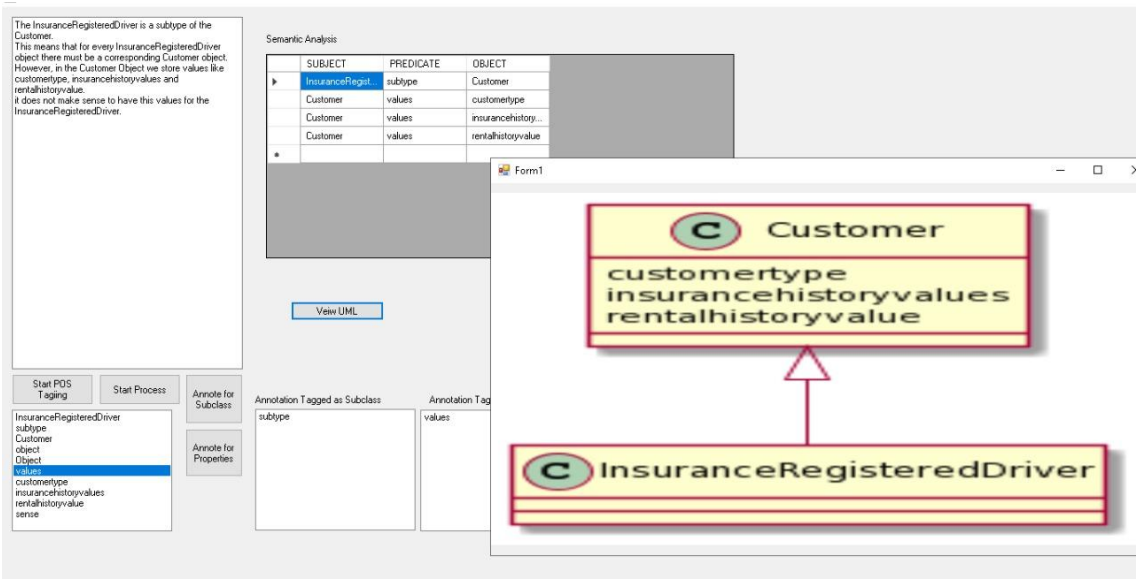

Figure 2 Multi-site software

Thanks to ontology-based software engineering can annotate or analyze software engineering conditions with the concept of software engineering ontology, and can check the necessary details and related information. You can see the insurance terminology in figure registered drivers, customers and rental customers can be annotation as a class; subtype expressions can be annotated as a subcategory; type of customer, conditions of insurance history the value and value of the rental history can be annotated as properties; driver clause registered in the insurance customer objects can be marked as objects.

Classes, subclasses, attributes, and objects apply separately the concept of class, generalization relationship, class attributes and objects in software engineering ontology. By specifying an instance of the related ontology class we can find information about these cases dynamic and automatic. For example, in figure 2 by annotating the word "customer" as a class, you get a class of detailed information customers such as their attributes, their operations and the relationship with other classes is automatic introduced in figure. Value, class diagram can be drawn automatically.

This is done using reference software engineering ontology states that the concept class has contains attributes, operations and relationship between other classes. Automatically Drawing details helps others to develop understand the content you want to reduce misunderstanding and disambiguation.

5. CONCLUSION

In this article, we introduced multi-site software development. We describe the issues and benefits in detail. We have identified several research area related to multi-site software development. After this, it has been confirmed that there is a need to address the availability of communication and coordination, a unified semantic knowledge sharing and knowledge sharing platform to solve problems related to multi-site software development. After that, we present one solution for that. Ontologies, knowledge-based systems, and knowledge management can be used as part of the conceptual solution for developing multi-site software development methods.

REFERENCES

[1] Force, S. E. T. (2001). Ontology driven architectures and potential uses of the semantic web in systems and software engineering, [Force, 2001].

[2] Witmer, G. Dictionary of Philosophy of Mind - Ontology. 2010 [cited May 11, 2011]; Available from: http://www.artsci.wustl.edu/philos/MindDict/ontology.html.

[3] Wikipedia. Ontology (computer science) From Wikipedia, the free encyclopedia. 2011 [cited 8 June 2006]; Available from: http:// en.wikipedia.org/wiki/Ontology _ computer _ science.

[4] Gruber, T.R. A translation approach to portable ontology specification. In Knowledge Acquisition. 2009

[5] Gruber, T.R. Toward principles for the design of ontologies used for knowledge sharing. In International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation.2008. Padova, Italy: Kluwer Academic Publishers, Deventer, the Netherlands.

[6] Acuna, S. T., Antonio, A. D., Ferre, X., Lopez, M., and Mate, L. The software process: Modelling, evaluation and improvement.

[7] Menzel, C. and Gr¨ uninger, M. (2001). Formal Ontology and Information Systems, chapter A Formal Foundation for Process Modelling, pages 256–259. New York, ACM Press.

[8] Acuna, S. T. and Ferre, X. (2001). Software process modelling. In Proceedings of ISAS-SCI (1), pages 237–242.

[9] Acuna, S. and Juristo, N. (2005). Software Process Modeling, volume 10 of International Series in Software Engineering, chapter Software Process Modeling: A Preface, pages xiii–xxiv. Springer NY.

[10] Acuna, S. and Sanchez-Segura, M. (2006). New Trends in Software Process Modeling, volume 18 of Series on Software Engineering and Knowledge Engineering, chapter Preface, pages v–ix. World Scientific.

[11] Brown, A., Conallen, J., and Tropeano, D. (2005). Model- Driven Software Development, chapter Introduction: Models, Modeling, and Model-Driven Architecture (MDA), pages 1–16. Springer-Verlag Berlin Heidelberg.

[12] Hruby, P. (2005). Ontology–based domain–driven design. In Proceedings of Object-Oriented Programming, Systems, Languages And Applications (OOPSLA'05), San Diego, California, U.S.A.

[13] Falbo, R., Guizzardi, G., and Duarte, K. (2002). An ontological approach to domain engineering. In Proceedings of 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02), pages 351–358, Ischia, Italy.

[14] Tanasescu, V. (2005). An ontology–driven life–event portal. Master, Computer Science.

[15] Ruiz, F. and Hilera, J. (2006). Ontologies for Software Engineering and Software Technology, chapter Using Ontologies in Software Engineering and Technology, pages 49–102. Springer-Verlag Berlin Heidelberg.

[16] Hesse, W. (2005). Ontologies in the software engineering process. In Lenz, R., editor, Proceedings of Tagungsband Workshop on Enterprise Application Integration (EAI2005), Berlin, Germany. GITO–Verlag.

[17] Falbo, R., Menezes, C., and Rocha, A. (1998). A systematic approach for building ontologies. In Heidelberg, S.-V. B., editor, Proceedings of 6th Ibero–American Conference on AI, number LNCS1484 in Lecture Notes in Artificial Intelligence, pages 349–360, Lisbon, Portugal.

[18] Liao, L., Qu, Y., and Leung, H. (2005). A software process ontology and its application. In Proceedings of Workshop on Sematic Web Enable Software Engineering (SWESE), Galway, Ireland.

[19] Ruy, F., Bertollo, G., and Falbo, R. (2004). Knowledge-based support to process integration in ODE. CLEI Electronic Journal, 7(1).

[20] Abran, A., Cuadrado, J., Garcıa-Barriocanal, E., Mendes, O., Sanchez-Alonso, S., and Sicilia, M. (2006). Ontologies for Software Engineering and Software Technology, chapter engineering the Ontology for the SWEBOK: Issues and Techniques, pages 103–121. Springer-Verlag Berlin Heidelberg.

[21] Ruiz, F. and Hilera, J. (2006). Ontologies for Software Engineering and Software Technology, chapter Using Ontologies in Software Engineering and Technology, pages 49–102. Springer-Verlag Berlin Heidelberg.

[22] Asikainen, T., T. Männistö and T. Soininen, 2007. Kumbang: A domain ontology for modelling variability in software product families. Adv. Eng. Inform., 21: 23- 40. DOI: 10.1016/j.aei.2006.11.007

[23] Duran-Limon, H.A., C.A. Garcia-Rios, F.E. Castillo- Barrera and R. Capilla, 2015. An ontology-based product architecture derivation approach. IEEE Trans. Software Eng., 41: 1153-1168. DOI: 10.1109/TSE.2015.2449854

[24] Strmecki, D., D. Radošević and I. Magdalenić, 2015. Web form generators design model.

[25] Magdalenic, I., D. Radosevic and T. Orehovacki, 2013. Autogenerator: Generation and execution of programming code on demand. Expert Syst. Applic., 40: 2845-2857. DOI: 10.1016/j.eswa.2012.12.003

[26] Nguyen, T., A. Colman and J. Han, 2015. A feature based framework for developing and provisioning customizable web services. IEEE Trans. Services Comput. 1374: 1-1.

[27] Ringert, J.O., B. Rumpe and A. Wortmann, 2014. Multiplatform generative development of component and connector systems using model and code libraries. Proceedings of the 1st International Workshop on Model-Driven Engineering for Component-Based Systems, (CBS' 14), Valencia, Spain, pp: 26-35.

[28] Pahl, C., 2007a. An ontology for software component matching. Int. J. Software Tools Technol. Transfer, 9: 169-178. DOI: 10.1007/s10009-006-0015-9

[29] Lilis, Y., A. Savidis and Y. Valsamakis, 2014. Staged model-driven generators: Shifting responsibility for code emission to embedded metaprograms. Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development, Jan. 7-9, IEEE Xplore Press, pp: 509-521.

[30] Rodrigues da Silva, A., 2015. Model-driven engineering: A survey supported by the unified conceptual model. Comput. Lang. Syst. Struct., 43: 139-155. DOI: 10.1016/j.cl.2015.06.001

[31] Pahl, C., 2007b. Semantic model-driven architecting of service-based software systems. Inform. Software Technol., 49: 838-850. DOI: 10.1016/j.infsof.2006.09.007

[32] Katasonov, A. and M. Palviainen, 2010. Towards ontology-driven development of applications for smart environments. Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops, Mar. 29-Apr. 2, IEEE Xplore Press, pp: 696-701. DOI: 10.1109/PERCOMW.2010.5470523

[33] Bartolo Espiritu, F., A. Sanchez Lopez and L.J. Calva Rosales, 2014. Towards an improvement of software development process based on software architecture, model driven architecture and ontologies. Proceedings of the International Conference on Electronics, Communications and Computers, Feb. 26-28, IEEE Xplore Press, pp: 118-126. DOI: 10.1109/CONIELECOMP.2014.6808578

[34] Zimmer, C. and A. Rauschmayer, 2004. Tuna: Ontologybased source code navigation and annotation. Proceedings of the Workshop on Ontologies as Software Engineering Artifacts, (SEA' 04), pp: 1-9.

[35] Czarnecki, K. and U.W. Eisenecker, 2000. Generative Programming: Methods, Tools and Applications. 1st Edn., Addison-Wesley Publishing Co., ISBN-10: 0201309777, pp: 832.