Symbiosis

www.symbiosisonline.org
www.symbiosisonlinepublishing.com

Review article Journal of Computer Science Applications and Information Technology Open Access

A Model-driven Ontology Approach for Developing Service System Applications

Petrenko OO[1*] and Petrenko AI[2]

[1]PhD student, National Technical University of Ukraine, Kiev Polytechnic Institute, Ukraine

[2] Professor, Head of System Design Department, National Technical University of Ukraine, Kiev Polytechnic Institute, Ukraine

Received: November 11, 2017; Accepted: November 22, 2017; Published: December 05, 2017

***Corresponding author:** *Petrenko OO, PhD student, National Technical University of Ukraine, Kiev Polytechnic Institute, 37 Peremogu Rd, Kiev, Ukraine, E-mail: petrenko.chiffa@gmail.com*

Abstract

This paper aims to survey the new methodology and tools for user-defined applications development, based on Service-Oriented Computing (SOC) and Model-Driven Approach (MDA), when all computing units, both hardware and software, can be treated as services and domain ontology acts as a meta-model basis to generate a conceptual model for specific information systems. It allows programmers to create new applications using ready-made service available on the network, from one side, and an application itself may create (compose) the infrastructure it needs to run, from other.

Keywords: Service-oriented computing (SOC); Web services; Model-driven approach (MDA); Services Repository; Application Software;

Introduction

In recent years there is a tendency to create information systems of the so-called "*extended enterprise*", which brings together the company itself, its suppliers, partners and customers in a single system to meet needs for integration and interoperability of applications. Therefore, applications interaction is required as within a single information system so between the systems of different participants in the business process. The solution, actively developed by leading providers of information technology (IBM, Microsoft, Oracle and others), is to move from a centralized IT infrastructure to an architecture which enables rapid creation of new systems from a set of available services. It means the transition to the so-called ***Service-oriented Architecture (SOA)*** and its implementation in forms of process-oriented business models ***(Model-driven architecture, MDA)*** [1-5].

SOA is a style of software architecture, where business functionality or application logic becomes available to users of SOA as a set of multiple shared services in IT networks. Services in an SOA are business process modules, or modules of the applications functionality with open interfaces, caused by messages. The main reason for the emergence of SOA is the cherished dream of the software industry to replace the "manual" programs coding by "industrial" applications assembling from the "standard components", as it happens in a car or other "traditional" industries. Program components may be located on different nodes of the network and be independent, loosely coupled, replaceable services-applications. Moreover, the integration of legacy applications (legacy) is advantageously carried out with the use of this technology, when certain, most important part of the existing functionality is "encapsulated" and submitted with standardized interface. In cases where the services represent computational procedures, systems are considered as systems of ***Service-oriented Computing (SOC)***. SOC refers to the set of concepts, principles, and methods that represent computing in a collection of loosely coupled services.

Now it is time to move to a ***new programming paradigm***, not associated with objects but with business processes and their components - business functions. The main idea is to build an application by discovering and using services, available on the network, to perform a certain task. This approach does not depend on specific programming languages and operating systems, and involves the use of SOA applications, built on the basis of formalized business processes and functions been presented in the form of reusable services with transparently described interfaces. The article discusses the development of systems services for the implementation of business processes via a process-oriented business model and features of ontologies relevant subject areas.

The rest of the paper is organized as follows. In Basic of Service Systems Engineering section and Ontologies in Service Systems Design section basic concepts are reviewed. Next, we discuss model - driven approach in service systems design. There is a comparison between the proposed approach to software development and other approaches in addition to the conclusion, and suggestions for future works.

Basic of Service Systems Engineering

At present, due to the leading World IT companies' efforts, the methodology and tool for design service-oriented systems are developed that in the best way is reflected in the serial publications ***"Industrial SOA"*** [6]. They are oriented on applications composing, mainly with the use of ***Web services*** (WS). Web services are referred to software components that use ***XML (Extensible Markup Language)*** as the data format, ***Web Services Description Language (WSDL)*** standards to describe their interfaces, ***Simple Object Access Protocol (SOAP)*** to describe the format of the received and sent messages and standard ***Universal Description Discovery and Integration (UDDI)*** to create catalogs of available services [7].

To compose services into necessary applications procedures ***"choreography"*** and *"orchestration"* are proposed, when Web services choreography defines the interaction between the services using the exchange notifications, and orchestration describes the interaction of services within a single business process, in particular, using the language ***WS-BPEL (Web Services Business Process Execution Language)***. The easiest way of business process presentation is a simple orchestration of Web services, which is the flow of tasks (***workflows***), when each service follows the previous, without delay or space and its effect ends just before the next service action begins [8].

The IBM Company offers SOA implementation tools named ODOE (On Demand Operating Environment) to meet the needs of service-oriented IT solutions, the main of which are [9]:

• ***Application Services*** (AS), which include all components required to build the information system been configured.

• ***Enterprise Service Bus*** (ESB), which acts as an intelligent layer (middleware) for connecting information systems, different data and other services, which are usually distributed throughout the enterprise IT environment.

• ***Business services*** (BS), which present an open part of the business process (business functionality), and which ensure the implementation of a given value to the customer's request.

• ***Common Services*** (CS) for providing personalized delivery and individual treatment, as well as other utilities, such as reporting. These services allow to define and to execute conditional flows, mainly for logical business function services.

• ***Information Management Services*** (BPM), which provide a standardized way of representing, accessing, maintaining, managing, analyzing and integrating data and content from heterogeneous information sources.

• **Infrastructure Services** (IS), which are independent from a platform, and which allow all the other services to be established, implemented and monitored in a specific infrastructure, consisting of operating systems and network and hardware systems.

Information about the most well-known SOC development tools are shown in Table 1 below [10].

Table 1: Major SOC Tools Comparison

SOC Tools	Description	Vendor
Websphere	It has many SOC features, such as service composition, discovery, and modeling.	IBM
.NET	This supports many features of SOC including service creation, discovery, composition, deployment.	Microsoft
HP SOA Center	This supports SOC features such as service composition, modelling, integration, management.	HP
Oracle SOA Suite	This supports service creation, deployment, composition, orchestration.	Oracle
Enterprise SOA	This supports service creation, deployment, composition.	SAP
Weblogic	This supports service deployment, composition, management, policy.	BEA

It should also be noted that at present time there are a number of known repositories of general purpose services, such as: the External EGI service catalogue for researchers (https://www.egi.eu/services/); Internal service catalogue for EGI Federation members (https://www.egi.eu/internal-services/); INDIGO IAM services (https://github.com/indigo-iam/iam); EUDAT service registry(http://eudat.eu/ support-request); GEANT Cloud services (https://clouds.geant.org/); Fi-Ware services for Internet of Things (https://www.fiware.org/ developers-entrepreneurs/),

With expanding attention to business Web-based solutions there are increasing demands for business services and business functionality. It is vitally necessary to identify possible ***invariant services*** for systems, focusing on human activities (electric networks, water supply, transportation systems, health-care system, the education system, the banking and financial systems, online retail, tourism system, media and entertainment, and others). This will help to create a repository of ***multidisciplinary invariant services as the building blocks*** of the relevant service systems. The amount of research is so great that requires the collective efforts of many partners and participators.

Such a repository could be the basis for creating the Service Science Knowledge Environment, which will bring together academia, industry and government, as well as other European institutions [11]. In this first stage the resources knowledge can belong to different application domains, such as: e-administration, e-government, e-health, production, energy consumption, with a focus on the ability of the Smart Grid, advanced software services, supply chain and logistics services, tourism and recreation, advanced services in the field of telecommunications. The list of service sectors, providing a real contribution to a modern economy, is not limited to the names listed above. Smart transport, smart buildings, smart water are only some other possible applications service sectors for which new services.

Web services described in WSDL, contain predominantly syntactic information, and this makes it difficult to organize the automatic Web service discovery and execution service compositions. Therefore, it was proposed (in addition to semantic annotations) to use directly semantic information in Web services descriptions, resulting in appearance of ***Semantic Web services (SWS)*** with own description languages, such as ***OWL-S (Web Ontology Language for Web Services)*** and ***WSMO (Web Service Modeling Ontology)***. SWS are created on the basis of the existing Web-2 technology to provide dynamic discovery services, their composition, and Web services call that are understood by the computer also. The result is the ability to automate these procedures which currently require software developers help to run [12].

The configuration and coordination of services in architecture, based on the services, and the composition of services and processes are equally important in modern service systems. As it was noted above, the services interact with each other via messages. Message can be accomplished by using a template ***"request/response"***, when at a given time only one of the specific services is caused by one user (an "one-to-one" connection or synchronous model); using a template ***"publish/subscribe"*** when on one particular event many services can respond to (an "one-to-many " communication or asynchronous model); using ***intelligent agents*** that determine services coordination, because each agent has at its disposal some of the knowledge of the business process and can share this knowledge with other agents [13]. NTUU "KPI" explores hybrid Event-Driven Service-Oriented Architecture (EDSOA), in which services generate events and thus translate the business process from one state to another. If there is a "network" of sensors and software agents that monitors important business events in all enterprise hardware and software components, then whole business is controlled technologically not blindly, but having a clear picture of everything that is happening at the moment in the company.

Ontologies in Service Systems Design

In recent years different levels of ***ontologies*** (Domain, Application, and Task ontologies) are used in service systems development [15]. Ontologies, described in the OWL language, are formal explicit descriptions of concepts in a given domain area. Classes are the central item of the ontology. For example, Action class may represent all procedures (running tasks, data transmission, data flow control, etc.). Specific procedures are instances of the class. A class can have ***subclasses*** that represent more specific concepts than the ***superclass***. ***Slots*** describe properties of classes and instances, say, a Task procedure may contain or the file ***(contains File)***, or to create a resource ***(creates Resource)*** or to depend on certain conditions ***(has Dependency).***

Slots can have different ***facets*** describing the value type, the numerical values of the permitted limits and other changes. Ontology together with a set of individual class instances constitutes ***a knowledge base***. In fact, it is difficult to determine where the ontology ends and where the knowledge base begins. Ontologies are shared knowledge. They enable high level of interaction between the human users and "intelligent" applications. Ontologies do not describe systems, only domains. Hence, in a software developing process, they should play the role of an analysis model, not of a design or implementation model.

OMG (Object Management Group) proposed a hierarchical system of service systems ***semantic description*** for their modeling, which consists of an ***M0 level*** *(objects)*, the ***M1 level*** *(models)*, the ***M2 level*** *(meta-model or language level)*, and the ***M3*** *(meta-meta-model or language description level)* which is often called ***Meta Object Facility (MOF)*** [16]. In general, meta-models are language specifications, not only of modelling, but also of arbitrary languages. If two Software Engineering tools agree on the same meta-model, they impose the same structure on their models, so that they can easily exchange them. Software processes, being specific work flows, can be meta-modelled and used to construct software environments.

The process of designing system services can be modelled as a process of transformation models from a meta-model as input, and using a set of conversion rules. The transformation itself is also a model. Transformation models can have a variety of applications:

• Creating low-level models, and as a result the initial code, moving from models of a higher level.

• Display and synchronization between models at the same level or at different levels of abstraction.

• Creating a presentation about the system on the basis of requests.

• The evolution of model problems, such as refactoring pattern, i.e. change internal system structure in order to make it easier to be understood and further changeable, while not altering the existing functionality.

• Reverse design from the lower level models (or even code) to models of a higher level.

Models conversion is a key element in services system design, which provides ontological processing means for generating conceptual models. Languages of models conversion models support different types of transformations, such as ***model-model (MM)*** or a ***model-code (MC)***. A feature of this approach is that most of the time a developer works not with the code, and with the models.

Ontologies are of great use for providing better human and machine experience/interaction and for software engineering in the new computing paradigm. Ontologies will provide a neutral authoring mechanism where resulting ontologies will be automatically converted into different information and application artifacts.

Model - Driven Approach in Service Systems Design

The ***Model - driven approach (MDA)*** for a system development was suggested, which is a variant of this refinement-based software development in which models are no longer loosely coupled, but connected in a systematic way: that is, model elements must be traceable from a more abstract model to a more concrete model and vice versa [17-23]. This is achieved through meta-modelling: meta-models define sets of valid models, facilitating their transformation, serialization, and exchange. There is the considerable amount of similarity between Ontology-driven and Model-driven approaches that makes really possible to use them both for automated software development based on higher abstraction. In this process, one specific type of model information, ***the platform information***, plays an important role. In MDA, models differ in how much platform information they contain. For instance, one platform can be the programming language of the system, another can be the employed libraries or frameworks, a third can be the binary component model. The designer begins with a high-level model that abstracts from all kinds of platform issues, and iteratively transforms the model to more concrete models, introducing more and more platform-specific information.

MDA uses the MOF-based models for the creation and manipulation of accurate, detailed, computer-readable description of the application structures independent of programming languages, operating systems and databases, which can be used to implement them. It should be noted that a key standard for MDA is the MOF, but not UML, as some people still believe. MDA, based on a three-layer approach, provides:

• ***Computational independent model (CIM)***, which describes a system with independent computing point of view, highlighting the structural aspects of the system, changing the emphasis from a domain modeling to architecture modeling. A CIM is sometimes called a domain model.

• ***Platform independent model (PIM)***, which can be considered as the definition of the system from point of view of a neutral virtual machine or computer abstraction. The platform independent viewpoint focuses on the operation of a system while hiding the details necessary for a particular platform. So it is suitable for use with a number of different platforms of similar type.

• ***Platform Specific Model (PSM)***, which usually covers the technical concepts and services comprising the implementation platform. That is, this model is aimed at a specific implementation services system technology.

A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.

The CIM, PIM, and PSM models live on level M1 (Figure 1). All of them are specified on level M2 by meta-models (CIM-MM, PIM-MM, PSM-MM), dialects of UML, enriching the UML core by profiles containing markup for model elements (stereotypes and tagged values) [24]. A meta-model describes properties and structures of each of the CIM, PIM and PSM models accurately. In our case a platform based on Web services is chosen. Web services options for accessing the service repository as well as protocols for communication between services and customers form a platform of SOA implementation.

In principle the MDA process starts with defining CIM or domain models, then these models can be transformed by professionals to PIM and PSM models. These transformations are called *vertical transformations* (Figure 1). A direct vertical transformation is not always possible because there gaps between the models that are too big to bridge in a single transformation. In these cases, additional transformations are used to, for instance, transform a certain abstract PIM model to a more detailed PIM *(horizontal transformations)*.

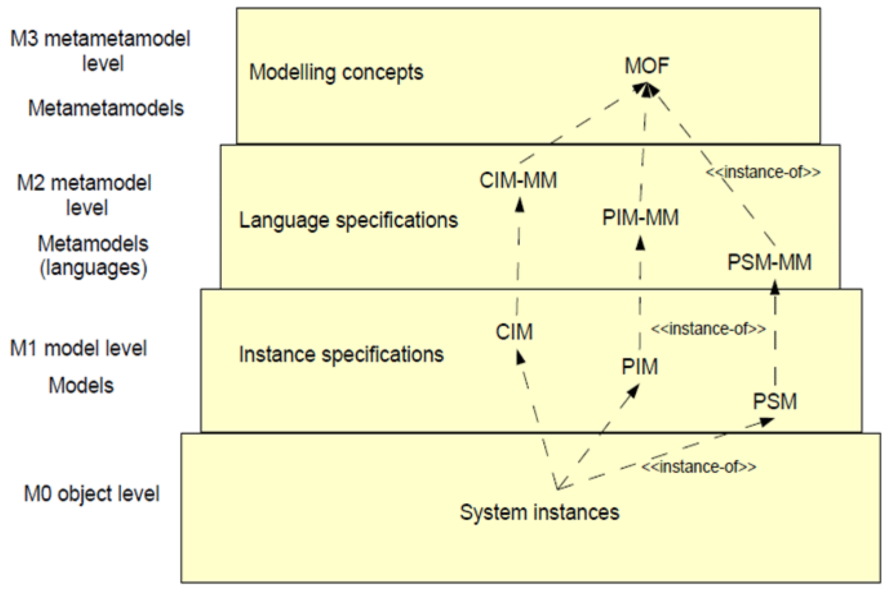

Figure 1: The meta-pyramid with CIM, PIM, PSM models

Various types of platforms may differ. A common feature of the platform is its service-oriented architecture, and the specific technologies of implementation are Web services and service arrangements (for example, Apache Axis, or Oracle BPEL). As it was noted above, metamodelling is one of the most important concepts of the MDA. Ontologies are considered as CIM model, so conceptual models PIM and PSM also reflect the semantics of the application for the domain corresponding to the semantics of real-world domain (Real World), regardless of the specific application needs. The designer starts from standardized analysis models, ontologies, which may have been defined long before project start. These domain and business models are refined towards design models. First, the requirements are added to yield a complete CIM. This is refined to a PIM and, then, conventionally, via several PSMs towards an implementation. Employing ontologies as analysis models should increase the reliability of software products, because these models are well engineered, often used, and hence trustworthy. This avoids the risks of a self-made domain analysis. Secondly, ontologies as analysis models offer a more common vocabulary for the software architect, customer, and domain expert. This should improve the understanding of the parties that order and construct software. Then, the standardization of the ontologies improves the interoperability of applications, because applications that use the ontology contain a common core of common vocabulary. Finally, domain and business ontologies can be reused in many software products. The exchange of meta-data has been simplified by the XMI standard [25]. Essentially, XMI defines meta-model mappings on level M2 between the UML meta-model, XML schema definitions, and a programming language—for example, Java. Based on these mappings, serialization of graph-like UML models to tree-shaped XML models can be automated. Also, Java class models, which use a restricted form of inheritance, can be generated automatically [26,27]. Thus, ontology acts as a meta-model basis to generate a conceptual model for specific information systems.

When Platform-Specific Model (PSM) is created the necessary services are discovered through the network, and the use of ontology improves the accuracy of the discovering such service which corresponds exactly to the customer's request instead opening arbitrary services, descriptions of which met the specified keyword [28]. It is well known that in various subject areas same concepts can be presented in different terms. The mechanism of ontology in these cases allows to create meaningful hierarchical relationships between services, that is, to realize the composition of services, able to satisfy the user's request, even though in its description there are no some of the key words from the input customers' query.

With regard to the development of software applications, model-oriented SOC design procedures are summarized in Table 2.

Table2: Comparison of different approaches to software development

Object-oriented Design	Model-oriented SOA Design
RESPONSIBLE EXECUTORS	
Software developers, most of the time working with the code	*Software architects,* most of the time working jointly with the domain experts in oncology and models
TECHNICAL REQUIREMENTS ANALYSIS STAGE	
The requirements are given in natural language for system analysts who convert their technical characteristics in order to enable developers/ designers to understand them.	Domain analysis is done by the service provider that allows application developers to focus only on discovering and composing services which meet the business/ technical characteristics.
DESIGN STAGE	
Refinement of the class structure, the design of the system architecture (Structure components and modules), the choice of building blocks.	Select or build ontologies of the application domain and application itself, repository services discovering.
IMPLEMANTATION STAGE	
Encoding in a particular programming language (Java, C ++, C #). Development is done by one (virtual or physical) team, which creates functions, classes, modules.	Service models are provided in machine-readable form that allows systems developers to convert them into executable code (CIM-> PIM -> PSM-> UML). New services can be dynamically created using the existing ones.
TESTING AND SUPPORT STAGE	
Testing is usually performed in the same organization on the basis of source code and functional specifications. Test scripts are defined by developers / testers.	Testing is divided between the service provider, the broker and the client with little or no interaction between them. Test scripts can be generated on the basis of service metadata and specifications.
GENERAL FEATHERS OF SOLUTIONS	
Project-specific solutions for selected domain with relatively short using time, which are difficult to reuse.	Cross-project solutions which are serving multiple domains and are shared by many designers and organizations.

Conclusions

The orientation of the world economy on the service industry, the emergence of the interdisciplinary science of services, expanding service approaches to technical systems (in particular, to software structure) motivate economists, sociologists, mathematicians, programmers and lawmakers to work together to achieve a very important goal: analyze, build, manage and development of complex service systems. The main task is to identify the logic of complex service systems and to introduce a common methodology for their modelling and the overall framework for service innovation. Service-Oriented Architecture (SOA) has been successfully deployed as an architectural paradigm for these purposes. The dominance of the Web services platform in distributed systems development and deployment has given a boost to SOA as an approach to software architecture.

In this paper, a formal ontology-based and model-driven approach is presented for developing Service System Applications. In the analysis phase (conceptual modeling) of any software system the emphasis is placed on the data (i.e., in the domain information or CIM model), rather than in the operations (i.e., the behavior). In this vein, the MDA allows representing models of the real-world using conceptual models that abstract key domain concepts, to represent them appropriately and allows transforming them into code correctly.

Considered approach to service systems design is well adapted to the peculiarities of modern distributed computing environments (such multi-clouds or grid), where Web services can be found in various geographically dispersed repositories (possibly in several equivalent embodiments), and their composition as applications are executed on different computing nodes of the environment, resources of which are freed at the beginning of the next execution of service compositions. As a result, the implemented applications have dynamically variable architecture and variable component compositions. So instead of traditional usages of programming languages for specifying ***how a system is to be implemented***, the approach in hands allows specifying ***what system functionality is required and what architecture is to be used***.

A study case of implementing the Model-driven Ontology to the design and execution of mobile application for supporting mHealth needs will be presented in the next paper.

References

1. Maglio P, Kieliszewski CA, Spohrer J. Handbook of Service Science. Service Science: Research and Innovations in the Service Economy. Springer. 2010:754.
2. Succeeding through service innovation: A service perspective for education, research, business and government. University of Cambridge Institute for Manufacturing (IfM) and International Business Machines Corporation (IBM). 2008;ISBN:978-1-902546-65-0:30.
3. Atkinson C, Kühne T. Model-driven development: A metamodelling foundation. IEEE Software. 2003;20(5):36–41.
4. Alahmari S, Roure DD, Zaluska E. A Model-Driven Architecture Approach to the Efficient Identification of Services on Service-Oriented Enterprise Architecture. EDOCW. 2010:165-172.
5. Schmidt DC. Model-Driven Engineering. IEEE Computer. 2006;39(2):25-31.
6. Jürgen Kress, Berthold Maier, Hajo Normann, Danilo Schmeidel, Guido Schmutz, Bernd Trops, Clemens Utschig-Utschig, Torsten Winterberg. Industrial SOA: http://www.oracle.com/ technetwork/articles/soa/ind-soa-toc- 1934143.html.5
7. World Wide Web Consortium. Web Services Architecture. 2006:98.
8. Workflow Tutorial. Available from: http://www.pnmsoft.com / resources/bpm-tutorial/workflow-tutorial/
9. The Solution Designer's Guide to IBM on Demand Business Solutions (ODOE). 2015;Available from: http://www.redbooks.ibm.com/redbooks/pdfs/sg246248.pdf
10. Tsai WT, Yinong Chen, Calvin Cheng, Xin Sun, Gary Bitter, Mary White. An Introductory Course on Service-Oriented Computing for High Schools. Journal of Information Technology Education. 2008;7:315-337.
11. Service Science Knowledge Environment. Available from: http://sske.cloud.upb.ro/sskemw/index.php/Main_Page
12. Semantic Web services (SWS). Available from: https://www.dfki.de/~klusch/i2s/cascom_book_ch3-4.pdf
13. Petrenko AA. Comparing the types of service system architectures (in Ukrainian). System Research and Information Technology, 2015;№ 4:48-62.
14. Petrenko A.A. Objects and methods of the services science (in Russian). System Research and Information Technology, 2015;№ 2:75-82.
15. Ruiz F, Hilera JR. Using Ontologies in Software Engineering and Technology. In Ontologies for Software Engineering and Software Technology. Springer Verlag. 2006:49-102.
16. OMG. MDA Guide V1.0. 2003; Available from: http://www.omg.org/cgi-bin/doc?formal/03-05-01
17. Favre JM, NguyenT. Towards a megamodel to model software evolution through transformations. Electronic Notes in Theoretical Computer Science. 2005;127(3):59–74.
18. María-Cruz Valiente, Cristina Vicente-Chicote, Daniel Rodríguez. An Ontology-Based and Model-Driven Approach for Designing IT Service Management Systems. International Journal of Service Science, Management, Engineering, and Technology. 2011;2(2):65-81.
19. Saad Alahmari, David de Roure, Ed Zaluska. A Model-Driven Architecture Approach to the Efficient Identification of Services on Service-Oriented Enterprise Architecture. Enterprise Distributed Object Computing Conference Workshops (EDOCW). 2010. DOI: 10.1109/EDOCW.2010.28
20. Khoshnevis S, Shams Aliee F, Jamshidi P. Model Driven Approach to Service Oriented Enterprise Architecture. 2009 IEEE Asia-Pacific Services Computing Conference (IEEE APSCC). 2009:279-286.
21. Sinan Si Alhir. Understanding the Model Driven Architecture (MDA). Fall 2003 issue of Methods & Tools. 2013.
22. Enterprise Architect. Sparx Systems. 2008. Available form: http://www.sparxsystems.com/products/ea/trial/request.html

23. Eldein I, Ammar HH. Model-Driven Architecture for Cloud Applications Development, A survey, International Journal of Computer Applications Technology and Research. 2015;4(9):698-705.
24. Uwe Aßmann, Steffen Zschaler. Ontologies, Meta-models, and the Model-Driven Paradigm. In Ontologies for Software Engineering and Software Technology. Springer Verlag. 2006:249-273.
25. OMG. XML Metadata Interchange (XMI). 2002; Available from: http://www.omg.org/technology/ documents/ format/xmi.htm
26. Pahl C, Barrett R. Semantic Model-Driven Architecting of Service-based Software Systems. 28.
27. Kalyanpur A, DJ Pastor, Battle S, Padget J. Automatic mapping of OWL ontologies into Java. Proc of 16th International Conference on Software Engineering and Knowledge Engineering. 2004:98-103.
28. Petrenko IA, Petrenko AA. Automated methods of required services search and discovery (in Ukrainian). Bulletin of the University “Ukraine”, Series “Information, Computing and Cybernetics”. 2015;№1 (17):55-64.