Journal of Software Engineering and Applications, 2015, 8, 324-338
Published Online July 2015 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2015.87033

Scientific Research Publishing

Ontology of Domains. Ontological Description Software Engineering Domain—The Standard Life Cycle

Ekaterina M. Lavrischeva

Moscow Physics-Technical Institute, Dolgoprudnuy, Russia
Email: lavryscheva@gmail.com

Received 12 May 2015; accepted 21 July 2015; published 24 July 2015

Abstract

Basic concepts and notions of ontological description of domains are implemented in the conceptual model being understandable to ordinary users of this domain. Ontological approach is used for the presentation of software engineering domain—Life Cycle (LC) ISO/IEC 12207 with the aim to automate LC processes and to generate different variants of LC for development systems. And the second aim of Conceptual Model must teach the student to standard process LC, which includes general, organizational and supported processes. These processes are presented in graphical terms of DSL, which are transformed to XML for processing systems in the modern environment (IBM, VS.Net, JAVA and so on). The testing process is given in ontology terms of Protégé systems, and semantics of tasks of this process is implemented in Ruby. Domain ontology LC is displayed by the several students of MIPT Russia and Kiev National University as laboratory practicum course "Software Engineering".

Keywords

Ontology, Life Cycle, Models, Processes, Actions, Tasks, Testing, DSL, XML, Protégé

1. Introduction

At the given work, new conception of automation in general processes of LC and generation of variants specialized are offered for their use in the modern programs, information systems and technologies and implementations in the distributed environments of Grid and Clouds processing, highly productive cluster systems and in web-semantic to the Internet.

This conception is formulated by the author for the students of MIPT and Kiev National University (KNU) at

How to cite this paper: Lavrischeva, E.M. (2015) Ontology of Domains. Ontological Description Software Engineering Domain—The Standard Life Cycle. *Journal of Software Engineering and Applications*, **8**, 324-338.
http://dx.doi.org/10.4236/jsea.2015.87033

the basic course lections of “Software Engineering” (2010-2013). Standard LC ISO/IEC 12207-2007 is a general mechanism of construction of various program systems (PS) and program products (PP). The 17 processes enter into his composition, 74 under processes (actions) and 232 technological tasks. The automation of LC is a very thorny and heavy problem. Variants of the standard LC will be implemented by many companies in case of development of the different application systems. A submachine gun is absent. Offered by our conception of LC automation through the formal conceptual model LC is an attempt in development of the Case commons instruments for support LC to the future industry PP [1]-[5].

In addition, for implementation of this conception we use new languages of description in conceptual models of knowledge: OWL (Web Ontology Language), ODSD (Ontology-Driven Software Development), XML (Extensible Markup Language), MBPN (Modeling Biasness Process Notation) and others like that. There are systems of design of domains—ODM (Organizational Domain Modeling), FODA (Feature-Oriented Domain Analysis), DSSA (Domain-Specific Software Architectures), DSL (Domain Specific Language) Tools VS.Net, Eclipse-DSL, Protégé and others like that. That is terms are used for the formal specification of the LC processes and design from them of different PP. The ontological approach ODSD allows getting descriptions of classes from notions to the domain. Unlike previous, models to the domain can be used not only for the generation of code, but also can be “executable” artifacts.

An important aspect of design in different domains is the notion base and system of notions, by which all problems are formulated to the domain. The notion base is given by terminology, substantial relations between notions and their interpretations.

Among the relations the main are [1]:

–Concretization, as an union of notions in the new notion, the substantial signs of which can be a sum of signs of notions or substantially new;

–Association, that approves a presence of communication between notions without clarification of dependence of them from maintenance and volumes;

–Aggregation of terminology, notions, characters for their relations and paradigms of their interpretations in scopes to the domain is accepted to name the ontology of domain knowledge.

In the same general case, ontology is an agreement about the general use of notions, which includes facilities of subject knowledge and agreement about methods of reasoning representation.

The ontology appears by the semantic networks, knots of which are been by domain notion, and by arcs—copulas or relation, associations between them. On the given time the ontological approaches got the wide distribution in the decision of problems of knowledge, semantic integration of informative resources, informative searches and others like that representation. They allow getting descriptions of classes of objects domain, which are specific at notion and knowledge about them. Some ontology domains are given by knowledge, dictionaries of notions, concepts and relations between them. So, the XML become a standard language for marking of various data domains for their saving and exchange between different domains. It is a mean of automatic transformation of descriptions of model domains in the modern ontological languages to the charts, which are suitable for work in the different applied applications.

Offered conception ontology was considered in the different models of LC (spiral, interaction, incremental and so on) on student lections and on the scientific seminars of the Theory Programming and Information Systems departments of the Kiev National University (KNU), and also case of the discipline teaching the “Software engineering” [3]. Within the framework of this discipline the students learn modern methods and design facilities domains and PS constructions, and also learning standards of LC ISO/IEC 12207-2007 and General Data Types ISO/IEC 11404 GDT-2006. At the practical classes evaluation of facilities is conducted for general description and implementation of some experimental ontology on DSL Tools VS.Net and Protégé. For example of description of some fragments of science domains by the ontological facilities they built models PS with purpose of their use in case of PP construction [4] [5]. Some students of department of the IC and TP faculty of cybernetics defended diploma works on given topics with the use of ontological facilities and notions (classes, axioms, slots, facets and others like that) for description of calculable geometry, GDT and LC on the ITC of developing object and component and configuration them [6]. (http://sestudy.edu-ua.net)

Approach to LC automation and its evaluation by students in curriculum of the model LC and standard ISO/IEC 12207-2007, namely to study LC structure, processes and actions, and also use of ontology facilities for their description and implementation in the open ontological instruments are offered—DSL Tools VS.Net and Protégé. Students received ontological knowledge may apply them in the implementation of other application areas.

2. Ontology as a Basic Formal Description of Subject Areas

Ontology is a conceptual tool to describe base set of concepts and relations for some domains (or subject **area-SA**). The concept of the SA is classified and dictionary and thesaurus database schema knowledge is created. Domain Ontology—is a system of concepts or conceptual model which is supplied with a set of entities and relationships between them. Now, many anthologies for various scientific and applied areas are created. For example: ontology Census general knowledge of English natural concepts (70,000 more terms and their definitions); ontology concepts of e-commerce; global ontology products and services (UN); commercial ontology SCTG, Rosetta Net-traffic products from 400 companies. Medical ontology's: Galen-to determine the clinical condition; UMLS-US National Library of Medicine; ON9-famous for certification of health systems; chemical, biological ontology; all-Web portal mathematical resources; universal mathematical system Math Lab, Ret, etc.

A basic instrument of implementation of the subject description is the DSL Tools VS.Net and Eclipse-DSL [7]-[9], the result of the tools is described in the XML (Extensible Markup Language) language, which actually became a standard of data marking for their saving and exchange of information between different applications. XML serves to transformation of the domain ontology model in XML-charts, suitable for work of applied applications.

Using the properties ontology for description of processes LC was given the subject-oriented DSL (Domain Specific Language), and also language the BPMN description of semantics of these processes, the author offers approach of implementation of suggested conception. In the example implementation of the given conception we select the process of testing by the ontological facilities and semantics description by language of programming. At developments to the domain LC used ontological instrumental facilities, DSL Tools VS.Net, DSL Eclipse and Protégé. Set of different methodologies, facilities of language for description of domains are shown in **Figure 1**.

Ontology of LC is absent. We consider two means of implementing domain LC-language OWL and tools DSL for the overall presentation of the model standard ISO/IEC 12207 and basic concepts of the testing process LC in the standard system Protégé.

Ontology Means

The form of representation of ontology is a conceptual model (CM) on the reflecting system concepts with common properties (attributes), attitude and behavior rules. CM serves as a communication (between people, between computer systems), storage of information in a computer environment and the recycling of finished objects stored in libraries and repositories. To describe the use of ontology language OWL (Web Ontology Language) with a range of languages and markup languages RDF is to access and exchange ontological knowledge in the Internet. Description in ontology language OWL is a sequence of axioms and facts, information about classes, properties and resources for ID documents and Web imported URI in the form: <DatatypeID>:: = <URI reference> (see **Figure 2**).

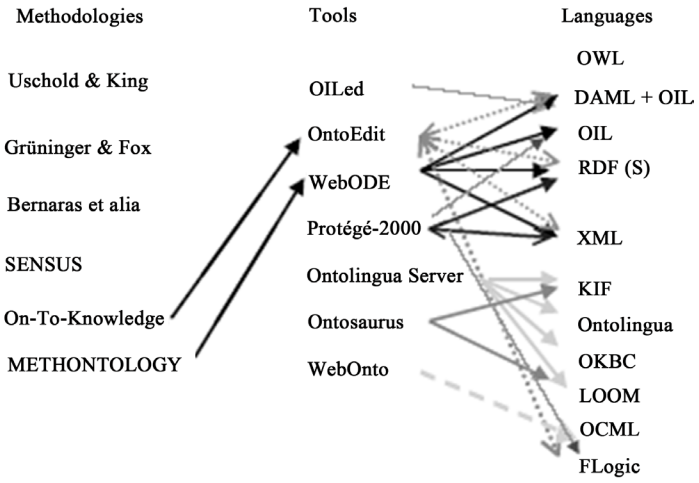

Figure 1. List of methodologies, facilities and the ontology language.

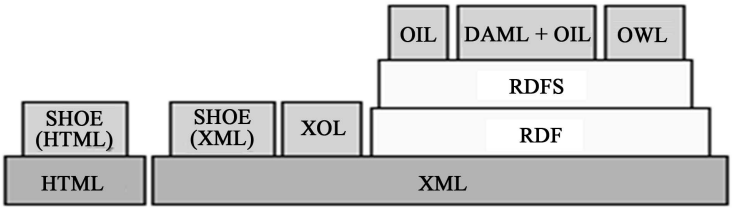

Figure 2. Languages definition of ontology.

Axiom class is a set of more general classes and restrictions on local properties of objects.

The class or a subset of the intersection of more general classes of constraints may be equivalent.

Axiom class in OWL is a set of specifications that can be in the form of generalized classes, restrictions, sets of resources, Boolean combinations of descriptions and more.

As ontology editor was used Protégé 3.4. It seems ontology classes, slots, facets and axioms. (http://protege.stanford.edu/)

Classes describe concepts of, and slots—properties (attributes) classes.

Facets describe the properties of slots (specific types and ranges of possible values).

The axioms define additional constraints (rules).

Classes can be abstract or concrete.

Abstract classes are classes and concrete containers may contain abstract attributes (which do not contain specific values).

Attributes of concepts in a domain called Protégé slots. Specific classes contain specific slots, which can be assigned a value (a copy attributes). To determine the types and limits on the value (like the rules of XML-Schema) used facets.

Protégé supports multiple inheritances but a class can have more than one superclass.

3. Life Cycles Ontology of Software Systems

LC received evolution from the beginning of programming, from simplified life cycle for each application models to spiral, iterative and so on. They formed a separation in group development of various types PS. As the result the standard ISO/IEC 1996 (first edition) was introduced, and in 2007 the second edition of its life cycle appeared, which reflects the overall structure of processes that may be involved in the development of the different PS. These standards should be studied by students who will participate in the joint development of various applications and commercial systems [3].

3.1. Presentation of Formal Specification of the LC Standard ISO/IEC-12207

The LC processes are given in standard by three categories (see **Table 1**).

In the each process it defines types of activity (actions-activity), tasks, aggregate of results (going out) of activity and decision of tasks design, testing, assembly and others, and also tracing some specific requirements. A list of works for the basic, organizational and support processes is led in standard, but method of their implementation and form of presentation not available. Next, we give a general description of the basic, organizational and support processes.

To the *basic processes* belong:

–Acquisition process determines actions of buyer at automated system or service. Actions are initiation and preparation of query, legalization of contract; monitoring and acceptance;

–Delivery process determines actions from the transmission of product or service to the buyer. It has preparation of suggestions, legalization of contract, planning, implementation and control of product, and also its estimation and delivery;

–Development process determines the processes and actions (development of requirements, planning, encoding, integration, testing, the system testing, and installation) for development of PP;

–Exploitation process (introduction, support of user, testing functions, exploitation of the system) determines actions of operator from maintenance of processes the system during its exploitation by users;

–Maintenance process (management by modifications, support of current status and functional fitness, PP installation in the operating environment, accompaniment and modification, development of system modification

Table 1. The process is of standard life cycle.

№ п/п	Process (subprocess)	
1. Category the "Basic processes"		
1.1	Order (agreement)	
	1.1.1	Preparation of order, choice of supplier
	1.1.2	Monitoring supplier activity, acceptance by user
1.2	Delivery (acquisition)	
1.3	Development	
	1.3.1	Exposure of requirements
	1.3.2	Analysis of system requirements
	1.3.3	Planning system architecture
	1.3.4	Analysis of system requirements
	1.3.5	Planning the system
	1.3.6	Constructing (code) the system
	1.3.7	Integration of the system
	1.3.8	Testing the system
	1.3.9	System integration
	1.3.1	System testing
	1.3.1	Installation of the system
1.4	Exploitation	
	1.4.1	Functional application
	1.4.2	Support of user
1.5	Accompaniment	
2. The support "Processes category"		
2.1	Documenting	
2.2	Management by configuration	
2.3	Providing a quality guarantee	
2.4	Verification	
2.5	Validation	
2.6	General review	
2.7	Audit	
2.8	Decision of problems	
2.9	Providing product applicability	
2.10	Evaluation of product	
3. Category the "Organizational processes"		
3.1	Management	
	3.1.1	Management at level organization
	3.1.2	Management by project
	3.1.3	Management by quality
	3.1.4	Management by the risk
	3.1.5	Organizational providing
	3.1.6	Measuring
	3.1.7	Management by knowledge's
3.2	Improvement	
	3.2.1	Introduction of processes
	3.2.2	Evaluation of processes
	3.2.3	Improvement of processes

plans, PP migrations on other and others like that), which determines actions of organization, that development PP.

The LC standard contains description of the ancillary proceeding, that regulate the additional actions from verification of product, management by project and his quality.

The *support process* contains: documenting, management by versions, verification and validation, revisions, audits, evaluation of product and etch. To the *organizational processes* belong: management by project (development management) and perfection of processes.

The *management process* includes the processes of management by configuration, project, quality, risks, changes and others like that. The *perfection* process includes introduction, project estimation and his perfection.

Quantity of processes, actions and LC standard tasks are shown in **Table 2**.

Depending on the purpose of concrete project the main developer and project manager choose the processes, actions and tasks, line up the LC chart for application in the concrete project. Description of semantics of processes and methods of their implementation (objective, component, service and so on) written in kernel of SWEBOK knowledge and [3]. (www.swebok.com)

Theoretical, applied methods, quality standards, general and fundamental types of data (ISO/IEC 15404, ISO/IEC 9126, ISO/IEC 11404 and others), and also recommendations and methods of this standards are used at every technology of the PS programming with the use of the LC standard.

Task of automation of standard LC arose up in the students groups MIPT and KNU of course Software Engineering. Taking into account this task, the author discussed with the students the features of standards and machineries of their presentation in the modern operating environments. On the practical lessons the students learned LC processes and gave their description for DSL Tools VS.Net. The students executed LC ontology description in graphic (**Figure 3**) and the XML kinds within this framework. Then they used DSL Eclipse and Protégé.

XML description of general, support and organization processes are given on web-site ITC.

3.2. Formal Presentation of Conceptual Model Domain LC

Starting from **Table 2**, we give description to the conceptual model (CM) domain LC standard, described highly from terms: P—processes, A—actions and T—tasks.

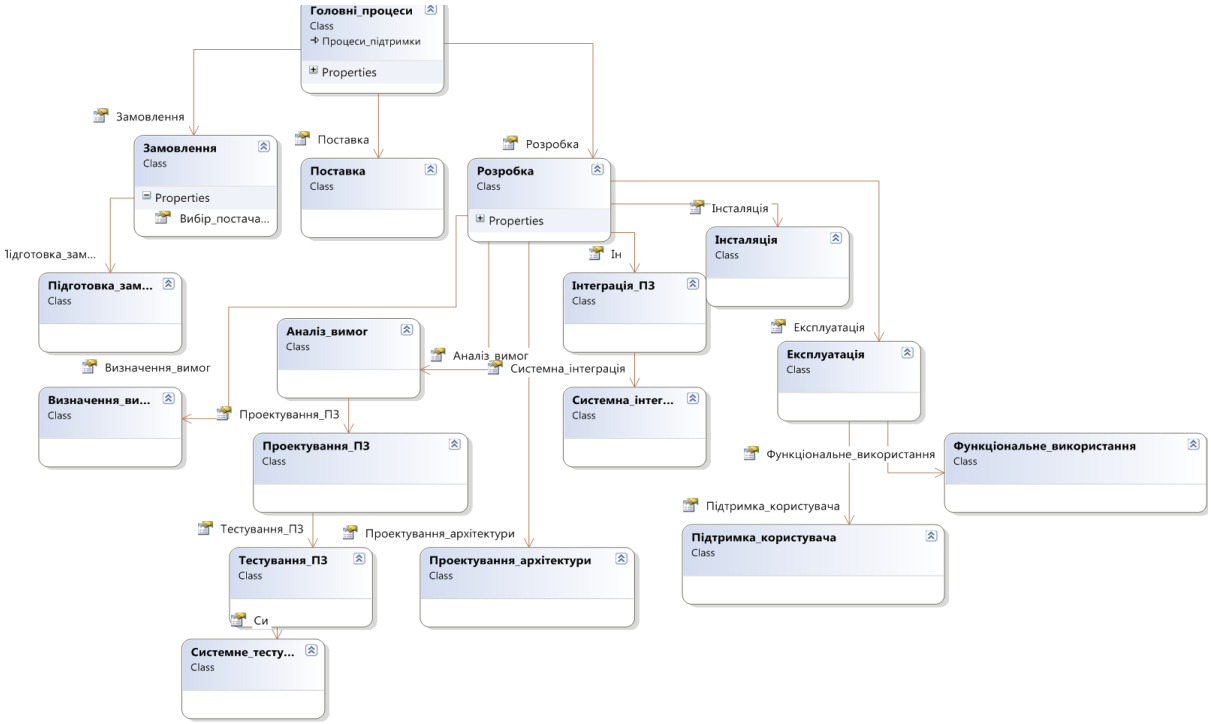

Figure 3. Graphical representation of the basic life cycle processes.

Table 2. Processes, under processes tasks and LC actions.

Classes	Process	Action	Task
Basic processes	5	35	135
Support processes	10	25	70
Organizational processes	2	14	27
All	17	74	232

The LC model has such kind:

$$M_{\text{LC}} = \left(P_{\kappa}, A_{\text{м}}, T_n\right)$$

where $P\kappa = \left(P_{1\kappa}, P_{2\kappa 1}, P_{3\kappa 2}\right) - P_{1\kappa,\kappa\ =1-5}$ —basic processes of the LC first category,

$P_{2\kappa 1,\ \kappa 1=1-8}$ —processes of LC support of the second category;

$P_{3\kappa 2,\ \kappa 2=4}$ —organizational processes of the LC second category;

$$A_{\text{м}} = \left(A_{\kappa r}, A_{\kappa 1l}, A_{\kappa 2\,j}\right)$$

where $A_{\kappa,\ r\ r=\ 1-35}$ —action on the basic processes LC,

$A_{\kappa 1l,\ l=1-25}$ —Action on the LC support processes,

$A_{\kappa 2\ j=\ 1-14}$ —action on the processes LC;

$T_n = \left(T_{n\kappa}, T_{nl}, T_{nj}\right)$

where $T_{n\kappa},\ \kappa = 1 - 135$ —tasks of the basic processes LC;

$T_{nl,l=1-70}$ —tasks of LC support processes;

$T_{nj,\ j=1-27}$ —tasks of the organizational processes LC.

The goal processes, operations given is highly contained in essence, and description of maintenance of tasks on it is led them in standard. The tasks not formal and will be in the future at the first given description of their setting, then selection languages for the formal specification for realization to their semantics.

For presentation the structure of the CM LC is used graphic language DSL. This language has an expressive feature, directed on the reflection of the process specific LC, while languages of the general setting (Java, C++, C#, Ruby and others) oriented on description of actions of any programs of the data processing. The DSL contains general abstractions for the reflection of classes of objects domain type process, action, and also relations between them [2] [9]. On its maintenance this language near to the HTML, XML, WSDL and others like that.

Model LC it is described by one DSL, can be transformed in model by other DSL. It allows freely to integrate between itself different parts of system processes, written in the different DSL. That is domain LC can be described at one level of abstraction, and then regenerate with the additional going into detail on the more low level of abstraction, that allows complementing a model domain by the repeated components and objects. Main to the CM domain LC there is a model of general descriptions of processes as objects domain.

Processes of transformation of the LC models in the DSL at the different levels are given in **Figure 4**.

Transformation of description of the models LC in this language DSL is conducted by facilities of the model-guided development MDD (Model Driven Development). According to this model system architecture are designed at two levels—platform of level independent on the PIM (Platform Independent Model) model and platform of dependent level on the PSM model (Platform Specific Models).

The LC domain CM model can be automated with the use of specific languages, be tuned especially for processes and actions, which are in class language ontology. The models can contain information about the union of processes and actions, including artifacts, which participate in it, and also their dependence between itself. They can also contain information about the configuration structure of the programs of treatment of processes, vehicle and program resources, necessary in case of implementation of the programs of automation of processes and their development.

3.2.1. Ontology of Domain Characteristic Model

DSL development pre-condition is made by the detailed analysis and structured to the domain. Among the existent methodologies of domain analysis most knowing such: ODM (Organization Domain Modeling), FODA

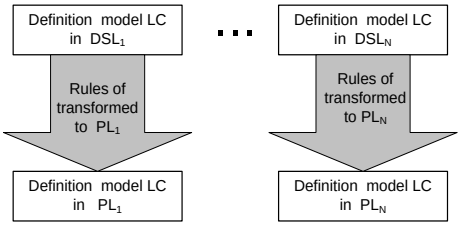

Figure 4. Transformation of description of models LC in DSL.

(Feature-Oriented Domain Analysis), characteristic analysis to the domain and DSSA (*Domain*-Specific Software Architectures) [9]-[11].

In case of analysis to the domain is created a model of characteristics. This model secures *generalization* and *disagreements of the PS* domain processes by the indication of general characteristic for all processes and excellent characteristic each of the LC processes.

A model of characteristic is given by *diagrams of characteristic* with description of relations between them. Conception of diagrams is inherited from the FODA method, which gives possibility briefly to describe all possible configurations of processes within the limits of different categories of the LC processes, which are considered as instances, selecting general and alternative characteristics, which can be excellent for each configuration of the LC processes.

For the given time notation of characteristic diagram is executed by the DSL language under the FDL (Feature Definition Language) name, as languages of description of characteristic of notions to the domain and formal definite operations for treatment of FDL expressions.

The diagrams of characteristic is given system characteristic the different domains. In case of creation of automated instruments, intended for construction of diagrams of characteristic and their treatment, text presentation is necessary. It inflicts all information, which exists in the graphic diagram. The determination consists of great number of characteristic (feature definitions), names of characteristic and *expression* (feature expression), that includes:

–Atomic characteristics;

–Composition characteristics: names of which determination elsewhere;

–Optional characteristics (*optional*) of expression, is it completed by the “character”?

–Obligatory characteristics (*mandatory*) of expressions, what reserved in construction of *all* ();

–Alternative characteristic (*exclusive*-choice): y expression of *one-of* ();

–Exceptional set of descriptions (“or-features”) from the list of characteristic expressions of *more-of* () and their combination;

–Value of characteristic by default (default)—atomic to description;

–Other (indefinite) characteristic in the form of “.”.

The specification of FDL characteristic gives formals for determination of syntax, which it is possible to compare to the BNF (P.Naur) form for conducting a lexical and semantic analysis of described characteristic of model domain, which is used for creation of the different variants PS.

3.2.2. About Machineries of Dependence of Characteristic

Offered approach is contained on principle of inherences characteristic with such terms:

– Every characteristic answers class;

–*Associations* (copulas) between classes are noticed to so call <stereotype>, which marks a type of characteristic dependence:

–Obligatory (*mandatory*) dependence between aggregations in classes;

–Optional (*optional*) dependence between association and range of cardinal numbers (by power of great number or quantity of elements of great number) from 0 до1;

–Obligatory list of *one-of* and *more-of* in specified class each of alternatives.

The result of translation of description of characteristic in FDL can be given by the XMI language, as a format of exchange by information of Meta data (the XML Meta data Information Exchange format). The XML–documents can be imported in the UML design instruments, such as Rational and UML, and also for the generation of the Java classes. After creation of the DSL language to the domain it is necessary to use the FDL language. Approach to description of model of domain it is used for developed the LC processes of variants PS

by configuring different processes for automation the PS. On the given model LC are solved the task of providing a generation of special variants from necessary processes for realization of the set PS. Every variant will be addition of semantics of some tasks for included processes. For the receipt of the working variant LC PS a use of Java facilities is planned [6] [7].

3.3. Standard Life Cycle Ontology in DSL Eclipse

For description of ontology of domains there is other approach of Eclipse DSL [3] [6]. This development environment is used for presentation of the graphic models LC because it has effective instruments for description the object of this domain. On beginning it is necessary to develop a visual model of domain LC. Than it make description of classes of sections processes LC domain and relations between them (**Figure 5** and **Figure 6**).

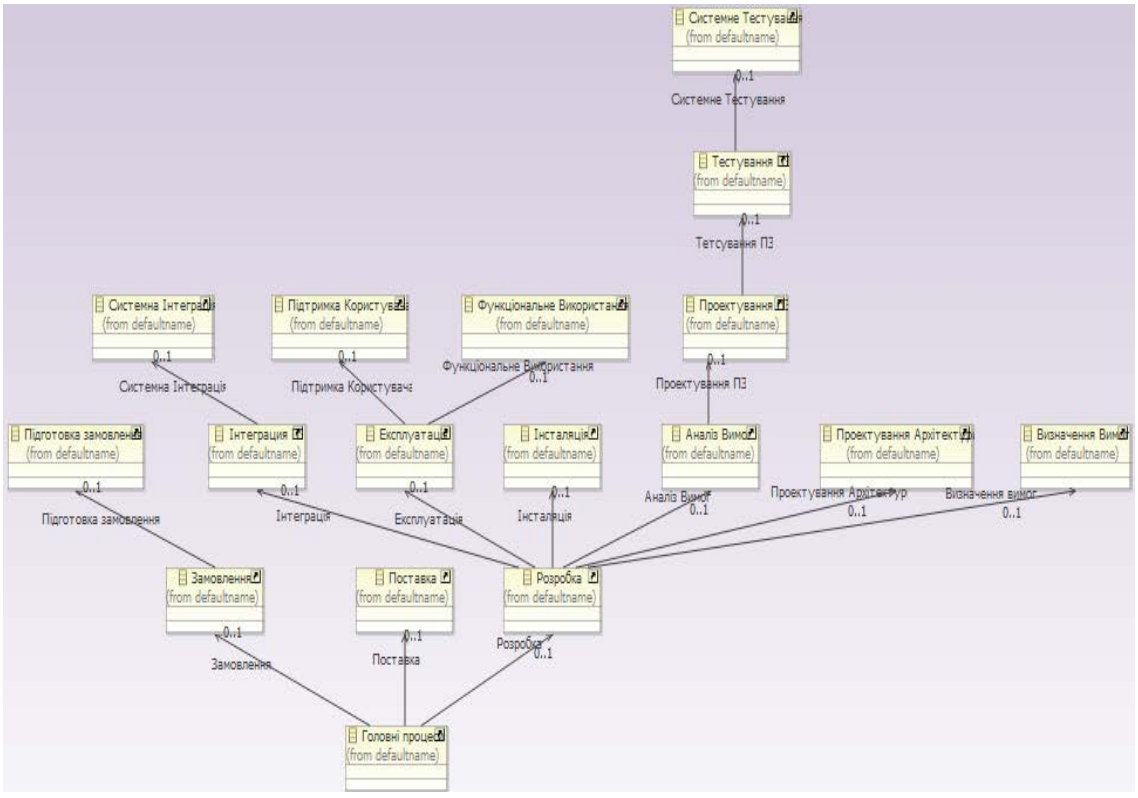

Figure 5. Structure organization process LC in DSL.

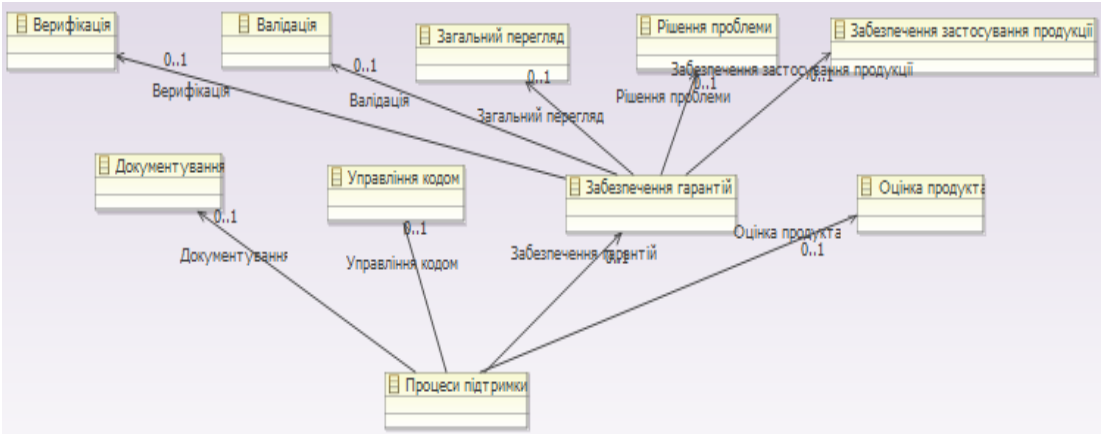

Figure 6. Ontology of the basic processes LC in DSL.

The types of relations allow realizing basic logic of project. Present methods and fields necessary are described in every class for functioning a project. The support processes contain all processes that are executed after the domain construction and support his capacity and actuality. Their ontological structure answers a structure of basic processes and is pointed it will not be.

A next step is been by the generation of text presentation of present graphic models, and then generation in XML. A process of the LC testing is annotated by facilities of knowledge domain notion and their relation in Protégé representation.

Text description of the LC processes by XML

Given graphic presentation of the CL processes was used for the receipt of text in the XML.

Errors in the graphic description, which were found by designer and correction by, correspond to the editor. After it a result of every process is given in XML. An example is below led to the description fragment to the fragment of the main processes LC in the XML.

```
<? Xml version="1.0" encoding="utf-8"?>
   <Association Line Name="Determination of requirements" Type="Main. Determination of requirements"
Manually Routed = "true" FixedFromPoint = "true"   </Association Line>
   <AssociationLine Name = "Integration PS" Type = "Main. Integration PS"
   <AssociationLine Name = "Installetion" Type = "Main. Installetion"
   <AssociationLine Name = "Exploatation" Type = "Main. Exploatation"
     <Property Name = " Integration PS" />
     <Property Name = " Installetion" />
     <Property Name = " Requirements Analyses" />
     <Property Name = "Exploitation" />…
```

For receipt descriptions of the LC processes in XML is given their semantic description. Annotating is executed on example of the LC testing processes by Protégé facilities [10] [11].

Facilities of the Protégé for Description Ontology

To the basic facilities of the Protégé for description of ontology belong:

–classes (or notion);
–relation (or properties, attributes);
–functions;
–axioms,
–copies (or individuals).

Classes—it is abstract groups, collections or sets of objects. They can include the copies, other classes, or halving both that and second. *The relations* give a type to co-operation between notions domain. *Functions*—it is the special case of relations, in which an n-element of relation is simply determined by the n-1 previous elements. *Axioms* are used for determination of complex limitations on the value of attributes, arguments of relations, for verification of information correctness, or for inference of new information.

By these facilities Protégé forming an ontological model to the LC domain is conducted.

The classes answer the types of artifacts, which, in same queue, answer the roles of program components in system and in the functional properties product/Classes are reflected in Protégé as an inheritance (inheritance hierarchy) hierarchy, which is disposed in to the window navigator of classes (Class Browser). By root of tree of classes in Protégé, by default, appointed class THING (thing, something). All created classes are to be inherited immediately or mediocre.

The protégé will be use for presentation CL testing processes.

It are a new type of description LC and testing processes, which are very necessary for e-learning students for practice preparing some tests for testing the programs [3] [5] [11].

4. Description of Ontology of Process Testing LC

The conceptual model of process testing of the PS has a kind [4] [5] [7] [8]:

$$SFT = \{TM, TD, TA, Env\}$$

where *TM*—subprocess of management by testing;

TD and *TA*—subprocesses of testing accordingly domains and applications;
Env—conceptual and informative environment of testing process of the PS.
To all three subprocesses will give the compatible formal presentation:

$$TM = \{Task(TM, TD, TA), CM(TA,\ TD), Env\}$$

where *Task*—tasks of correspond under process;

$$Env = En(TM) \cup En(TD) \cup En(TA)$$

En—conceptual and informative environment of correspond under process;
CM—under model of co-ordination of operations of correspond under process.
Environment composition is determined by expression:

$$Env = TG \cup SG \cup T \cup P \cup RG \cup RP$$

where *TG* and *SG*—test active voices and prepared programs;
T and *P*—tests and application for testing;
RG and *RP*—reports about implementation of the tests of programs.

Ontological description of testing process. For description of this process used ontological system Protégé. In her knowledge about the process model are given by *classes*, *slots*, *facets* and *axioms*. Similar possibility give also and other instruments. For example, diagrams of classes in the UML Rational Rose, which can translate in the program code of a few languages of programming.

For presentation of testing ontology use two groups of notions: *simple* and *complex.*

Testing—simple notions. It such: Tester (*Tester*), Context (*Context*), Action (*Activity*), Method (*Method*), Artifact *and Environment*.

Can have simple notion attributes. In quality *attributes* such are selectable under notions, which characterize base (paternal) notion and can accept the concrete values. Will give short maintenance of basic concepts.

Tester—the subject or object, which executes testing determines. The group of testing has a leader, which is a notion attribute, and him name—by the value of attribute. Attributes are been by *name, type, duties*. Tester attribute-duties-describes, that can do a tester in the process of testing. Notion duties–complex notion, which is determined on the basic of simple notions. For this notion it is possible to select next attributes: tester name, tester type, duties. Tester attribute-duties-describes actions, which can be done by tester in the process of testing.

Example to the tester XML-fragment:

```
<! -- TESTER -->
<xs: element name = “TESTER”>
  MaxOccurs = “unbounded” ref = “TESTER”/>
    </xs: sequence>
    <xs:attributename = “TESTER_TYPE” use = “required”>
    <xs:simpleType>
        </xs:attribute>
    <xs:attributename = “TESTER_NAME” use = “required”/>
```

C**ontext** determines the proper levels, methods of testing, entrances and going out tasks of testing. In ontology this notion determines one attribute: Context type (Level of testing) on form

Level of testing = {module, integration, system, regressive}.

Action consists of notions, that go into detail the steps of process of testing: planning testing, development (generation) of tests, implementation of tests, estimation of results, measuring test coverage, generation of reports and others like that. For this notion one attribute is inflicted-type of action (Activity type) with the possible values: type of action = {planning, development of tests, implementation of tests, and verification of results, coverage estimation, and preparation of report}.

Method—this notion, which is answered by a few methods of testing. For example, the module testing—methods of the structural and functional testing. Every method in relation to the initial code can be classified as a “white small box”, “black small box” or based on specification of testing (specification-based). Fragment of method notion XML–chart:

```
<!-- METHOD -->
<xs:element name = “METHOD”>
 <xs:complexType>
    <xs:attribute name = “METHOD_NAME” use = “required”>
     <xs:simpleType>
       <xs:restriction base = “xs:token”>
          <xs:enumeration value = “CONTROL_FLOW_TESTING”/>
          <xs:enumeration value = “DATA_FLOW_TESTING”/>
          <xs:enumeration …
```

The methods based on coda subdivide on: structural; over seeding of errors; mutational. The structural methods subdivide on: testing a stream of management and testing a data flow. The methods of testing a management (control-flow methods) stream include coverage of operators, coverage of branches and different criteria of coverage of ways. This concrete methods testing is copies of different class of methods of testing.

By the similar rank methods of “black small box” are classified or based on specification: functional; on supposition about the errors; heuristic and so on.

From other side, in relation to process of search of errors and refusals, it is possible to divide all methods into systematic (search of errors) and stochastic (statistical)–exposure of refusals.

Thus, for description of method of testing will enter next attributes:

–Name (method name), “laying out on category”;

–Method type (structural, based on the errors);

–Approach, based on code, on specifications, statistical.

Such lying out of methods allows simply classifying every method of testing and extending ontology.

Artifacts. Every action from testing can include a few artifacts, such as an object of testing, intermediate data, results of testing, plans, sets of tests, scripts and others like that. Name them “test active voices”. The objects of testing can be different types: initial code, the HTML files, the XML files, built-in images, sound, video, documents and so on. All this artifacts mapping in ontology. Every artifacts is also associated with place of its saving, data, history of creation and revision (for example, creator, upgrade time, version number and others like that).

Environment. Program environments, where testing is executed, as a rule notion such given: name, type and product version. Given notion is broken up on two under notion: vehicle and program with attributes. Attributes of hardware environment are: *device name, model, and producer*.

Attributes of software environment are: *product name, product type and version*. The possible values of attribute can be seen by such:

Environment = {*ОС*, *БД*, *Compiler*, *web-browser*}.

Complex notions of process of testing. Such belong to the complex notions: tester (capability) duties and task (task). They are determined by simple notions.

In the distributed system co-operation between components is executed by interfaces (reports). After treatment of report, the component which got him returns the *answer*. That is why in ontology expediently to enter additional notions of report and answer. With every report it is possible to link the attributes *Type* and *Value*. With every answer it is possible to link its state, which is set as an attribute with two possible values:

The State answer = {*Success*, *Refusal*}.

Tricking into result to description of basic terms ontology of testing it will present an ontological model of process of testing with the use of led notions is given on the **Figure 7**.

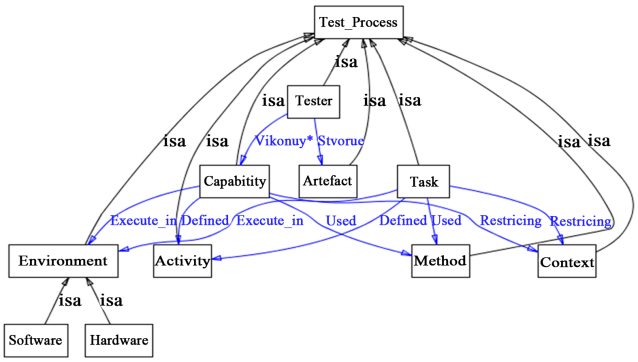

Figure 7. Count of ontology of testing.

For transformation this count to XML gets out format in windows Protege system (**Figure 8**).

Fragment of ontology of testing in the XML, automatically generates Protege 3.4 and is presented in the UNICODE terms:

```
<class>
        <name>: STANDARD-SLOT</name>
        <type>: STANDARD-CLASS</type>
        <own_slot_value>
                <slot_reference>: ROLE</slot_reference>
                <value value_type="string">Concrete</value>
        </own_slot_value>
        <superclass>:SLOT</superclass>
        <template_slot>: DOCUMENTATION</template_slot>
        <template_slot>: SLOT-CONSTRAINTS</template_slot>
        <template_slot>: SLOT-MAXIMUM-CARDINALITY</template_slot>
        <template_slot>: SLOT-MINIMUM-CARDINALITY</template_slot>
        <template_slot>: SLOT-NUMERIC-MAXIMUM</template_slot>
        <template_slot>: SLOT-NUMERIC-MINIMUM</template_slot>
        <template_slot>: SLOT-INVERSE</template_slot>
        <template_slot>: SLOT-DEFAULTS</template_slot>
        <template_slot>: SLOT-VALUES</template_slot>
        <template_slot>: ASSOCIATED-FACET</template_slot>
        <template_slot>: DIRECT-SUBSLOTS</template_slot>
        <template_slot>: DIRECT-SUPERSLOTS</template_slot>
        <template_slot>: DOCUMENTATION</template_slot>
        <template_slot>: SLOT-CONSTRAINTS</template_slot>
        <template_slot>: SLOT-MAXIMUM-CARDINALITY</template_slot>
        <template_slot>: SLOT-MINIMUM-CARDINALITY</template_slot>
        <template_slot>: SLOT-NUMERIC-MAXIMUM</template_slot>
        <template_slot>: SLOT-NUMERIC-MINIMUM</template_slot>
        <template_slot>: SLOT-INVERSE</template_slot>
        <template_slot>: SLOT-DEFAULTS</template_slot>
        <template_slot>: SLOT-VALUES</template_slot>
        <template_slot>: ASSOCIATED-FACET</template_slot>
        <template_slot>: DIRECT-SUBSLOTS</template_slot>
        <template_slot>: DIRECT-SUPERSLOTS</template_slot>
```

Using these notions, the KNU students on the practical getting by busy created a variant of ontology of process of testing and realization of the program of testing by the Ruby (**Figure 9**). Erroneous characters are marked on the checked program (**Figure 9**) on the right, and a correct record is on the left given.

Model of testing ontology and this program made two students in magister works and are placed on web-site http://sestudy.edu-ua.net. It is necessary to do the appeal to, which by pressing on the name "Ontology" word at the main panel of this site.

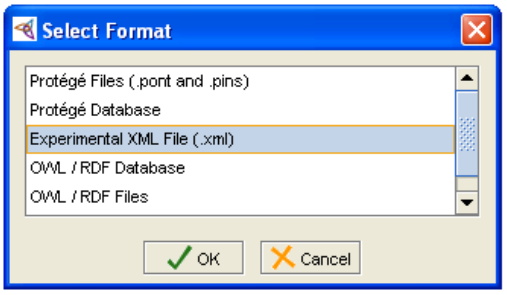

Figure 8. Saving ontology in the XML format.

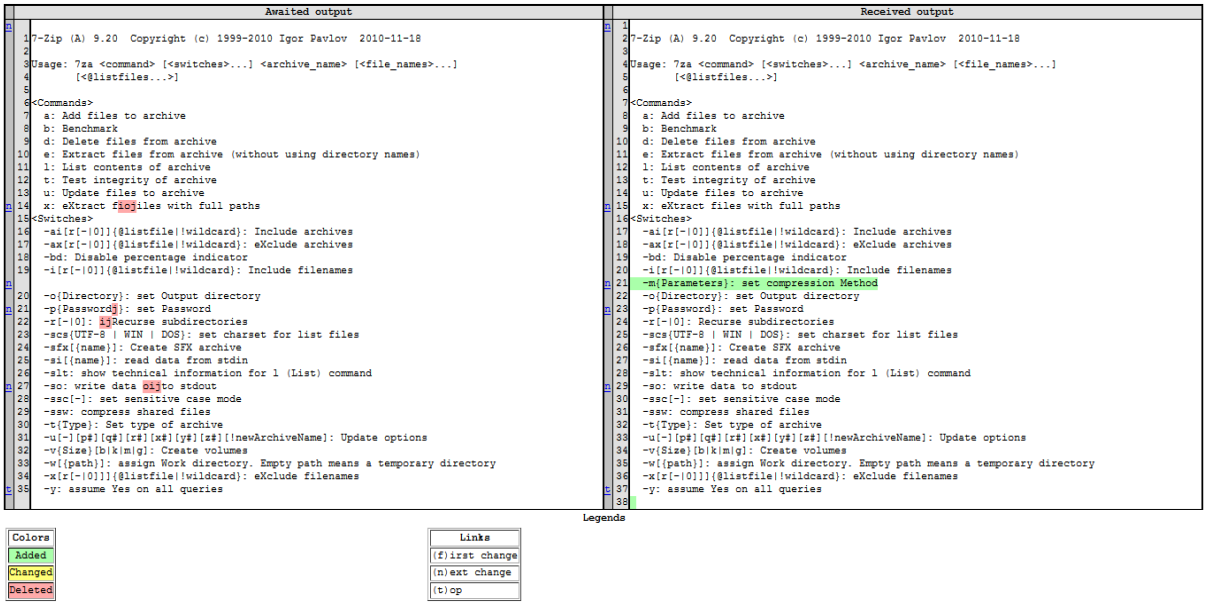

Figure 9. Testing program with the marked errors in it.

5. Life Cycle Ontology on Site

Complex technology that includes a spectrum of technologies, facilities, instruments of planning and reuses specification is realized in ITK of web-site [6]. (http://sestudy.edu-ua.net)

This site is based on standard systems (Eclipse, Protégé, CORBA, MS.Net and others), systems of support of co-operation of the programs, systems and environments between itself VS.Net “Eclipse” Java [6] [11]-[15].

The main menu of web-site has a few sections: TECHNOLOGIES, INTERPRABILITY, INSTRUMENTS, TEACHING and others. Realization of specified operations from class of operations of components development, assembling, change and their configuring is led in the “Technology” section. On this section it is given such position:

–Generation of DSL description to the LC domain;

–Ontology of presentation of the standard LC domain and domain of calculable geometry;

–Beb-services for interconnection different components in environment MS.Net, IBM, Eclipse;

–Transformation of general types of GDT data to fundamental FDT and others.

Web-site is oriented on realization by LC ontological facilities with the use of the Protege system. After its help by the student T Litho of departments the “Informative systems” KNU are developed ontology of calculable geometry, which behaves toward the normative course. Web-site is developed by three languages (Ukrainian, Russian, and Eng.). As Google statistics show, to web-site apply from the different countries (more 35,000 users–teachers and students). This site contains a textbook the “Software Engineering” and is used by author for the E-teaching to all aspects of this discipline. By me lecture at the ICTERI-2012 conference were done, in which mapping new approaches to teaching students of SE.

6. Conclusions

The essence of this work focuses on the automation LC by ontological description conceptual model. It is new approaches to the description of domain SE Standard ISO/IEC 12207-2007. Perform three basic tasks: to develop a conceptual model LC and describe this model in terms of language (DSL, OWL); to generate variants LC for development different systems; to consider the training of students scheme using LC by described ontology.

The formal terms for describing the conceptual model LC of the domain ontology are given. The table description of general, organizational processes and support processes of the standard LC is used for presentation processes in the language of DSL. The characteristic domain model and process model LC in DSL is done. A scheme describing LC in DSL is transformed to a lower level XML for processing systems in the environment (IBM, VS.Net, JAVA and so on).

Submission LC processes graphically DSL Tool VS.Net, and in the language XML is described. Conception of automation of the LC and realization testing process is discussed. As a practical implementation process is selected by process testing. A formal description of the conceptual model testing in terms of Protégé systems and algorithm testing in the language Ruby is realized.

It is noted that the ontological model of LC and computational geometry is implemented by the MIPT, KNU students. Technology of work with that ontology is presented on the website, which gives an access to realize ontology. (http://sestudy.edu-ua.net)

References

[1] Gomes-Perez, A., Fernandez-Lopez, M. and Corcho, O. (2004) Ontological Engineering. Springer-Verlag, London, 403 p.

[2] Mernik, M., Heering, J. and Sloane, A.M. (2006) When and How to Develop Domain—Specific Languages. *ACM Computing Surveys*, **37**, 316-344.

[3] Lavrischeva, E.M. (2014) Software Engineering Computer Systems. Paradigms, Technologies, CASE-Tools Programming. Nauk, Dumka, 284 p. (In Russian)

[4] Lavrischeva, E.M. (2013) Ontological Representation for the Total Life Cycle of AC Line Production of Software Product. *Proceedings Conf.TAAPSD*'2012, *Theoretical and Applied Aspects of Building Software Systems*, Yalta, 25 May-2 June 2013, 81-90.

[5] Lavrischeva, E.M. (2013) The Approach to the Formal Submission of Ontology Life Cycle of Software Systems, Vesnik KNU, a Series of Physical and Math. *Sciences*, **4**, 140-149.

[6] Lavrischeva, E.M., Zinkovich, V., Kolesnik, A., *et al*. (2012) Instrumental and Technological Complex for Development and Learning Design Patterns of Software Systems. State Intellectual Property Service of Ukraine, Copyright Registration Certificate No. 45292, 103 p. (In Ukrainian)

[7] Korotun, T.M. and Lavrischeva, E.M. (2002) Construction of the Testing Process of Software Systems. *Problems of Programming*, **2**, 272-281. (In Ukrainian)

[8] Korotun, T.M. (2005) Models and Methods Testing Engineering Programs Systems in Resource-Limited Settings. Autoref Dissertation, Kiev, 23 p. (In Ukrainian)

[9] (2005) Walkthrough. Domain–Specific Language (DSL) Tools.

[10] Protégé—Frames User's Guide. http://protege.stanford.edu/doc/index.php/PrF_UG

[11] Mens, C., Van Gorp, P. and Czarnecki, K.A. Taxonomy of Model Transformation. http://drops.dagstuhl.de/2–5/11

[12] Lavrischeva, E.M. (2013) Generative and Composition Programming: Aspects of Developing Software System Families. *Cybernetics and Systems Analysis*, **49**, 110-123.

[13] Lavrischeva, E.M. and Ostrovski, A. (2013) New Theoretical Aspects of Software Engineering for Development Application and E-Learning. *Journal of Software Engineering and Application*, **6**, 34-40. http://www.crirp.org/journal/jsea

[14] Lavrischeva, E.M., Stenyashin, A. and Kolesnyk, A. (2014) Object-Component Development of Application and Systems. Theory and Practice. *Journal of Software Engineering and Applications*, **7**, 14. http://www.scirp.org/journal/jsea

[15] Lavrischeva, E.M. (2013) Conception of Programs Factory for Presentating and E-Learning Disciplines Software Engineering. 10*th International Conference on ICT in Education, Research and Industrial Applications*, Ukraine, 16 June 2013, 15. http://senldogo0039.springer-sbm.com/ocs/

[16] Lavrischeva, E., Ostrovski, A. and Radetskyi, I. (2012) Approach to E-Learning Fundamental Aspects of Software Engineering. 8*th international Conf. ICTERI—ICT in Education, Research and Industrial Applications*, Kherson, 6-10 June 2012. http://ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-p-176-187