Ontological Mapping for Conceptual Models of Software System

Guskov G.U.
Namestnikov A.M.
Ulyanovsk State Technical University
Ulyanovsk, Russia
Email: g.guskov@ulstu.ru
Email: nam@ulstu.ru

***Abstract*—The paper proposes a integration methodology of conceptual models and domain ontology. Also a system based on the use of this technique was considered. The developed system implements the integration of UML-diagrams as conceptual models and ontologies represented on OWL [15]. The results of this system for projects from open source repositories and the ways of further development are presented.**

***Keywords*—ontology, engineering design, conceptual model, UML-diagramm.**

I. Introduction

Description of the application domain [9] in the form of formalized set of documents in OWL format is rare in industrial projects. A formal description of the application domain is time-consuming, both of experts and engineers that have the skills to work with ontologies. Artifacts created during the development of software products are usually described formally. These artifacts include requirements specification, requirements, conceptual models, source code with comments, version control system logs, etc. Transfer the knowledge from these artifacts in an ontology is much easier than from non-formalized texts on relevant topics.[10] [13] This article describes an approach to extracting knowledge from conceptual models and further integration conceptual models with domain ontology.

Generally any manufacture can be automated to some degree. Automation is done using specialized software packages. All stages of working with the products need to design: production, implementation, modernization, marketing, etc. During the creation of software products a qualitative formalization of application domain have a place, because software when is being introduced into production it passes multiple levels of testing. Conceptual models were been chosen as an example of artifacts generated by the development of the software. Conceptual models have a high degree of formalization and semantically close enough to the subject area.

The UML-diagrams are discussed as conceptual models in this article. UML-diagrams are most extended in software development today. By using the UML- daigrams it is possible to describe the system from different view points and all diagrams will be built on the common meta-model.[12]

UML is being developed by a consortium of OMG, while OWL belongs to W3C. Because different consortiums specify UML and OWL technologies it is not surprising that direct way of integrating from the manufacturer does not exist. UML is defined as a comprehensive meta-model in the form of a text description.The users get UML in the form of implementation of the software from one of the vendors, such as Visual Paradigm, StarUML, ArgoUML and others. Each of the vendors interpret UML standard in different ways and stores diagrams in its unique internal format. The way to integrate diagrams from different vendors is XMI format (XML Metadata Exchange). Nevertheless, diagrams from different publishers in the exported XMI format also have unique features.

Designers can use the UML in different ways depending on the purpose of creating diagrams. M. Fowler [8] identifies three different approaches to the use of language: sketches, detailed design and visual programming language.This article presents the results of frequency analysis of UML elements and approaches to using UML.

OWL most relevant language for describing ontologies.To create OWL ontologies software Protégé is used. By using Protégé it is posible to create and edit the ontology and execute queries using different reasoners. Importantly, the Protege allows to export and import OWL ontology in various formats of knowledge representation. Thus OWL format is compatible with most formats of knowledge representation.

The relevance of the integration problems are caused by the accumulation of a large number of artifacts in the projects of large software organizations. It is necessary not only to fix the design decisions in the form of artifacts, but also automate process making changes to the project. Purpose of processing UML-diagrams is to compile a list of conflicts and alternatives. In this area there are several papers devoted similar subject. There are a similar software systems for the transfer of conceptual models to the production knowledge base [1] using its own notation representation of the productive knowledge [2]. Examples of transformation of the same works are presented in [3], [4] [14]. Authors are positioning transformation as a system for the transfer knowledge from the diagrams in the ontology.

The difference of this article from the above is the approach to the problem. Projects in large enterprises have a long history and can be supported and expanded by decades. It is therefore necessary to ensure a permanent connection between conceptual models and the application domain. Such an approach would avoid the most costly mistakes in the development

of projects and ensure their semantic coherence. In order to integrate conceptual diagrams and ontology, it is necessary not only automated conversion tool from UML to OWL, but also a system to work with UML an enhanced knowledge by OWL. This research is an attempt to produce integration the domain ontology in the software development process at the level of conceptual models. Thus,the normative documentation [5] [11] and source code should be linked with the ontology.It will allow to automatically detect conflicts and inconsistencies between projects in the same domain ontology.

II. Formulation of the problem

Modern trends in the development are to increase the flexibility and design of information systems, as well as the need to become more flexible.The flexible design suggests frequent modifications version of the conceptual models rather than building them again each time. Designing from the beginning is extremely expensive. Consideration of single system module separately of the remaining components will cause problems. Among the developers it is widely believed that the program can not be described better than the source code, if it is written qualitatively. But this statement is rather the consequence of a lack of understanding and ambiguity the interpretation of conceptual diagrams. Besides the user of conceptual diagrams can be a non-programmer but the domain expert, the customer (or its qualified representative) or manager.

In our opinion, the design should be more standardized and more closely integrated into the development process. We understand integration of design into the process of an information system development not like automatically generate source code, but the exact structure of the program description and the nuances of its interaction. The technology will be accepted by the community when its use reduces the time required to execute a certain work. In order to increase the design flexibility it is necessary automatically perform the consistency check UML diagrams by reasoner working with ontology in which these diagrams are integrated.

For the integration of domain ontology and design diagrams necessary to determine:

1) Format of the conceptual models description;
2) Format of the domain ontology description;
3) Rules for the conversion of conceptual models to the domain ontology;
4) Architecture of software for integration system;
5) The algorithm of integration system;
6) The results of transformation experiments.

III. Format of the conceptual models description

There are several common languages and notations that allow to describe the conceptual models, such as: IDEF0, IDEF1x, IDEF3, DFD, eEPC, UML etc. Some of these notations have not been widely adopted and were considered obsolete, such as IDEF2. Other notations, such idef0 or dfd are successfully applied.Notations and modeling languages are described in the official documentation. Documentation defines the purpose, the basic elements , communication between them and the semantic interpretation of the diagrams. Often diagrams are built without the help of special software, as an image on paper or image file. Diagrams that have been saved as an images can not be updated in case of changes in the system. The value of such diagrams is determined by the date of its creation. These diagrams could be developed to synchronize the view points of developers on the system at the time of creating diagrams. If the diagram is constructed in the form of image, it is impossible to produce on an automated processing. Therefore it is necessary to focus on diagrams constructed with the help of specialized software.

The Unified Modeling Language (UML) was chosen as the format of conceptual models of the for the following reasons:

- availability of detailed and unambiguous documentation;
- implementation a set of diagrams describing all aspects of the system;
- wide popularity among engineers community and IT engineers in particular;
- standardized diagrams export format - XMI .

Different organizations may have their own custom UML design rules somewhat different from the standard or extend it.It is necessary to support not only the most common diagram elements, but also be able to add new items and delete obsolete.

IV. Format of the domain ontology description

As an ontology description format was chosen OWL.OWL has several modifications. Selection of modification defines the semantic power of language by the set of available syntactic rules.For solving the problems of integration are best suitable OWL Full as the most complete implementation of the OWL. OWL Full allows to describe the same concept as a class and as an object in different situations. But using OWL Full has quite impressive drawback, namely the absence of guarantees of solvability query to the ontologies in finite time. Therefore, in the process of implementation of the system we will try to harness the power of OWL full only where it is needed. In other cases, we should use OWL DL.

V. Rules for the conversion of conceptual models to the domain ontology

The elements of conceptual models should be translated into the concepts of the ontology with regard to their semantic interpretation. Semantics of the whole diagram is being formed from the semantics of diagram elements and the semantics of their interaction. Accordingly, it's important to translate the semantics of the elements of diagrams in view of the global UML meta-model. At an earlier stage of development of the integration system [7], rules for the transfer of some objects the class diagrams have been identified. Consider the rules for conversion of some elements of class diagrams to the concepts of the ontology.

A. Element and Relationship

The Element and Relationship are root concepts and provide the basis for modeling all other concepts in UML. UML-diagram contains Elements. Descendants of Element provide

semantics appropriate to the concept they represent.Each Element can hold other Elements. Elements hierarchy are being represented in the ontology as a hierarchy of classes, in which the certain classes will be presented as nodes of last level.
A Relationship is an Element that specifies some kind of relationship between other Elements. Descendants of Relationship provide semantics appropriate to the concept they represent. By this definition Relationship can be translated to the ontology as a subclass of the class Element. But in terms of the structure of OWL is better to move a hierarchy with root Relationship as ObjectProperty.This problem can be reduced to the problem of dualism of concept and attribute. This problem is enough common, one option of solving is considered in the article [6].

B. Type and DataType

To determine the Elements such as a class with its attributes and operations, it is necessary to correlate the data types of UML and OWL. DataType in UML is a subset of Type. Type is a subclass of Element in turn. A Type specifies a set of allowed values known as the instances of the Type. Basic types of OWL and UML data are taken from XSD that means that the types can be transformed directly without additional logic.The data types in UML and OWL are not completely identical. Unique data types, in turn, are based on the same basic XSD data types, and are specified by the restrictions on the basic data type.

C. Classes

The concept of class exists as in the UML, as well as in OWL.In OWL class is a set of individuals. As a built-in classes are offered the set of all instances of the Thing class and the empty set of instances Nothing class. Custom classes are interconnected by a relationship of ObjectProperty.Custom classes and connected with literal values by using the DataTypeProperty. UML class diagram considers a class as an aggregate internal structure and behavior of objects. Concepts of Class differ semantically, but the translation will be available at the level of the class hierarchy. Internal structure classes from UML-diagrams can be translated in OWL classes using ObjectProperties and DatatypeProperty. Most object-oriented programming languages do not support multiple inheritance because of the ambiguity problems of inheritance of individual members of the class. However, such construction is permissible in the description of domain ontology.

During the the research it was necessary to solve the problem of translating the hierarchy UML class in the hierarchy of OWL classes. Inheritance relationship in UML, and the OWL determined as follows, each instance (an individual) is a subclass of the base class instance.The inheritance relationship in UML, and the OWL defined equally, each instance of a subclass is instance of the base class.

If you need to extract data from a domain uml-diagram to fill the ontology class hierarchy can be translated uml-diagram hierarchy OWL classes without any additional changes. But in the case of integration uml-diagrams with the ontology is necessary to present UML meta model in the form of ontologies, and specify classes as individuals.In this case the generalization from UML-diagram is entered as relation between individuals. Generalization from UML will be translated to OWL as subproperty of Relationship, which in turn is ObjectProperty subproperty.

D. Classes:Attributes

Class attributes can be divided by data types on a attributes of primitive type (xsd schema) and custom attributes. Custom attribute contains a reference to the class object, transfer, etc.The attributes of the primitive data types are translated into the ontology as a DataTypeProperty. Domain of DataTypeProperty equal to the class that owns the attribute and Range equal to the primitive type. Enumeration translated into a user-defined data type DataTypeProperty for which are set predefined values. Class attribute that contains a reference to an object of another class is translated into ObjectProperty. Domain of ObjectProperty equal to the class that owns the attribute and Range equal to the class to which the attribute refers.

E. Links

Relations used in the design of UML class diagrams, usually do not have a direct analog in the ontological representation of the domian. A different approach to the description of the relationship between the elements is explained by the different objectives of creating diagrams and ontologies.

When transferring data from diagrams, it was required to establish a conformity of relationship UML-diagrams and ontologies.Attempts to implement this conformity in previous research were limited use of private cases. Also rarely the result such conversion can be uniquely interpreted in the reverse conversion. In order to implement long-term integration between diagrams and ontology, this approach can not be used.

When using ontology built on the basis of a meta-model of UML relationships all meta-model will be transformed into a hierarchy ObjectProperty. Specific relationships defined on a particular UML-diagram are converted into instance of ObjectProperty for classes translated as individuals.

VI. ARCHITECTURE OF SOFTWARE FOR INTEGRATION SYSTEM

General scheme of the system is shown in Figure 1. The diagram presents the key components of the system, the connection between them, the data format of relations and the role of users are interacting with them. It is assumed that the organization has the following staff: programmer (developer directly), designer (the most experienced programmer) and domain expert. We assume that the domain expert and a specialist in working with ontologies are one and the same person. Although are generally, in practice, it is still two different people. Furthermore, one specialist of ontologies usually aggregates in the ontology knowledge of a large number of domain experts.The same employee may be located in different roles at different times. For example, designer can also be a computer programmer.

The set of UML diagrams usually characterize more than one project. For example : server application, client applications for different platforms, tools, services, etc. can be realized as independent applications, but its UML-daigrams may substantially intersect. Programmers working on different projects,

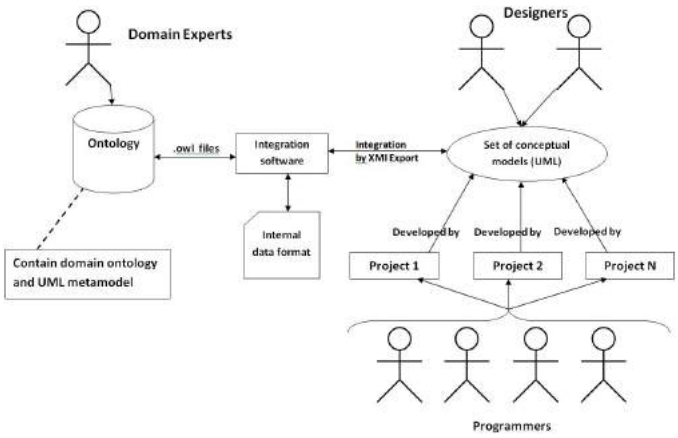

Figure 1. General scheme of the system

can spend less time on the synchronization of applications through the use of common parts of design.

Integration System creates an internal project that includes the data from the conceptual models and data from the domain ontology. UML- diagrams can be loaded into the project in the XMI format. Ontology in OWL format can also be added to the project.

Domain experts could work with ontology directly through the instrument to which they are accustomed. A feature of the ontology used in the scheme is a part responsible for the representation of meta-model UML diagrams.Not all elements of the UML diagrams can be unambiguously represented on the application domain.In this regard, it is proposed to link the elements of domain ontology with elements of the UML meta-model with the help of a special relationship.

Figure 2 shows a sequence diagram representing the workflow of the system integration during designing:

- 1.The designer creates a set of UML-diagrams;
- 2.Domain expert creates ontology.In this case, it does not matter which action will occur first and which is second. Usually the enterprise already has existing set of UML diagrams, because the creation of a UML diagrams set is shown first;
- 3. Domain expert formulate request to integrate ontology and a set of UML-diagrams.This action is performed by the domain expert, as it becomes relevant only after the creation of ontology;
- 3.1 Integration system creates a UML meta-model part in a given ontology;
- 4, 4.1, 4.2 Integration System takes data from domain ontology in OWL format;
- 4.3, 4.4 Integration System takes data from the UML-diagrams in XMI format;
- The integration system combines the data obtained from the UML-diagrams and ontology using rules of integration between 4.4 and 4.5 messages ;

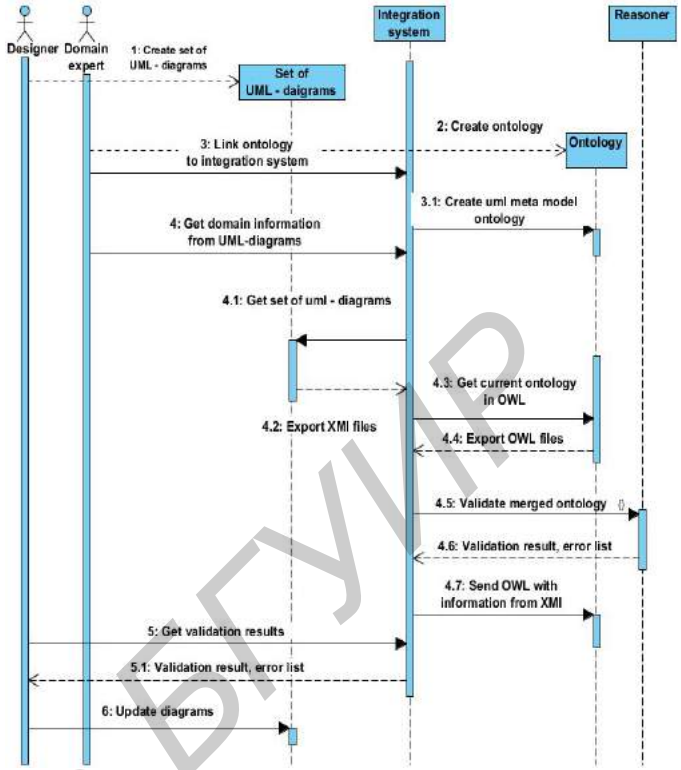

Figure 2. Workflow of Integration system submitted to the sequence diagram

- 4.5, 4.6 integration system checks the consistency of the ontology and uml diagrams for axioms specific to this application domain;
- 4.7 The integration system updates the ontology by the data derived from UML;
- 5, 5.1 The designer receives a diagram of processing results using reasoner from the system integration;
- 6 Designer updates UML-diagrams.

VII. The algorithm of integration system

The algorithm processes the XMI file, which is a specific structure of xml file. This feature allows to build non-trivial queries to the document. Thus, it is possible to not think about the implementation of a query at a low level (indexing, data structures for storage, etc.).

Listing 1. Parsing UML in XMI format

```
1 begin
2     classDiagram = getClassDiagramm(
          XMIroot);
3     if (classDiagramm == null) than
4         exit;
5     UserTypes [] = getTypes(
          classDiagram);
6     Classes [] = getClasses(ClassDiagram
          )
7     foreach(class in Classes[])
8     begin
9         class.Attributes = getAttributes
              (class);
```

```
10          class.Operations = getOperations(
                class);
11          class.Childs = getChilds(class,
                Classes[]);
12          class.Parent = getParent(class,
                Classes[]);
13        end
14      MatchingDataTypes(Classes [],
            UserTypes[]); // Replacing XMI ID
             on the type of names
15      Associations [] = getAssociations(
            root, Classes []);
16      Dependencies [] = getDependencies(
            root, Classes []);
17      //Generation of OWL
18      owl = writeHeader(owl);
19          //Generation ontology header
20      owl = writeDataTypes(owl, UserTypes);
21          //DataTypeProperties
22      owl = writeClasses(owl, Classes[]);
23          //Classes
24      owl = writeLinks(owl, Classes[],
            Associations [],
25          Dependencies []) //ObjectProperties
                ;
26      ErrorList [] = reasoner.Validate(owl)
            ;
27      foreach(error in errorList)
28      begin
29          print(error);
30      end
31  end
```

This algorithm was implemented in the Visual Studio development environment, the language C # in the form of a desktop application. The core modules are implemented as linked libraries.

VIII. THE RESULTS OF TRANSFORMATION EXPERIMENTS

The experiments have been some exceptions, but all diagram elements described in the article, have been successfully translated into ontology. It is worth noting that the developers rarely use a wide range of elements. Using UML in such a way leads to creation of sketches intended for understanding difficult situations in project functioning. The histogram in figure 3 represents the frequency distribution of the elements of the class diagram.

For testing implemented systems were selected for the class diagram of various projects of the Open Source GitHub. We were searched diagrams on the basis of the file expansion names. In this regard, we chosen development tools uml-diagrams supporting XMI export.

Table 1 contains a summary of the editors of uml-diagrams. Many editors do not implement export XMI at all, some supports outdated version of the format. Some editors allocates export to XML as a paid functionality. As a result, test a few editors were selected: Visual Paradigm, Enterprise Architech and Altova UModel.

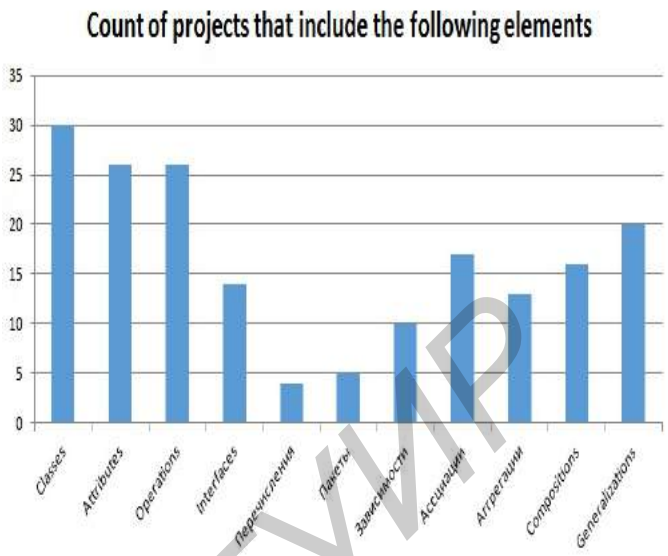

Figure 3. Histogram of the frequency of use of the elements

Table I. RESULTS OF TESTING BY EXPORT UML DIAGRAM TO XMI FORMAT

Editor	Version XMI	Export	Translation	Comment
ArgoUML	1.2	Success	Impossible	Outdated version XMI
Astah	1.1	Success	Impossible	Outdated version XMI
Altova UModel	2.1, 2.4.1	Success	Success	
Enterprise Architect	1.1, 2.1	Success	Success	
MagicDraw	2.1	Success	Impossible	Only in paid version
Innovator Enterprise	2.1	Success	Impossible	Only in paid version
modelio	1.1	Success	Impossible	
StarUML	1.1	Success	Impossible	
Visual Paradigm	2.1	Success	Success	

Figure 4 shows a histogram of the frequency distribution by use UML in projects. Classification of approaches to the use of UML borrowed from M. Fowler [8]. M.Fowler offers to share the use of UML on the way:

1) Sketches are local description of complex parts of the system. Sketches do not describe the global behavior of the system, the interaction of components at current moment;
2) Detailed description of the project represents the non-trivial part of the project taking into account all aspects, given the interaction of the system modules. Detailed description do not contain obvious part of the project,for example, the properties or fields are needed to implement the encapsulated class capabilities;
3) A visual programming language provides code generation designed by class diagram. Visual programming language suggests mandatory accounting of all aspects of the system, even the most obvious.

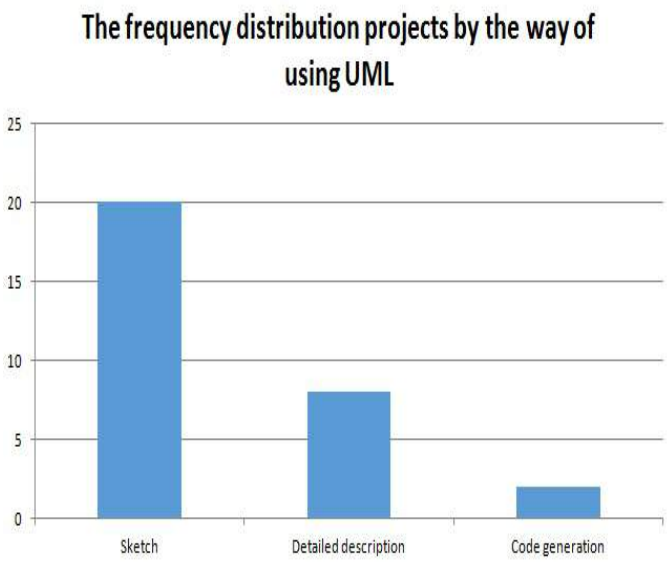

Figure 4. Histogram of the frequency distribution projects by the way of using UML

IX. Conclusion

This research modified and greatly expanded the concept of the system integration of conceptual models and ontologies. A new ontology structure consisting of two parts was developed.These parts are a presentation of a UML meta-model and domain ontology. Result of research reformulated conversion rules of uml class diagram elements and a special case of the conversion elements of the meta-model. We collected data allow to estimate way to use uml-diagrams.

Acknowledgment

This research are supported by Russian Foundation for Basic Research, project numbers 16-47-732120 and 16-47-732033.

References

[1] N.Dorodnich i A. Urin, *Ispolzovanie diagram klassov UML dlya formirovaniya produccionnih baz znanii*, Programnaya ingeneriya N4, 2015.

[2] M.Grishenko, A. Urin, A. Pavlov,*Razrabotka ekspertnih sistem na osnove transformacii informacionnih modelei predmetnoi oblasti*, Programnie producti i sistemi N3, 2013.

[3] D. Gašević and D. Djurić, V. Devedžić, V Damjanović*Converting UML to OWL Ontologies*,In Proceedings of the 13 th International World Wide Web Conference , NY, USA, 2004, pp. 488-489 (pdf).

[4] J. Zedlitz, J. Jorke,N. Luttenberger,*From UML to OWL 2*, In: Proceedings of Knowledge Technology Week 2011. Springer (2012).

[5] A. Filippov, *Formirovanie navigacionnoi structuri electronnogo arhiva tehnicheskih documentov na osnove ontoogicheskogo predstavleniya*, Avtomatizaciya processov upravleniya. 2013. N 3 (33). p. 61-68.

[6] U.Zagorulko, I Ahmadeeva, A. Serii, V Shestakov, *Postroenie tematicheskih intellektualnih nauchnih internet-resursov sedstvami semantic web*, Trudi 15 nacionalnoi konferencii po iscusstvennomu intelektu KII-2016, Smolensk, T2, 2016, p. 47-55.

[7] A.Namestnikov, G.Guskov, *Programnaya sistema preobrazovaniya UML-diagram v ontologii na yazike OWL* , Trudi 15 nacionalnoi konferencii po iscusstvennomu intelektu KII-2016, Smolensk, T3, 2016, p. 270-278.universe

[8] Fowler M. *UML. Osnovi*, 3e izdanie. - Per. c angl. - Spb: SimvolPlus, 2011.

[9] Dobrov V.B., Lukashevich N. V., *Lingvisticheskaya ontologiya po estestvennim naukam i tehnologiyam: osnovnie principi razrabotki i tekuwee sostoyanie*, // Desyataya nacionalnaya konferenciya po iscusstvennomu intellektu s mejdunarodnim uchastiem (Obninsk, 25-28 sentyabrya 2006 g.) – M.: Fizmalit, 2006.

[10] Gavrilova T.A., Horoshevskii V.F., *Bazi znanii intellektualnih sistem.* – Spb. : Piter, 2000. – 384 c.

[11] Namestnikov A.M., Subhangulov R.A. *Formirovanie informacionih zaprosov k elektronnomu arhivu konceptualnogo indeksa* // Radiotehnika N7 - 2014 p. 126-129.

[12] Namestnikov A.M. *Metauroven informacionnogo obespecheniya SAPR : ot teorii k praktike/* A.M. Namestnikov. - Ulyanovsk : UlGTU, 2015.

[13] Filippov A.A., Moshkin V.S., Shalaev D. O., Yarushkina N.G. *Edinaya ontologicheskaya platforma intellectualnogo analiza dannih*// Materiali VI mejgunarodnoi nauchno-tehnicheskoikonferencii OSTIS-2016, Minsk, Respublica Belarus, 2016.

[14] Almeida Ferreira D., Silva A., *UML to OWL Mapping Overview An analysis of the translation process and supporting tools.* Conference: 7th Conference of Portuguese Association of Information Systems.

[15] OWL 2 *Web Ontology Language Document Overview* – https://www.w3.org/TR/owl2-overview/

ИНТЕГРАЦИЯ ОНТОЛОГИЙ И КОНЦЕПТУАЛЬНЫХ МОДЕЛЕЙ ИС

Гуськов Г.Ю., Наместников А.М.

В статье предлагается методика интеграции концептуальных моделей и онтологии предметной области. Так же в статье рассмотрена система основанная на данной методике. Кроме того в статье приведены основной алгоритм системы интеграции и её поведение на диаграмме последовательностей. Проведён анализ подходов к использованию языка UML и частотности использования его элементов. Разработанная система позволяет интегрировать концептуальные модели в виде UML-диаграмм с онтологиями в формате OWL[15]. В завершении статьи представлены результаты работы системы над проектами из открытых репозиториев исходного кода и пути будущего развития системы.