BUSINESS PROCESSES MODELLING WITH DSM PLATFORM AT INTEGRATED SYSTEMS DEVELOPMENT

Anna Lubyagina, Lyudmila Lyadova, Alexander Sukhov

Abstract*: The first and most labor-consuming stage of information systems development is an analysis stage. At this stage the set of different formal models describing systems domains, different aspects of the created system functioning is created. The model-centric approach focuses attention on the models at the each stage of the development process. Modelling tools are used by developers and experts in specific domains. These tools must be affordable for different specialists. Created models must be clarity and understandable for all developers and domain experts. This feature is supported with domain-specific modelling tools. Language workbenches include means for domain-specific languages creation. Modelling with DSM platform is more suitable for the domain experts, system and business analysts. Different tasks of the system design and development request specific formalizing means needed for modeling. Different modelling languages and tools are used by analysts. They need to support continuity of development stages, reusability of created models for the solution of different tasks. DSM platforms provide model transformations, translation of models from one modelling language to another. These tasks are important for large projects of complex information systems development. The basis of language toolkits creation is metamodeling. The tools of MetaLanguage DSM platform, allowing designing domain specific languages and models for different domains, to transform models at systems integration, are described. Examples of metamodels and models, transformation rules are presented.*

Keywords*: modeling, domain-specific languages, language workbench, metamodeling, model transformations.*

ACM Classification Keywords*: D.2 Software Engineering: D.2.2 Design Tools and Techniques – Computer–aided software engineering (CASE); D.2.6 Programming Environments – Domain-specific architectures; Languages (e.g., description, interconnection, definition); D.2.13 Reusable Software – Domain engineering; Reuse models.*

Introduction

Currently, the problem of expediting the development of new informational and analytical systems and configuration of existing systems to meet new challenges, adapt to changing operating conditions and user needs, becomes more urgent. These tasks can only be resolved through the use of models that represent the formal descriptions of domains and requirements to information systems. The use of models, not only during domain analysis and formalization of requirements for information system, but throughout their entire life cycle let to create flexible systems with the maximum capacity of their adaptation. In this regard, tools that are based on the use of model-based approach to create an informational system become more popular. These tools allow

reducing the complexity of creating and maintaining systems at the expense of «reusability» of the previously described models to generate and configure the software of information systems.

Model-driven approach to software development involves the use of modeling languages, by which the models are built. The *visual languages* are more commonly used as graphical models have greater clarity and understandable not only for programmers, but also for experts in the relevant domains and end-users.

At designing informational and analytical systems the selected CASE-tools and BI-platforms necessitate the use of appropriate modeling languages. When integrating systems, designed by different developers, it is necessary to revise the models, described with using different languages, execute their *transformation* to allow their integration when creating models of the integrated system.

Another task of the system development efficiency increase is to ensure participation of experts in various subject areas in the process of system development. This requires the creation of appropriate conditions: developed models must be clear for not only to specialists in the field of information technologies, system analysts and programmers, but also for domain experts, who need to work with descriptions, in which the common for these concepts, terminology relevant subject area are used. This is possible if the models are created in terms of domain-specific languages (DSL).

For the development of domain-specific languages and model transformations, special software (language tool-kit, language workbenches or DSM platforms) are used.

Modelling Languages and DSM platforms

During the development of complex systems a lot of models are created. These models are used for solving various tasks, for describing the target system from different points of view, at different levels of abstraction. Thus, a hierarchy of models is created [Lyadova, 2008], in which the models are described by the means of modeling languages (modelling language is used as metamodel), and modeling languages are described by the means of meta-language.

There are many model-based approach implementations that use general-purpose modeling languages for model creation. For example, a general-purpose modeling language UML (Unified Modeling Language), together with the standard MOF (Meta-Object Facility) forms the basis for the concept of MDA (Model-Driven Architecture). But general-purpose modeling languages are often difficult to understand, not only for experts in a particular domain, which are involved in the system creation, but also, in some cases, even for professional developers. In addition, it is sometimes difficult to adequately express the domain concepts, operated by informational system users, using a general-purpose language. That is why the model-oriented software development is increasingly being applied visual domain-specific modeling languages (Domain-Specific Modeling Languages – DSML, Domain-specific Languages – DSL), designed to solve a specific class of problems in a particular domain. DSL are easy to use and understandable to the various categories of professionals, as they operate with usual terms for their subject area. Using a DSL, even users are able to cope with models modification. They can be

active participants of the models creation and modification. Users become actors in the informational system development and maintenance, system customizing.

To support the creation of DSLs, a special type of software, called language toolkits, language workbenches or DSM platform (DSM – Domain Specific Modeling) is used. There are various means to create visual DSLs with the ability to define graphical notations: MetaEdit+ ([Kelly, 2013; MetaCase, 2014]) Microsoft DSL Tools ([Cook, 2007; Overview, 2014]), Eclipse GMF ([Sorokin, 2010; Gronback, 2009]), QReal ([Terekhov, 2013; QReal, 2014]), and others. DSM-platforms allow to develop languages and models, as well as make their transformations associated with the need to transition models from one to the other notations, – *model transformation* (translation models, described using one language, into another language).

At the modelling vertical and horizontal transformations are realized. *Vertical transformations* control the process of model transformation during the transition from one level of the hierarchy to the other, such as mapping the metamodel elements to model elements. *Horizontal transformations* are transformations in which the source and target model belong to the same level of the hierarchy. An example of a horizontal transformation is the transformation of a model created by using a one modeling language, to the model described in another language. Means of model transformations can reduce the complexity of informational systems development, providing, on the one hand, the ability to automatically generate data structures and program code, but on the other hand – the ability to export and import models.

There are different approaches to the transformation of models: for example, the system AGG, GReAT, VIATRA uses rewriting graph rules (graph rewriting) to perform the transformations; MTBE approach is based on the programming method for sample [Suhov, 2012]. All currently implemented approaches have various restrictions, complicating their use to describe the transformation (in particular, the need for high skill users). Thus, one of the main problems to be solved when creating language toolkits is the task of developing the means of transformation, removing these restrictions. The implementation of such tools greatly enhances the use of DSM platforms, making them more accessible for different categories of professionals in the development of systems for different purposes and as a basis for the integration of information and analytical systems [Zamyatina, 2013].

Modelling with MetaLanguage DSM platform

System MetaLanguage is a DSM platform, *language workbenches* designed to create a visual domain-specific languages and models with their use, as well as performing the transformations of the created models.

The process of modeling in MetaLanguage includes *several steps* [Suhov, 2013]:

- *Domain-specific language development* (metamodel design) on the basis of metalanguage by using a graphical model editor;
- *Model development* with using the designed language and model validation.

At the *metamodel creation* it primarily needs to determine the basic design of a new language: metamodel entities, the relationship between them, the restrictions on the entities and relationships. The metamodel developer gets a scalable, dynamically configurable visual modelling language as a result of a metamodel

constructing. At the process of design of the first level DSL in MetaLanguage the basic constructions of system metalanguage are used: entity, relationship, constraint. In this case, metamodels developed in MetaLanguage can be used as a metalanguage for creating new languages. The process of modelling designing can be iterated. Thus, the hierarchy of models is built.

Using a developed DSL the system analysts can *create models* that contain objects describing the specific entities of the domain and the relationships between them. Developing the model, modelers need to check whether it satisfies the restrictions that have been imposed on model entities and relations. Validation of the created model performs.

When changes are made in the metamodel (in the description of the DSL), the system automatically makes the necessary changes to the model, created by using this metamodel (domain-specific language).

The component of transformation is used for automating model transformations in MetaLanguage.

Vertical transformation is a transformation of the model described in one level of the hierarchy to the model in a different level. Model transformation of the low-level model to the higher-level model of designed hierarchy in MetaLanguage corresponds to the operation of model creating. Reverse transformation allows the interpretation of the higher-level model, determination of the types of model's elements, the implementation of various operations of this model. Algorithms for performing the corresponding operations can automatically maintain consistency of models at different levels.

In order to perform the *horizontal model transformation*, it is needed to set the appropriate transformation rules, which are described using the corresponding metamodels. Rules define the correspondence between the constructions of languages (between the elements of metamodels). The transformation description must be carried out with the use of visual languages understood for the different categories of professionals. In MetaLanguage there is the ability to set horizontal transformations of the «model-to-model» type and «model-to-text» type.

In MetaLanguage system model transformation is based on an algebraic approach with a single ejection based on graph grammars. This approach is suggested in the implementations of a number of modifications ([Lyadova, 2012], [Suhov, 2012]). The transformation component provides the possibility to describe the transformation of element attributes of the metamodel, a multi-level description of metamodels of the left and right side of the production rule. It allows describing the transformation rules on the one hierarchical level, and their application – on the other.

Designing Complex System: Tasks and Approaches to Integration of Heterogeneous Informational and Analytical Subsystems

Now much attention is paid for methods and means of information systems integration, allowing combining a number of different subsystems, to expand their functionality, to improve the efficiency of their use.

The concept «integration» is the key word in the development of complex informational and analytical systems. Integration is needed to solve many problems at the complex systems development. The problem of informational and analytical systems integration is investigated in some directions:

- Data integration is defined as joint of data flows, actual and historical data, received from heterogeneous sources;
- Business processes integration is considered as assembly of common processes of different enterprises at their collaboration;
- Integration of inherited systems means integration of different heterogeneous analytical and informational systems as subsystems at the development of complex system.

In this paper focus is on the functional integration of inherited systems in complex integrated informational and analytical system. For example, national banks need software, which automates and integrates several monitoring operations, implemented by separate subsystems, to meet the challenges of monitoring of non-financial enterprises and their activities. Now much of this monitoring work is not automated. Thus, there is a need to establish a unified organizational and technical system of creating and maintaining summary information about the objects of monitoring obtained from distributed, separate sources. A comprehensive system should consolidate existing subsystems that will solve the problem of interaction of subsystems, combining their functionality [Lubyagina, 2014].

A promising approach to the functional integration of systems is the model-driven design (MDD, Model-Driven Design) and engineering (MDE, Model-Driven Engineering). The systematic use of models throughout the life cycle of informational system is suggested: the definition of data structures, code generation for the user interface and main functionality are model-based. The process of creating software is represented as a chain of transformations of the original model into the ready-to-use information system. In addition, different specialists, domain experts and developers require presentation of models in different languages; they describe models using different graphical notations, the different types of charts.

At the heart of MDE is the concept of DSM (Domain Specific Modeling). At developing the model of comprehensive informational and analytical system based on the existing subsystems the problem of integration of subsystems is solved as integration of models developed for different domains with using different visual languages.

To reduce the labor intensity it is suggested to use the DSM platform that allows reusing designed models.

Approach approbation is applied in the design of complex automation system for monitoring activity of non-financial enterprises with integrating some existing informational and analytical systems.

Design and Transformation of Models in the Integrated Informational and Analytical System with Use of MetaLanguage DSM platform

The basis for the development of domain-specific modeling language for monitoring service activity is a general-purpose (universal) language for modeling in the study domain. This language, on the one hand, must have

sufficient expressive power to describe the various activities, automated legacy systems, as well as for modeling complex integrated system. On the other hand, this language must provide the ability to create on its basis new specific languages for modeling various subsystems that take into account their specificities.

The project objectives are:

- To design domain metamodel on the basis of information on the enterprises activities under monitoring (general-purpose modelling language – the basic level of the system modelling);
- To create languages for modelling enterprises in various industries on the basis of the designed general-purpose language metamodel;
- To develop models representing specific business processes of enterprises in the studied domain with use domain specific modelling languages;
- To define transformation rules and execute transformations of developed models to design models of complex system integrating different models described on DSLs.

Figure 1 shows a simplified general-purpose language metamodel created with using MetaLanguage system.

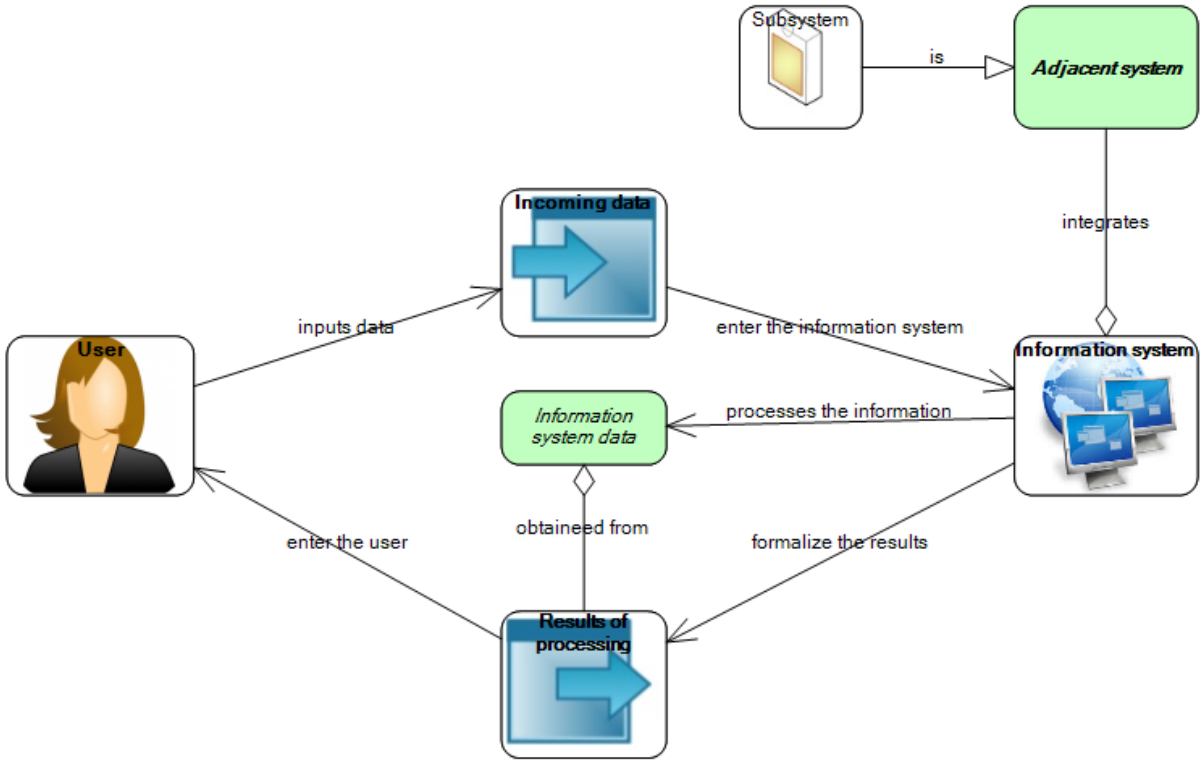


Figure 1. Simplified general-purpose language metamodel built with using the MetaLanguage system

Created general-purpose DSL has the following advantages: a sufficient expressive power, ensuring adequate representation of business processes in the study area; metamodel for ease of understanding, the high degree of its transparency; focus on the tasks to be solved in the development of a comprehensive monitoring system that allows users to easily learn the language and work with models using familiar engineering terminology.

In implementing the metamodel language were created following entities (see Fig. 1):

- «*Subsystem*» – this entity represents subsystems that are the parts of an integrated information system;
- «*Adjacent system*» – this entity represents any subsystem of the integrated system;
- «*Information system*» – this entity represents the complex (integrating) system;
- «*User*» – this entity represents all user roles for users that have access to work with an integrated system;
- «*Incoming data*» – this entity represents input data, incoming documents of the complex system;
- «*Information system data*» – this entity represents interim data at various stages of processing;
- «*Results of processing*» – this entity represents output data, reports in the integrated system.

Entity «*Subsystem*» has a relationship of *inheritance* type with «*Adjacent system*» («*is*»). Entities «*Adjacent system*» and «*Information system*» are connected by relationship of *aggregation* type («*integrates*»). The remaining relationships of entities are type of *association*.

A model of application execution (bid processing) by members of the monitoring group is designed on the base of general-purpose modeling language. Figure 2 shows a simplified version of this model.

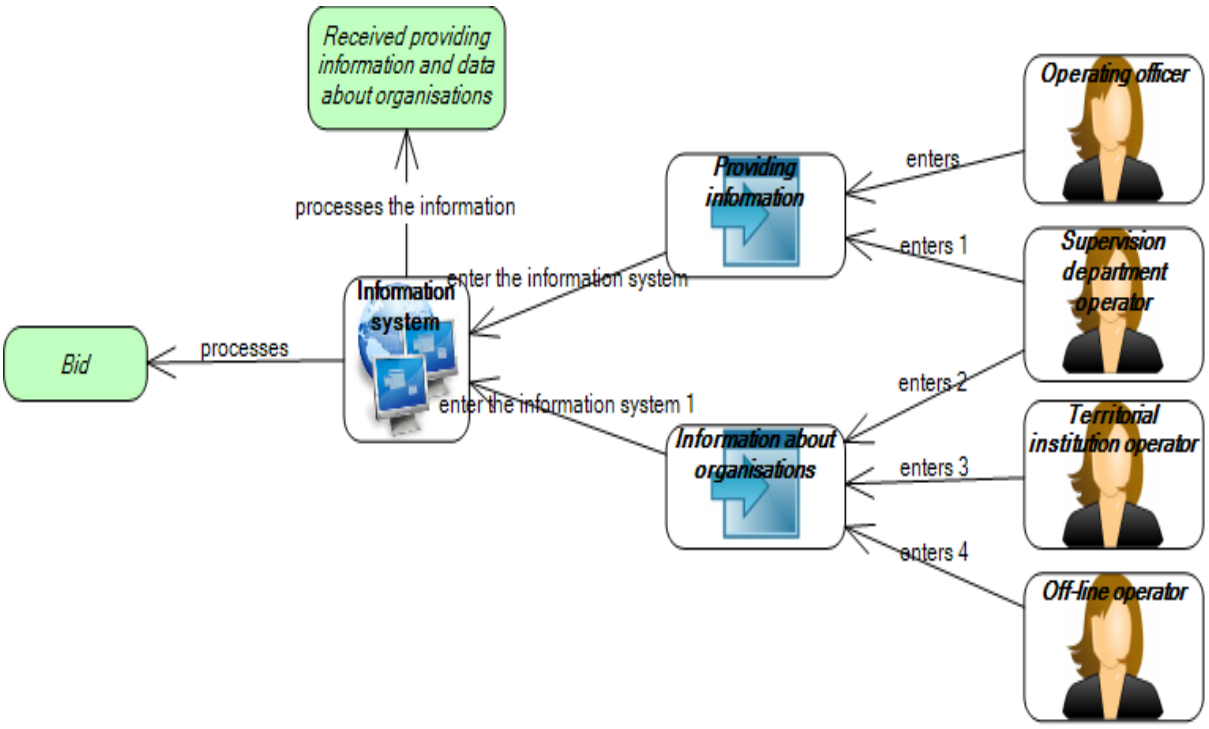


Figure 2. «Bid processing» model, built on the basis of the general-purpose metamodel

The model constructed with using the general-purpose language cannot represent all the operations of the application processing of monitoring system in detail. In addition, the constructed model does not reflect the partition of an integrated system for specific subsystems; functionality of each subsystem and the extent of their

use in the treatment are impossible to distinguish. Thus, the constructed model has a number of shortcomings: lack of visibility and specific details for end users, the unsufficient reflection of the specific functionality of each subsystem.

Modelers need to establish specific languages for specific tasks of various subsystems for specific categories of users in order to overcome these shortcomings.

The general-purpose language (metamodel is shown above) can be used as a metalanguage for designing the second level DSLs. These languages allow describing more clearly the business processes of systems that automate the different tasks of monitoring. New object-oriented language is created in a specific syntax notation of the general-purpose modeling language (see Figure 3).

More detailed models are built on the basis of the new constructed object-oriented metamodel. An example of the simplified model of application processing by members of the monitoring group is shown below (see Figure 4).

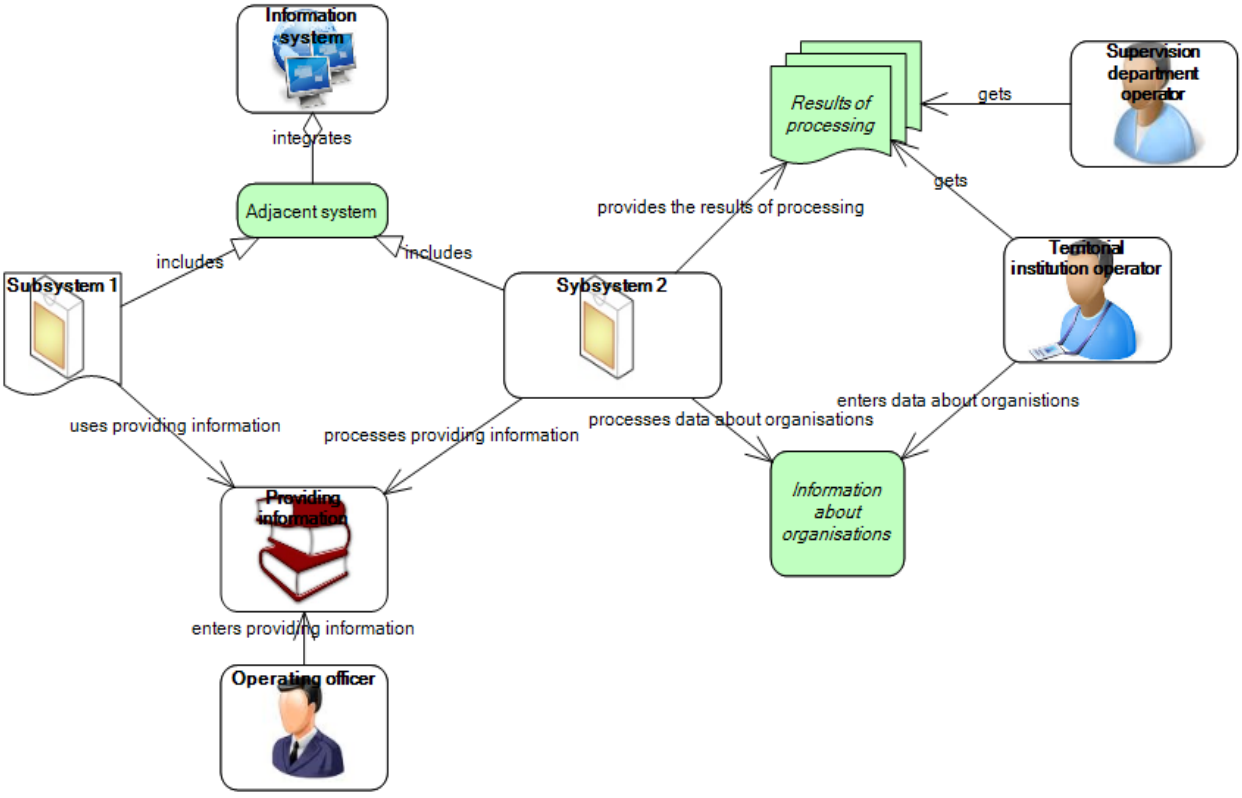


Figure 3. A simplified metamodel, built on the DSL of the first level

This model (Figure 4) is developed with using specific object-oriented language of the second level. It has more detail, reflects the specific end-user experience and functionality of the subsystem, and allows working in terms familiar for users of subsystem.

Modeling languages to describe the other subsystems can also be created on the basis of a general-purpose modeling language.

These examples demonstrate the possibility of the suggested approach to ensure the creation of models that can be set to the desired level of detail and reflect the specificity of the subsystems and users. Modelling languages and models for the two pilot subsystems of integrated informational and analytical system are developed at the research project execution.

However, to meet the challenges of the project, there was a need to translate the constructed models into other languages to transfer the results of the modelling to developers of the complex informational system. In particular, developers need to build ER-diagrams, Use-Case diagrams, etc. Automated translation of the constructed models from designed domain specific notations into standard notations needed for developers were executed with the transformation tools of MetaLanguage.

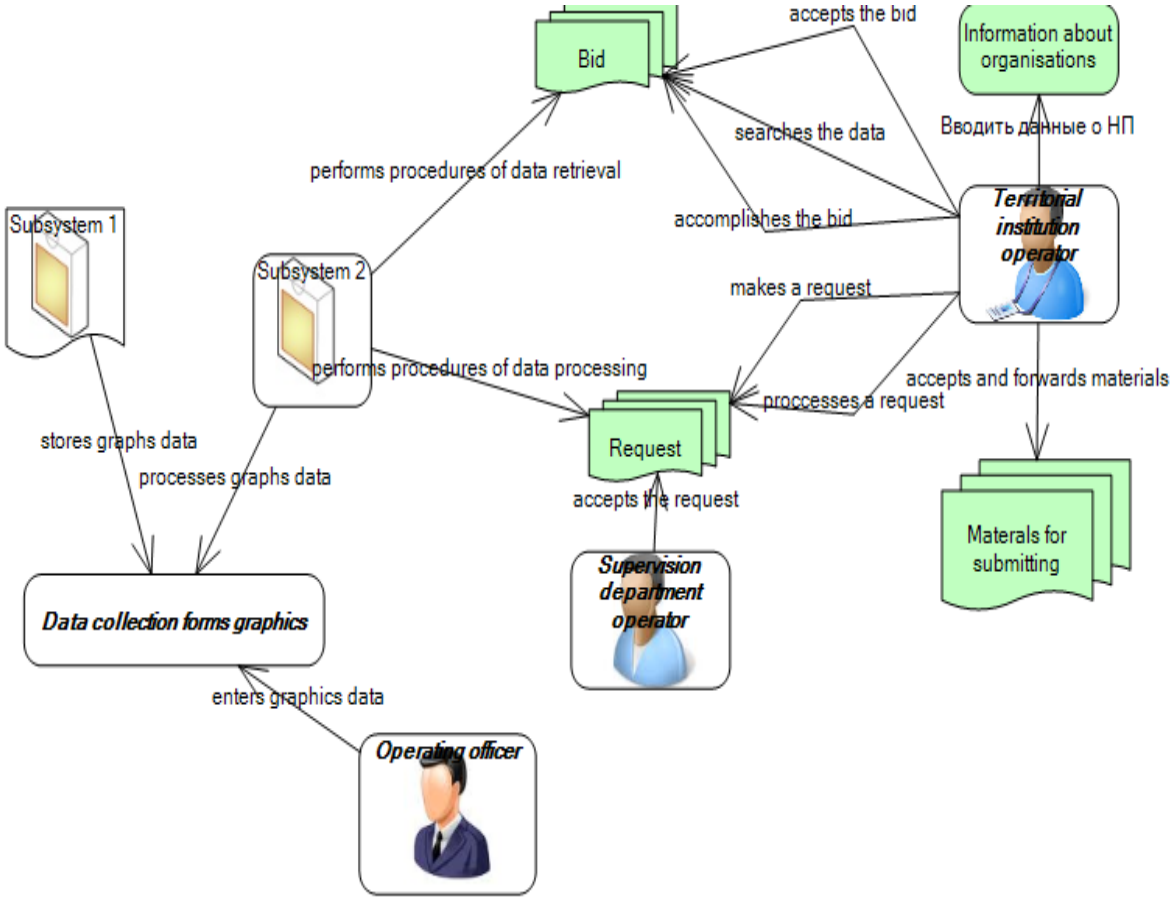


Figure 4. A simplified «Application processing» model, built on the DSL of the second level

To translate models by using the transformation component of MetaLanguage, developers have to execute two steps:

– To describe transformation rules using metamodels of the source and target models (languages);
– To make a model transformation (to translate the created models into the target language in accordance with the described rules).

Transformation rules are described by using visual constructor: developer has to define the left and right parts of the rule. Constructions of the source and target languages are used in the rule definition. If the transformation rules are given, then the second step is performed automatically in accordance with these rules. Moreover, the described rules can be applied to the transformation of models that can be described by using the same languages later, significantly reducing complexity of the translation. A set of rules can be expanded later. In the description of the rules, as well as descriptions of modelling languages, it is possible to make changes.

To translate the models described on the created DSL into the notation of UML (activity diagrams), metamodel shown in Figure 5 was used.

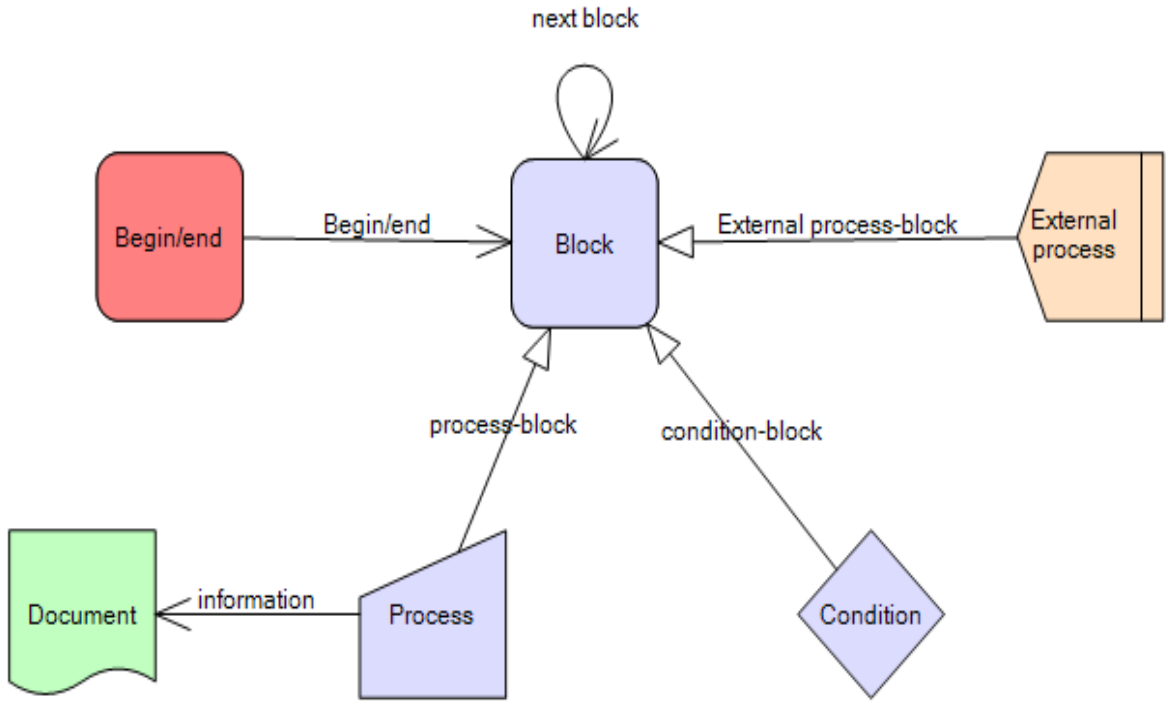
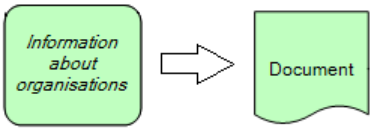
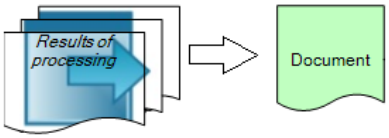
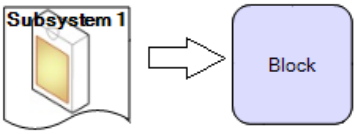
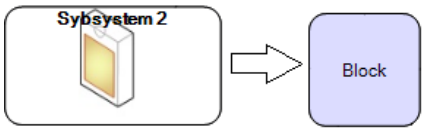
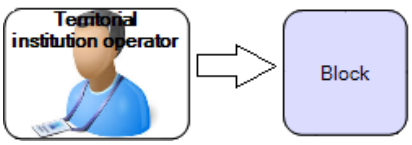
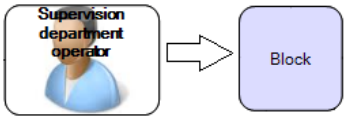
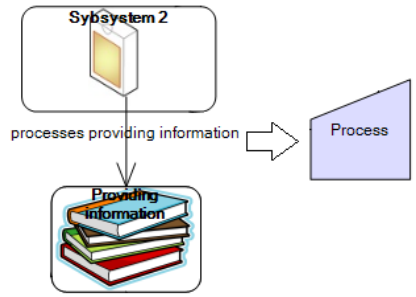
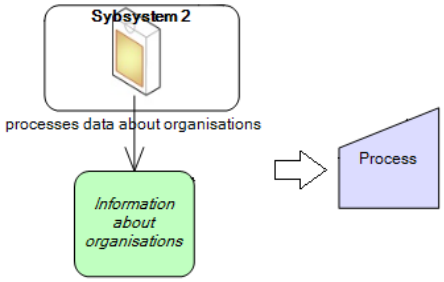
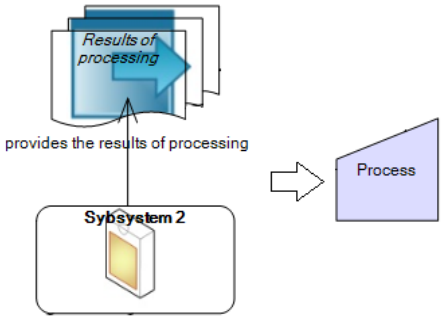
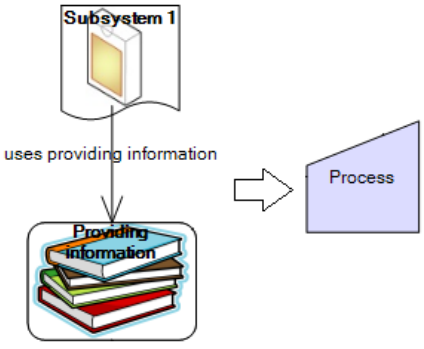


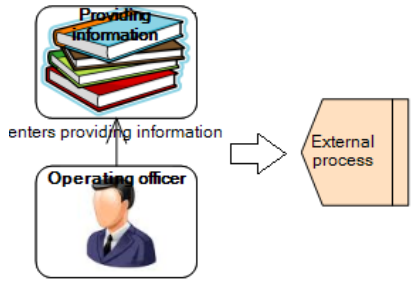
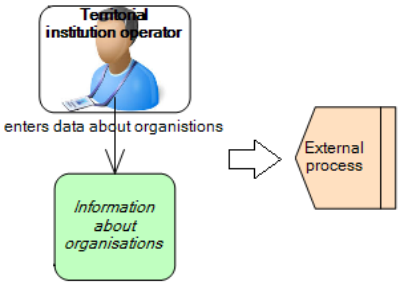
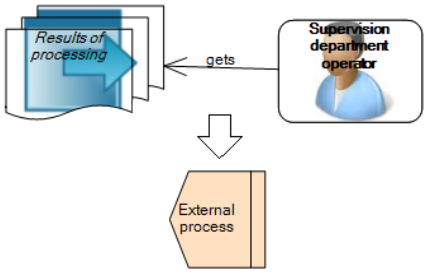
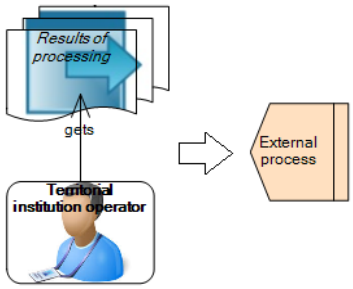
Figure 5. Metamodel of activity diagram in MetaLanguage system

Transformation rules developed in the MetaLanguage system for translation of the constructed models for language UML (for building activity diagrams) are shown in Table 1. Figure 6 shows the result of the transformation of the «*model-to-model*» for the simplified model «*Application processing*».

Table 1. Transformation rules of «application process» model in the UML notation (activity diagram)

Transformation rule (left and right parts)	Graphical mapping
Information about organizations – Document	
Providing information – Document	
Result of processing – Document	
Subsystem 1 – Block	
Subsystem 2 – Block	
Operating officer – Block	
Territorial institution operator – Block	
Supervision department – Block	

Transformation rule (left and right parts)	Graphical mapping
Processes providing information – Process	
Processes data about organizations – Process	
Provides the results of processing – Process	
Uses providing information – Process	

Transformation rule (left and right parts)	Graphical mapping
Enters providing information – External process	
Territorial institution operator – External process	
Gets – External process	
Gets – External process	

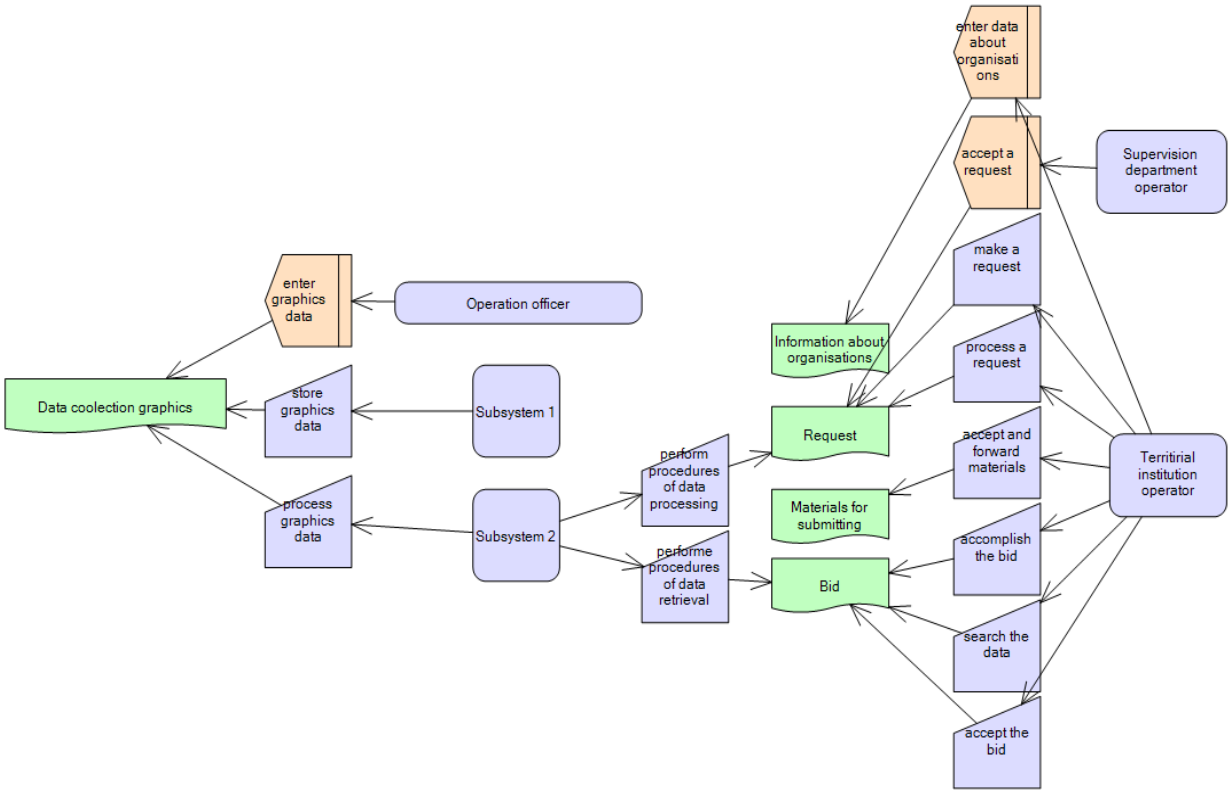
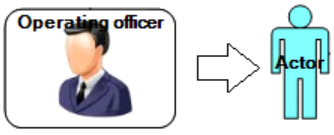
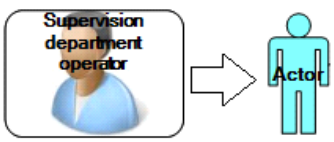
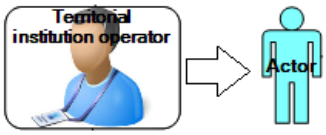
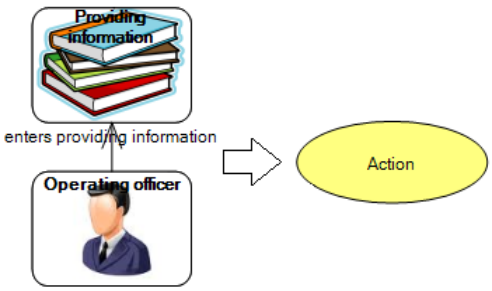
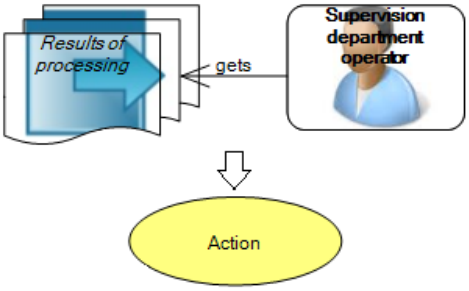
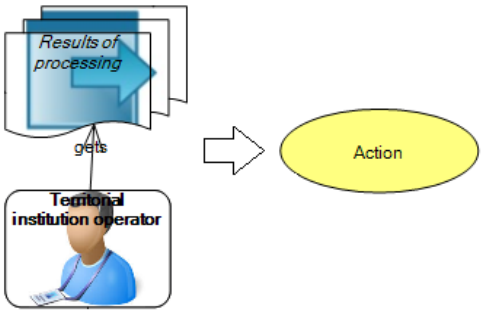
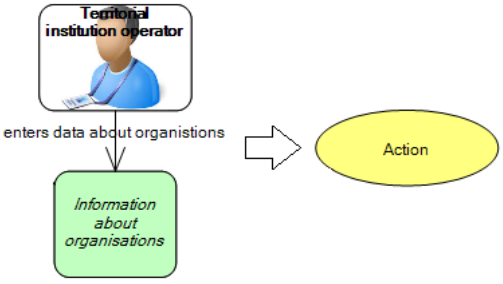


Figure 6. The result of the model «Application Processing» transformation – Activity diagram

In addition, the model of the application processing has been transformed into Use-Case diagram notation. Figure 7 presents a metamodel of Use-Case diagram, and Figure 8 shows the result of transformation of the model formed in accordance with the rules shown in Table 2.

Table 2. Transformation rules of «Application processing» model into the UML notation (Use Cases Diagram)

Transformation rule (left and right parts)	Graphical mapping
Operating officer – Actor	
Supervision department operator – Actor	

Transformation rule (left and right parts)	Graphical mapping
Territorial institution operator – Actor	
Enters providing information – Action	
Gets – Action	
Gets – Action	
Enters data about organizations – Action	

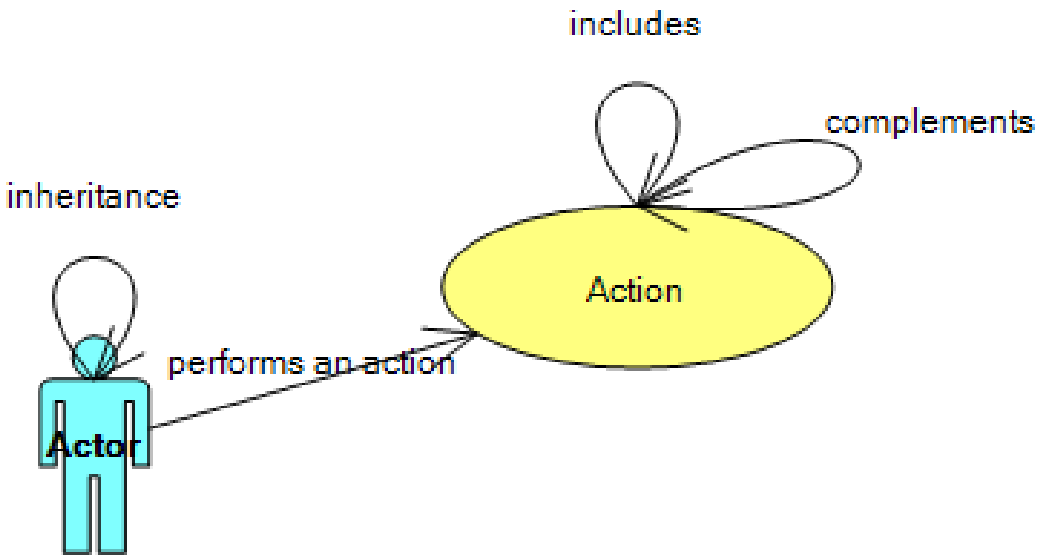


Figure 7. Metamodel of Use Cases Diagram

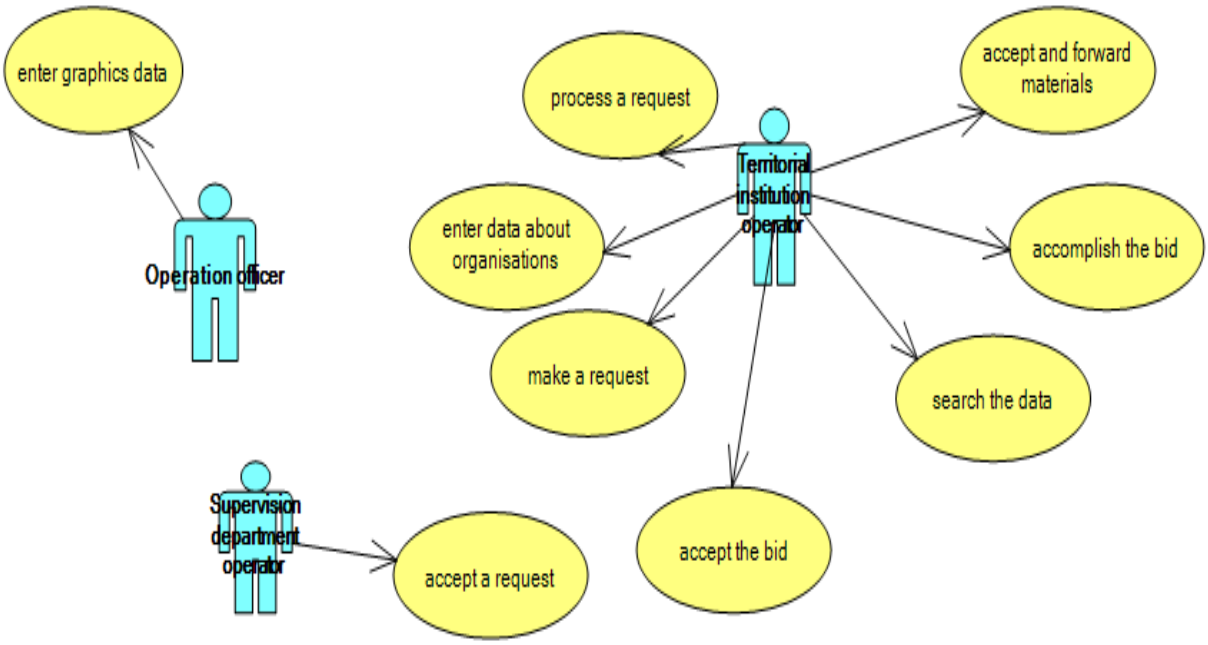


Figure 8. The result of the model «Application Processing» transformation – Use-Case diagram

Description of «model-to-model» transformation does not require any knowledge of any specific programming language. All operations are implemented in the same environment and at the same level of skill requirements of users as to the development of languages and models.

At the research project also were described the rules and made the transformation of the type «model-to-text» to generate code in the language of SQL. The resulting code can be used for creation of the integrated information system database.

Conclusion

The research project has shown that the modern DSM platforms can be used not only for the development of domain-specific languages and models, but also as a means of integration of various informational and analytical systems. Earlier domain-specific modelling languages were developed for creation of simulation models, transformation rules were described and transformations of the developed models for simulation experiments in GPSS were executed.

The ability to dynamically change of languages and models, their configuration in the development of informational and analytical systems can significantly reduce the complexity of problems solving of modeling due to re-using previously established models, automation of their transformations.

In addition, the modeling tools, based on using language toolkits that provide advanced tools for the development of languages and models, do not need high requirements for the qualification of users; these workbenches are available for different categories of users.

Bibliography

[Cook, 2007] S. Cook, G. Jones, S. Kent, A.C. Wills. Domain-specific Development with Visual Studio DSL Tools. Reading: Addison-Wesley, 2007. – 560 p.

[Gronback, 2009] R.C. Gronback. EclISse Modeling Project: A Domain-specific Language Toolkit. In: Reading: Addison–Wesley, 2009. – 706 p.

[Kelly, 2013] S. Kelly, K. Lyytinen, M. Rossi, J.P. Tolvanen. MetaEdit+ at the Age of 20. In: Seminal Contributions to Information Systems Engineering. Springer, 2013. P. 131-137.

[Lubyagina, 2014] A.O. Lubyagina. An approach to the functional integration of technology-based MDE. In: Izvestiya of SFU. Engineering science. – 2014. – № 6 (155). P. 159-163.

[Lyadova, 2008] L.N. Lyadova. Multilevel models and languages DSL as a basis to create intelligent CASE–systems. In: Proceedings of international scientific and technical conference «Intelligent systems» (AIS'08) and «Intelligent CAD» (CAD–2008). A scholarly edition in 4 volumes. T. 2. Moscow: Fizmatlit, 2008. P. 37- 41.

[Lyadova, 2012] L.N. Lyadova, A.P. Seriy, A.O. Sukhov. Approaches to the description of the vertical and horizontal transformations metamodels. In: Mathematics of Program Systems: Collection of scientific papers. – Perm State University, 2012. – Vol. 9. – P. 33-49.

[MetaCase, 2014] MetaEdit+ Version 5.0: Workbench User's Guide. [http://metacase.com/support/50/manuals/mwb/Mw.html] (Checked: 18.04.2014).

[Overview, 2014] Overview of Domain-specific Language Tools. [http://msdn.microsoft.com/en–us/library/bb126327.aspx] (Checked: 18.04.2014).

[QReal, 2014] QReal: BP. [http://qreal.ru/static.php?link=QRealBP_usage] (Checked: 18.04.2014).

[Sorokin, 2010] A.V. Sorokin, D.V. Koznov. Overview of EclISse Modeling Project. In: System Programming. MY. 5 S. 2010.

[Suhov, 2012] A.O. Sukhov. Methods for transforming visual models. In: Proceedings of the III International Scientific and Technical Conference «Technologies of Information Systems Development» (TRIS–2012). Taganrog Technological Inst. of SFU, 2012. – Vol. 1. – P. 120-124.

[Suhov, 2012] A.O. Sukhov, A.P. Seriy. Using graph grammars for model transformation. In: Proceedings of the conference «CSEDays 2012». Ekaterinburg: Ural State University, 2012. – P. 48-55.

[Suhov, 2013] A.O. Sukhov. Tools for creating visual Domain-specific modeling languages. In: Fundamental research. – 2013. – № 4. – P. 848-852.

[Terekhov, 2013] A.N. Terekhov, T.A. Bryksin, Y. Litvinov. QReal: platform visual object-oriented modeling. In: Software Engineering, 2013, № 6. P. 11-19.

[Zamyatina, 2013] E.B. Zamyatina, L.N. Lyadova, A.O. Sukhov. An approach to the integration of simulation and information systems on the basis of DSM-platform MetaLanguage. In: Proceedings of the IV International Scientific and Technical Conference«Technologies of Information Systems Development» (TRIS–2013). Taganrog Technological Inst. of SFU, 2013. – Vol. 1. – P. 61-70.

Authors' Information

Anna Lubyagina *– National Research University Higher School of Economics, Department of Business Informatics, student; 38, Studenceskaia St., Perm, 614070, Russia; e-mail: lubyagina@prognoz.ru.*

Major Fields of Scientific Research: Business informatics, Business process modelling, Analytical systems, System analysis.

Lyudmila Lyadova *– National Research University Higher School of Economics, Department of Business Informatics, associate professor; 38, Studenceskaia St., Perm, 614070, Russia; e-mail: LNLyadova@gmail.com, LLyadova@hse.ru.*

Major Fields of Scientific Research: Information systems development, Modelling languages, Domain specific modelling, CASE-tools, Graph models.

Sukhov Alexander *– National Research University Higher School of Economics, Department of Business Informatics, teacher; 38, Studenceskaia St., Perm, 614070, Russia; e-mail: ASuhov@hse.ru.*

Major Fields of Scientific Research: Software Engineering, Modelling languages, Visual modelling, Domain specific modelling, Domain specific language, Language workbenches