AN APPROACH TO MULTIFACETED BUSINESS PROCESS MODELING WITH MODEL TRANSFORMATION TOOLS

Roman Nesterov, Lyudmila Lyadova

Abstract: *The approach to models generation automation and implementation of multifaceted business process modeling on the basis of graphical model transformation is described. To create graphical models of diverse notations (diagrams in notations of visual modeling languages) one can exploit visual modeling software tools and language workbenches, DSM platforms. Domain specific modeling tools allow simplifying model design process, to involve domain experts (they are not masters of information technologies and have not programming skills) to formal model development. Newly-created models can be converted into simulation models or specific analytical models with the model transformation tools. Therefore, at new task solving process with modeling tools modelers have not to duplicate model development with new tools in new language notation. Model designers can use most suitable tools and most expressive languages for models development in their domain to solve their tasks. Obtained models after transformation can be examined with means of specific simulation modeling systems including, for instance, AnyLogic, or with mathematical software packages such as Mathcad, Maple or Mathematica. The visual business process modeling notation choice is substantiated. Mathematical model named DFD-graph is used as mathematical basis of model generation tools. The normalization rules form the backbone to the DFD business process model normalization. This algorithm is the basis of automating model generation software implementation.*

Keywords: *business process modeling, visual modeling languages, business process analysis, mathematical modeling, model development automation, model transformations, model reusing.*

ACM Classification Keywords: *D.2 SOFTWARE ENGINEERING: D.2.2 Design Tools and Techniques – Computer-aided software engineering (CASE), Programmer workbench; D.2.13 Reusable Software – Domain engineering, Reuse models. I.6 SIMULATION AND MODELING: I.6.2 Simulation Languages; I.6.3 Applications; I.6.4 Model Validation and Analysis; I.6.5 Model Development. G.4 MATHEMATICAL SOFTWARE: Algorithm design and analysis, User interfaces.*

Introduction

The efficient company management is impossible without employing informational and analytical systems being created and functioned following the business process models developed by analysts. The process of modeling itself has two tasks: model development (in terms of visual language notations or mathematical constructs) and model examination with tools meeting the needs identified by analysts and supporting the decision-making process. Analysts and domain experts need a great variety of analytical tools for the multifaceted exploration of business systems and processes via different models expressing various aspects to be studied to solve decision-making tasks. These models are based on the formal notations varied in form or in mathematical apparatus suitable for analytical tasks solving.

Nowadays there exists a lot of diverse visual (graphical) notations one can use to build visual business process models including, for example, IDEF0 notation used for identifying causal order of operations; DFD – for data flows representation; eEPC – for event-driven processes description et cetera. Different instruments and graphical model editors can be exploited for the development of visual business process models.

The modeling software choice often determines the modeling language or formal notation to be used while developing visual business process models. The same systems and processes can be described using various languages regarding the primary modeling objectives. Thus, analysts are forced to build models of object in question all over again while moving among modeling objectives achievement.

To conduct holistic model analysis by means of mathematical software toolkits (Matlab, Maple, Mathcad etc.) it is necessary to develop analytical models in terms of corresponding constructs and formats being utilized in those tools listed above. Moreover, analytical or numerical model development corresponding to high dimension systems to perform the analysis employing those mathematical software packages can prove to be non-trivial task: apart from model scope itself, another reason making the movement from visual business process model to its analytical representation can be the lack of data required for full-fledged model development. Notice, however, that having the high dimension visual business process model, described with means of chosen graphical notation, the movement to a corresponding analytical model representation being applicable for further examination can be automated.

There are partial solution to the problem of transformation of process and system models based on Petri net exploitations [Dorrer, 2011; Zhow, 2015], simulation modeling techniques and tools [Lantsev, 2013] and specific visual model transformation mechanisms [Poryazov, 2005; Poryazov, 2008; Poryazov, 2009].

This paper suggests to find a solution to the following tasks connected with model transformation aimed at further multifaceted business processes analysis execution:

1) to create the abstract graph model of business process relevant to the model proposed and developed by analyst in terms of the notation of the visual modeling language applied by domain expert to solve tasks of business process analysis;
2) to develop an algorithm transforming visual model, represented with chosen visual notation, into created on the first step abstract graph model;
3) to perform the transformation of the obtained abstract graph model into the classical Petri net;
4) to develop the queuing system model corresponding to the abstract graph model obtained on the first step.

These mathematical models (Petri net and queuing system) can give the comprehensive view of the business process in question, especially possible bottlenecks. In addition, their analysis can be automated and implemented with the help of mathematical software toolkits listed in this section.

An Approach Logic

Taking into account the existing transformation methods of visual business process models developed in the different notations, the general layout of abstract graph model of business process development is being proposed below. It is based on the model normalization and generic internal representation generation that can be transformed in its turn into different analytical representations according to the primary objectives.

The proposed approach includes following steps:

1. A visual business process models development by analysts in terms of chosen modeling language and notations relevant for analysis tasks.
2. An identification whether the visual model meets the requirements specified for the normalized visual model in a given notation.
3. The normalization process is to be implemented in case of any non-compliances identified at the previous step.
4. The generation of the business process model internal graph representation taking the normalized visual models as an input (model export).
5. The parse and loading the internal graph representation of the business process model into the environment of the chosen mathematical software tool.
6. An extension (refinement) of business process model definition (namely, determination of any control parameters of the model relevant to modeling goal) in the environment of the mathematical software

package exploited by analysts.

7. Automatically converting of the extended internal graph representation into an algebraic or differential equation system with means of tools offered by the chosen mathematical software package.
8. Finding the numerical or symbolic solution to the equation system obtained on the previous step.
9. A generation of a report on the business process analysis (the equation system solution output and analysis).

This proposed sequence of actions is supported by the implementation of algorithms aimed at the generation of analytical representation taking the visual business process model as an input.

The approach logic is shown in Fig. 1.

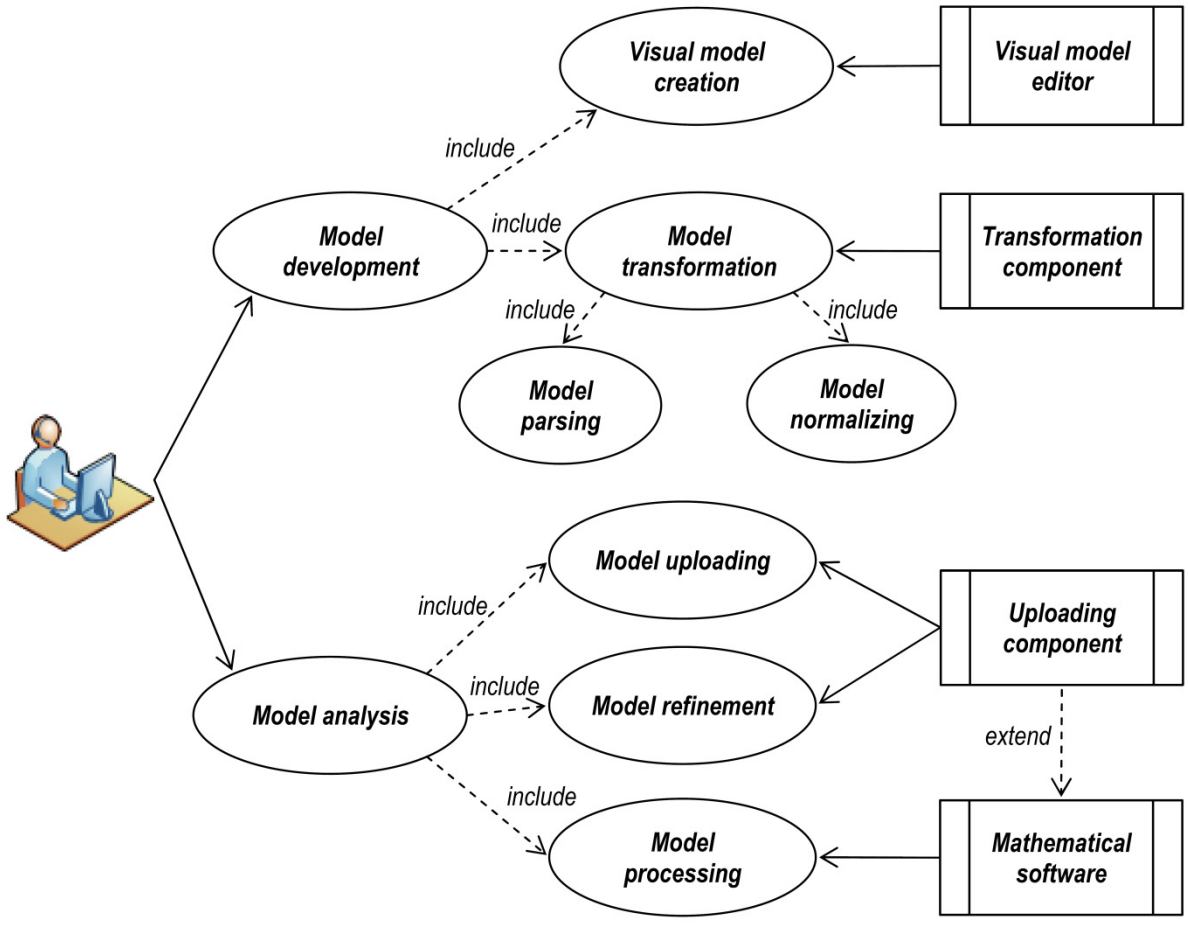


Figure 1. An approach to multifaceted business process modeling with model transformation tools and mathematical software

Furthermore, analysts can use DSM platforms to create domain specific languages (DSL) for the multifaceted business process modeling [Lyadova, 2014]. Domain specific modeling (DSM) tools allow to combine domain expert knowledge and model designer skills to develop and analyze models (Fig. 2). The language toolkits can become the basis for integration of various analytical tools (in particular, simulation systems). Maximal flexibility of modeling tools may be obtained with creating the multilevel models describing the researched systems and processes from various points of view and with different levels of details. For matching of various system descriptions it is necessary to develop the whole hierarchy of models (model, metamodel, meta-metamodel, etc.), where model is an abstract description of system characteristics that are important from the point of view of the modeling purpose, metamodel is a model of the language (DSL metamodel), which is used for models development, and meta-metamodel (metalanguage) is a language, on which metamodels are described in DSM platform.

Fig. 2 shows model development and analyzing with DSM platform MetaLanguage.

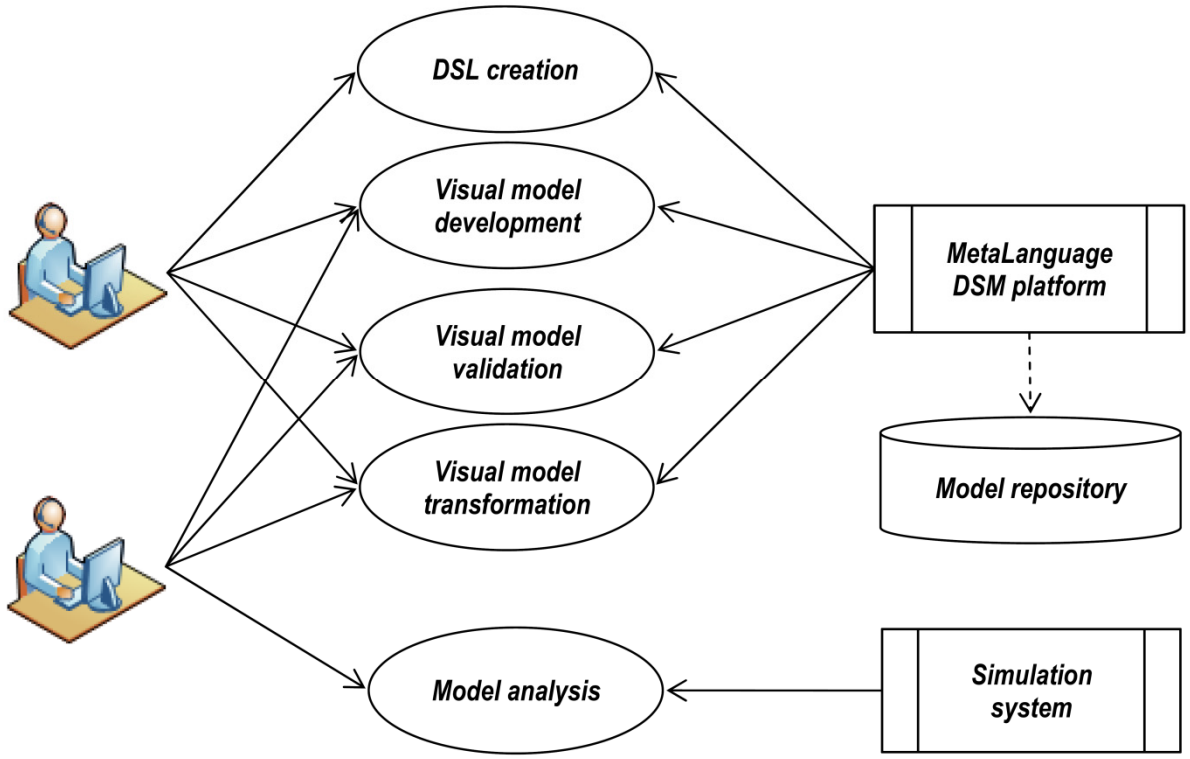


Figure 2. An approach to multifaceted business process modeling with model transformation tools and simulation systems

The Metalanguage system as an integration tool is presented in [Lyadova, 2014]. Transformations of business process models, described with MetaLanguage in various notations, into GPSS and AnyLogic models were realized.

The model editors and transformation tools allow to use various visual notations for business modeling. Therefore, analysts can choose more expressive and available modeling language to create models.

Business Process Modeling Notation Choice

To develop and pilot the proposed visual business process model transformation method to use mathematical software for model analysis it is necessary to choose the modeling language allowing an analyst to develop the model for business process of interest.

While choosing the graphical notation it is necessary to take into consideration the following:

1) the ability to yield the formal business process description;
2) the necessity to alter the developed model to implement the analysis with mathematical software toolkit instruments;
3) the ability to develop “end user friendly” business process model definition extension algorithm (business process control parameters specification).

Table 1 shows the results for comparison of different visual business process modeling notations against the listed criteria.

Table 1. The Comparison of Visual Business Process Modeling Notations

Notation	Possible indicators to examine	Mathematical apparatus
IDEF0	1) operation execution time; 2) actor time occupancy; 3) operation cost evaluation; 4) scenario probabilistic assessment	Algebraic equation systems

Notation	Possible indicators to examine	Mathematical apparatus
IDEF3	1) operation execution time; 2) operation cost evaluation; 3) scenario probabilistic assessment	Simulation modeling tools; Algebraic equation systems; Differential equation systems; Probabilistic evaluation models; Cols evaluation economical models
DFD	1) operation execution time; 2) operation cost evaluation; 3) a number of data access operation and time needed; 4) actor and external participant time occupancy; 5) data flow time characteristics; 6) the volume of data extracted and uploaded; 7) scenario probabilistic assessment	
BPMN	1) operation execution time; 2) operation cost evaluation; 3) actor time occupancy; 4) control flow time characteristics; 5) scenario probabilistic assessment	
eEPC	1) operation execution time; 2) operation cost evaluation; 3) actor time occupancy; 4) scenario probabilistic assessment	

According to the analysis results, DFD notation is chosen [Le Vie]. It is also chosen because it is not that “popular” notation exploited to model business processes and existing visual business process modeling tools offers too few tools to examine the models developed in terms of DFD notation. Analysts while building the models for business process mainly takes advantage of UML [Badreddin, 2010;

Bendraou, 2010; Hansen, 2012, *a*; Hansen, 2012, *b*; Mellor, 2002], IDEF3 or eEPC [Kim, 2001; Kim, 2003; Sterle, 2015]. These notations illustrate flow of operations, event chains and other related indicators. Nonetheless, an analyst can extract a lot of useful information that can be taken advantage of in the process of taking managerial decisions from extended DFD model defining it with control parameters.

The Main Business Process Indicators Identification

According to the DFD modeling capabilities main parameters values of which one can define or compute while processing the analytical representation of a visual business process models are listed below (the domain of interest is chosen regarding the usage of information resources while executing the business process defined with the help of DFD):

1) process is characterized by:
   a) execution time consisting of the time spent by all actors for executing this process operations;
   b) resource spend consisting of fixed and variable costs depending on the number of actors directly involved and time they spent;
   c) process run quantity generally defined by the input control parameters values and corresponding execution results;

2) flow is characterized by:
   a) speed/time of data movement (queries, requests, queries results et cetera) along the flow from one process to another or from a process to a data warehouse and vice versa;
   b) the amount of data moving across the flow of interest (only for flows connecting processes and warehouses);
   c) flow operation costs needed for data transmission organization including both fixed and variable ones;

3) data warehouse is characterized by:
   a) the number of extraction queries implemented by the processes;
   b) the number of update/insert queries implemented by the processes;
   c) the amount of data uploaded into the warehouses by the processes;
   d) the amount of data extracted from the warehouses by the process.

Apart from numerical indicators describing the specific business process DFD model parts, it is necessary to identify figures that can be computed using above listed control parameters as an input and that can help build the comprehensive picture of a business process in question, namely:

1) total execution time for one instance of a business process modeled with the help of DFD notation;
2) total amount of data uploaded and extracted from data warehouses according to queries done by the process parts of current instance of a business process in question;
3) total costs needed for the implementation process of one business process instance (including both fixed and variable costs).

It is necessary to mention that a part of all indicators cling to the currently studied business process is to be explicitly determined by the user. Thus, dependent parameters describing both single business process elements (processes, flows, data warehouses) and the business process on the whole is to be computed according to the mathematical and computing model described in the next section.

Mathematical Model of Business Process

The main formal definitions of business processes model obtained in terms of DFD visual modeling notation are listed below.

Definition 1. Graphical business process model M is a *DFD-model* if it can be represented as a directed marked graph as follows:

$$M = (P, D, F), \tag{1}$$

where $P = \{p_1, p_2, \ldots, p_n\}$ – a set of nodes representing *processes*, $D = \{d_1, d_2, \ldots, d_m\}$ – a set of nodes – *data warehouses*; and a set F represents *data flows* and being the union of two sets:

$$F = F_P \cup F_D, \tag{2}$$

where $F_P = \{f_{p1}, f_{p2}, \ldots, f_{pn}\}$ – represent flows between processes, $F_D = \{f_{d1}, f_{d2}, \ldots, f_{dn}\}$ – flows between processes and data warehouses.

Definition 2. Cross-process data flow in DFD-model M is a directed arc $f_{ij} = (p_i, p_j)$ from a subset F_P that meets: $p_i, p_j \in P$ and $p_i \neq p_j$.

So, the characteristic property for the elements of F_P is as follows:

$$F_P = \{(p_i, p_j) \mid p_i, p_j \in P, p_i \neq p_j\}. \quad (3)$$

Definition 3. "Process-Warehouse" data flow in DFD-model M is a directed arc $f_{ij} = (p_i, d_j)$ from the subset F_D meeting $p_i \in P$, $d_j \in D$.

Definition 4. "Warehouse-Process" data flow in DFD-model M is a directed arc $f_{ij} = (d_i, p_j)$ from the subset F_D meeting $d_i \in D$, $p_j \in P$.

The characteristic property for the elements of F_D is defined as follows:

$$F_D = \{ ((d_i, p_j) \mid d_i \in D, p_j \in P) \vee ((p_i, d_j) \mid p_i \in P, d_j \in D))\} \subset D \times P \cup P \times D. \quad (4)$$

Definition 5. Process p_i from a set P in DFD-model M is called *starting* if its in-degree is equal to 0, i.e. $d_{in}(p_i) = 0$. Starting process is defined by the means of subgraph $M_1 = (F_P, P)$.

Definition 6. Process p_i from a set P DFD-model M is called *terminating* if its out-degree is equal to 0, i.e. $d_{out}(p_i) = 0$. Starting process is also defined by the means of subgraph $M_1 = (F_P, P)$.

Definition 7. Business-process scenario defined via DFD-model M, is a path of the graph M connecting starting and terminating processes. Thus, scenario S_i ordered process subset $\{p_1, p_2, \ldots, p_k\}$ of the set of all processes P where each neighbor processes in S_i connected with cross-process data flow, i.e the set $\{(p_1, p_2), (p_2, p_3), \ldots, (p_{k-1}, p_k)\}$ is a subset to a set of all cross-process data flows F_P.

Fig. 3 shows the example of business process visual model, and Fig. 4 illustrates its corresponding abstract graph model M before additional marks for control parameters are introduced.

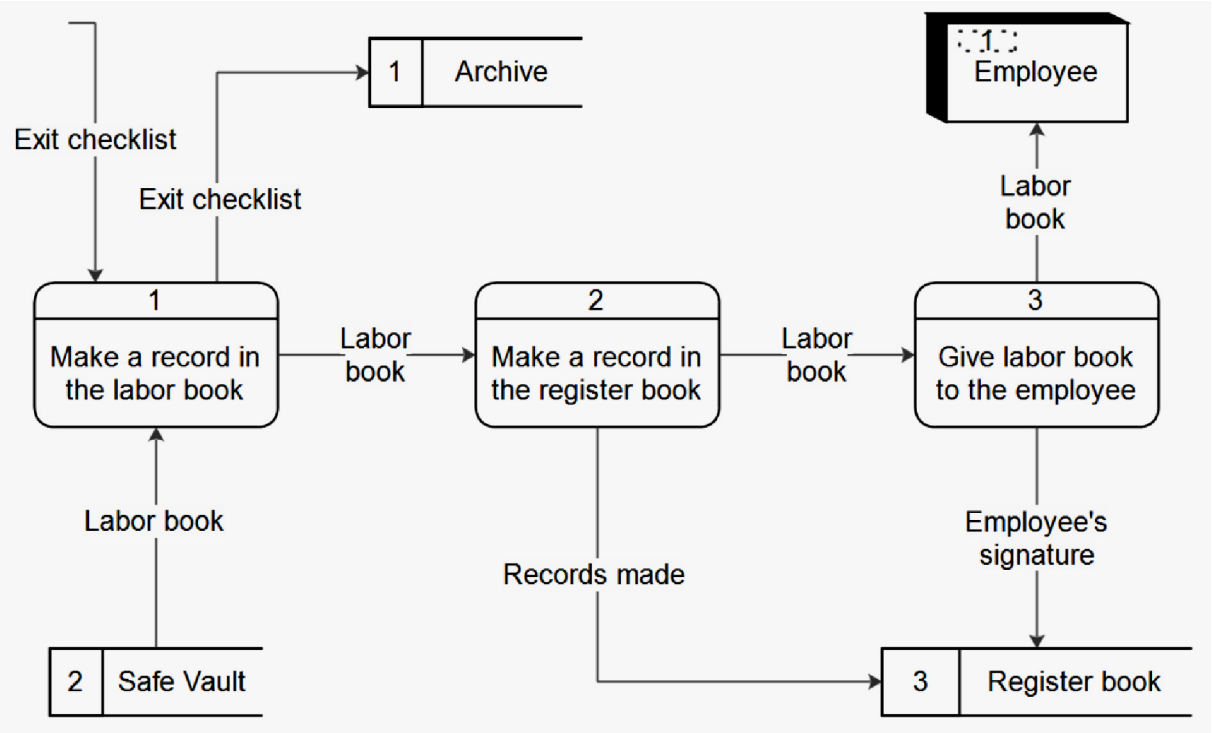


Figure 3. Visual DFD-model for a dismissal process

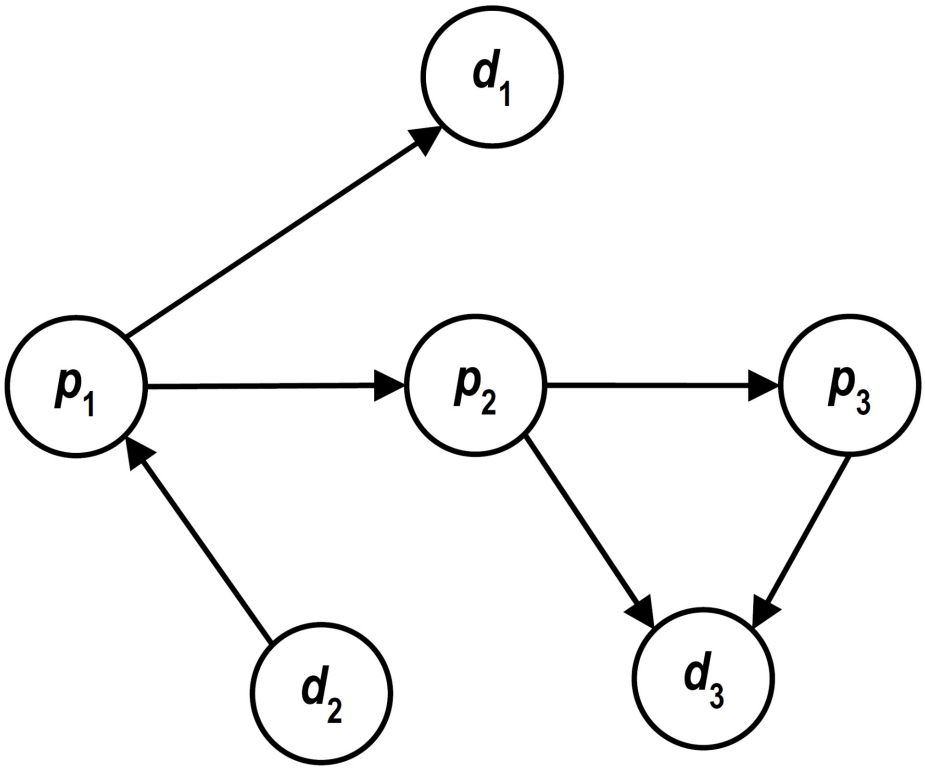


Figure 4. Dismissal process corresponding graph

Model Normalization Rules

As it was mentioned above, one of the main stages for the generation of business process analytical representation is normalization during that input model is checked to meet a set of predefined requirements. The set of normalization rules indicates the requirements for the visual DFD model of a business process providing the ability to develop the corresponding analytical representations pursuant to the formats listed in previous sections. Normalization rules is described in terms of definitions listed in the previous section.

Rule 1. DFD-model M can have only one starting process p_{start}. Otherwise, the model M is to be divided into several submodels having single starting processes.

Rule 2. DFD-model M can have at list one terminating process p_{finish}. Otherwise, it is necessary to introduce additional termination process node to the input graph M. The absence of a terminating process can be caused by graph cycles, that's why one need to introduce arc providing an exit from a cycles.

Rule 3. Each process of the DFD-model M is to be connected with another process p_j (at lest one), i.e.. Otherwise, the isolated process node not having connections with other process nodes is to be deleted.

Rule 4. Each process node p_i of the DFD-model M has to be connected with at least one data warehouse node d_j through the "Process-Warehouse" or "Warehouse-Process" data flows, i.e. $(p_i, d_j) \in F_D \vee (d_j, p_i) \in F_D$. Otherwise, the process node not having connections with warehouse nodes is to be deleted.

Rule 5. Each data warehouse node d_i of the DFD-model M is to have the connection with at least one process p_i across the "Process-Warehouse" or "Warehouse-Process" dataflows correspondingly. Otherwise, process node not having any connections with the process nodes is to be removed from the initial graph.

The normalization rules described above form the backbone to the DFD business process model normalization software implementation. Besides, the normalization algorithms is not described in this paper.

Figures Computing Technique

To facilitate the process of computing main process indicators and figures listed above, it is necessary to execute the additional business process abstract graph *M* transformation introducing arcs and nodes inscriptions (the input of this information is implemented by the user). Formal definitions to these parameters and definition extension algorithms will not be discussed in this paper.

To define the computing mechanism for business process indicators an initial abstract graph is to be divided into two subgraphs including a cross-process data flow subgraph $M_1 = (F_P, P)$ and a bipartite subgraph $M_2 = (F_D, P)$ indicating the interaction between processes and warehouses. Fig. 5 shows the example of graph division.

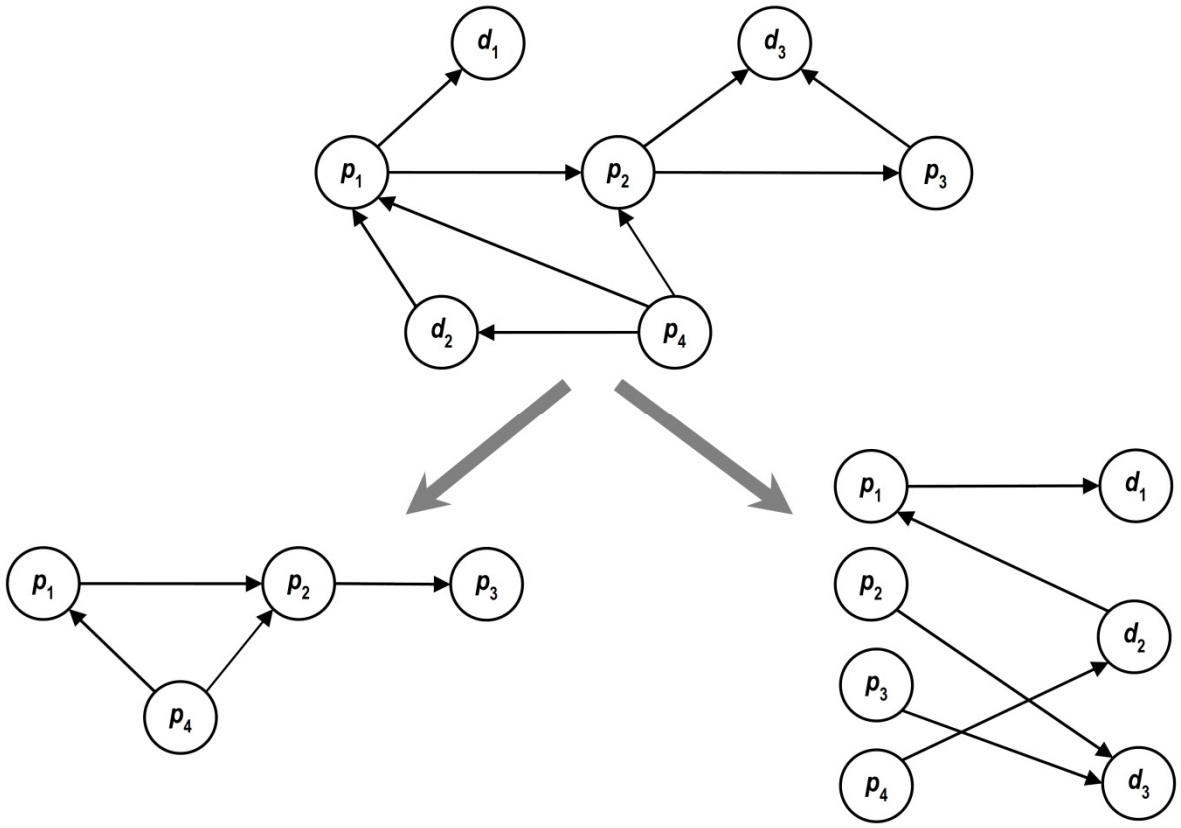


Figure 5. Initial business process abstract graph division

The cross-process data flow subgraph is used to analyze the business process implementation scenarios, and process-warehouse interaction subgraph is used to define and compute volume of information processed while business process instance operation.

As far as business process can be implemented through different scenarios, it is vital to inscribe each cross-process dataflow (p_i, p_j) with P_{ij} showing the probability of choosing this flow among the number of all available ones.

That is why, the total probability of business process implementation scenario S_i can be computed in the following way:

$$P(S_i) = \prod_{\substack{p_i, p_j \in S_i, \\ p_i \prec p_j}} (P_{ij}) \tag{5}$$

Formulas for each scenario characteristics computation are constructed (but these formulas aren't shown here):

- $T(S_i)$ – the execution time of the business process scenario S_i;
- $I(S_i)$ – the amount of information processed during implementation of the business process scenario S_i;
- $C(S_i)$ – the costs necessary for the execution of the business process scenario S_i.

By using the above mentioned extended graph and numerical indicators describing the single elements of the process, the following total business process (modeled with DFD-model M) figures computing algorithm were implemented:

- T_{total} – the total business process execution time;
- I_{total} – the total amount of information processed during the business process implementation;
- C_{total} – the total costs necessary for the execution of the business process in question.

The computing of DFD-model M above listed total figures, one need to get the values for the corresponding single business process element indicators.

Calculations can be executed with mathematical software.

DFD-model Transformation into a Classical Petri Net

To get more precise view on a possible problems that can occur while the business process instance implementation, one can exploit the apparatus offered by classical Petri net following the DFD-model transformation rules based on the correspondence of the DFD-model elements and Petri net constructs (see Table 2).

Table 2. Petri net and DFD-model correspondence

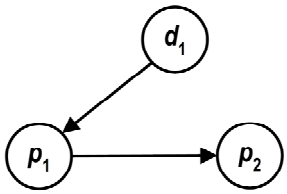
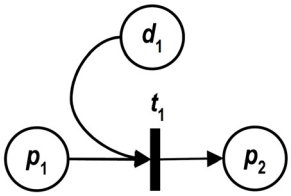
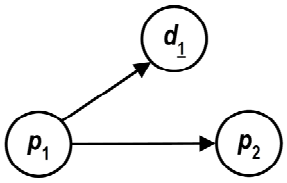
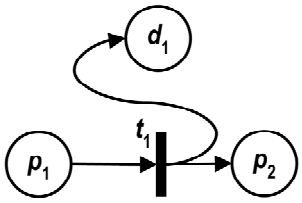
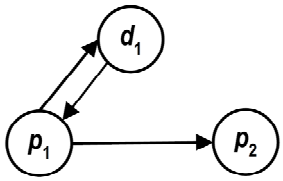
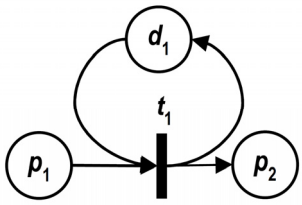
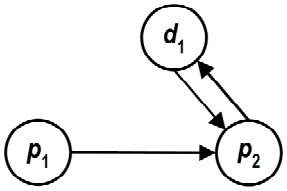
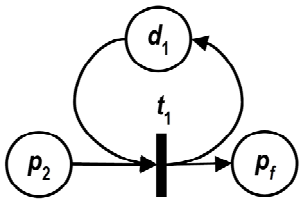
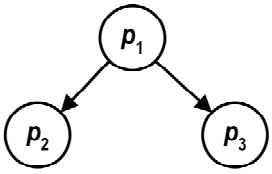
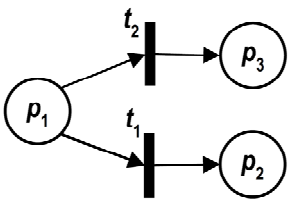
Element Meaning	DFD-model Elements	Petri Net Elements
Query data from a warehouse during process implementation		
Upload data to a warehouse during process implementation		
Simultaneous data query and upload while process implementation		
Data query and upload while terminating process implementation		
Alternative business process evolution		

Fig.7 shows the result of applying these transformation rules to a DFD-model abstract graph form a Fig.6.

The obtained Petri net will allow an analyst to perform additional analysis capabilities by means of Petri net specific algorithms: invariants, traps, deadlocks et cetera.

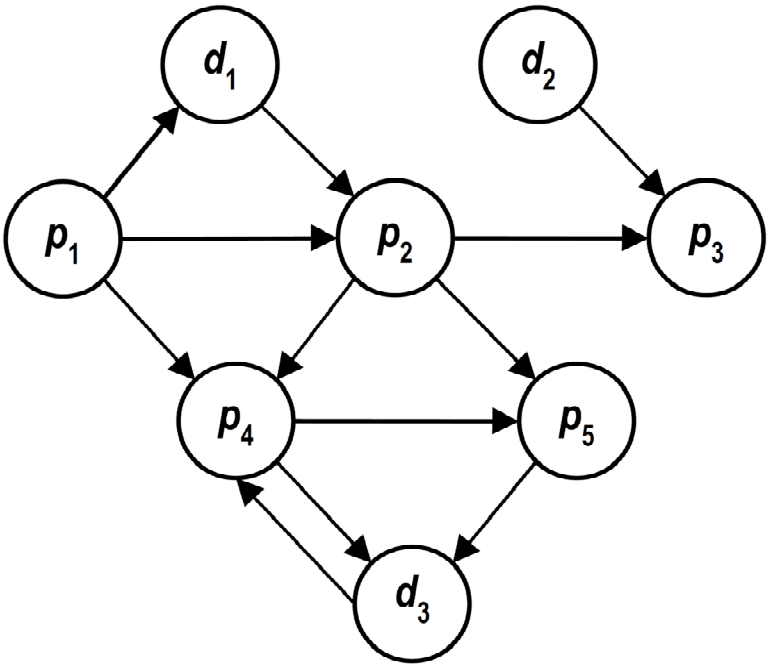


Figure 6. Abstract graph example of a DFD-model for a business process

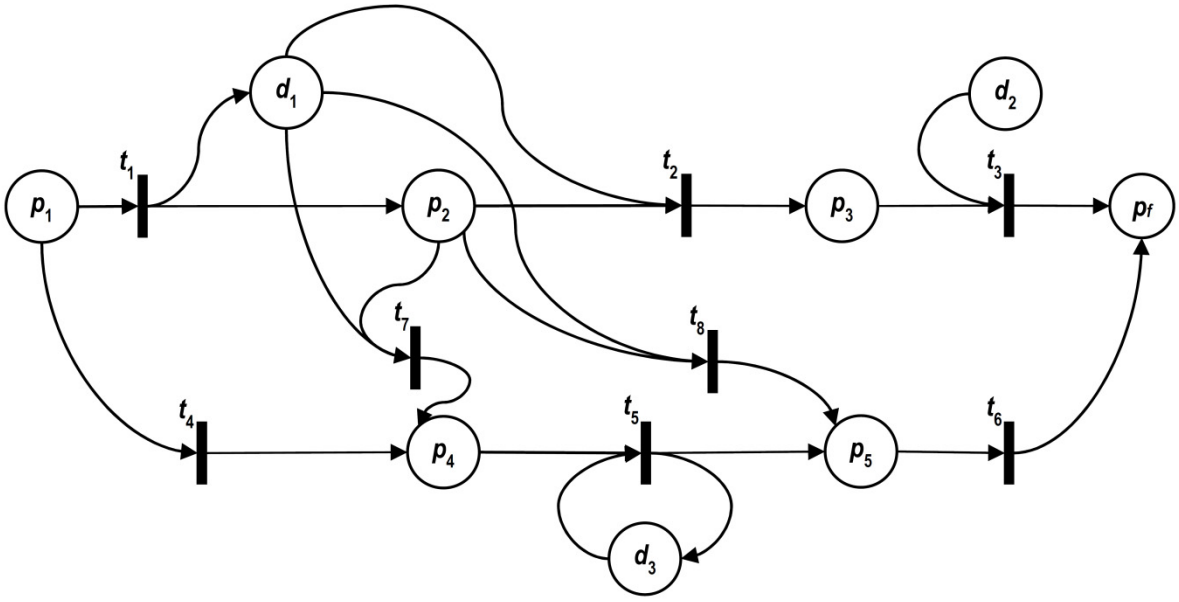


Figure 7. Corresponding Petri net

The generated Petri Net model can be uploaded to mathematical program to investigate its properties.

Queueing Model Application

Above described model do not resolve all the business process analysis problems. The other aspects concerning the business process modeled by using DFD notation can be examined with the help of queueing models, namely *process of destruction and multiplication*.

Within the framework of this model one can define the probabilities for a system being in one of the n possible states. The queuing system state S_i described by the occupancy of i queues from all n available ones.

The results obtained after the queueing model implementation can serve as a business process reengineering decision support. First of all, they can be applied to configure the queues work intensity, i.e. the probability of a system being in the idle state is to be decreased (when all queues are idle, or only the little part of them is occupied). Secondly, queueing modeling results can also help to redistribute the load among queues (more even possible system state probabilities distribution).

Software Implementation

Visual business process model transformation software research prototype has the following components implemented:

1) the visual business process model creation and its internal representation generation component;
2) the component for uploading and processing the internal representation into mathematical software tool Matlab.

The general scheme of the described above approach implementation is shown in Fig. 8.

The visual business process model creation and its internal representation generation component were implemented with the help of following instruments:

1) Microsoft Visual Studio 2012;
2) XML-file C# processing tools;
3) CASE-tool for automated model-driven development CASEBERRY (ICS company software);
4) ASP.NET MVC tools for model definition extension module development.

The component for uploading and processing the internal representation into mathematical software tool Matlab was implemented by means of following instruments:

1) Matlab Java integrated development environment;
2) Matlab graphical user interface development environment (GUIDE);
3) Java programming language XML-file processing tools.

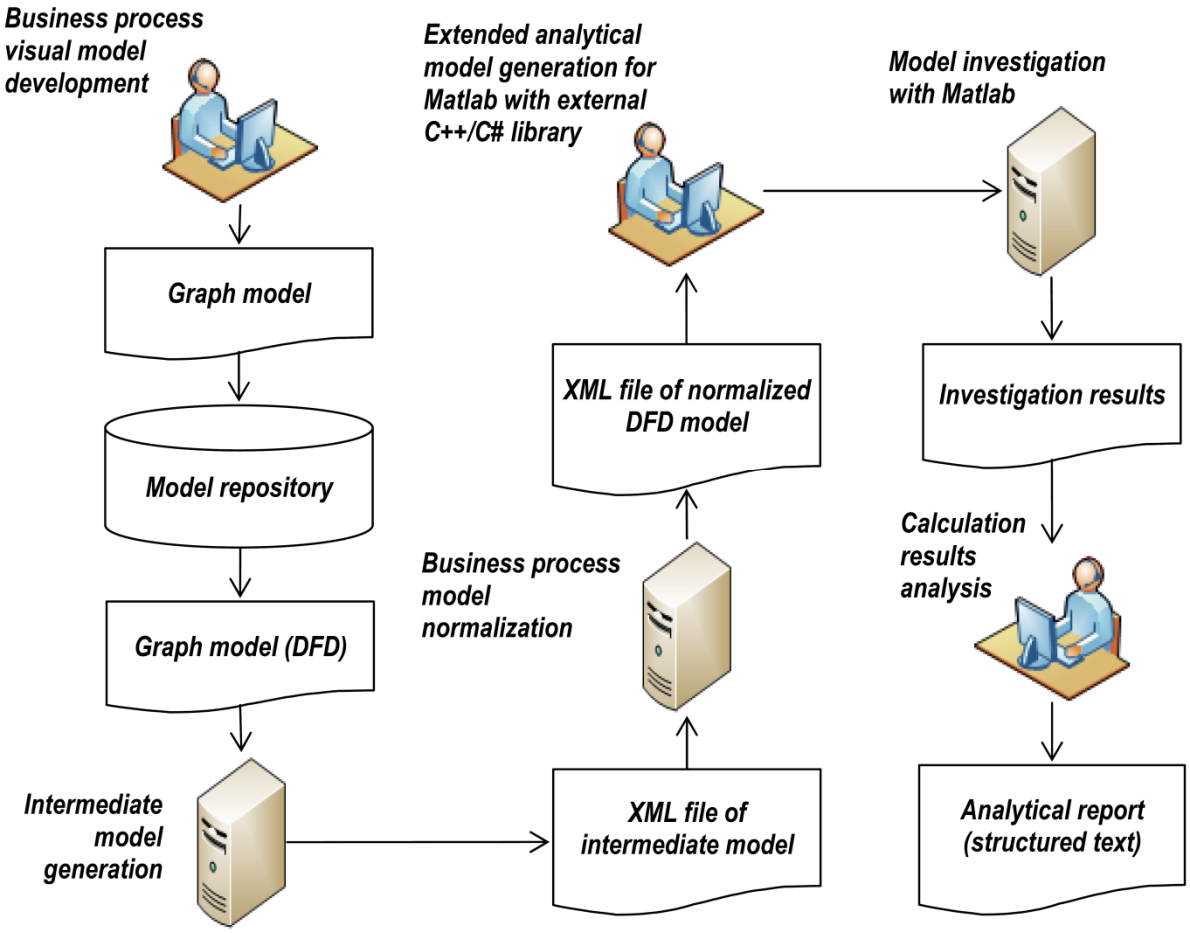


Figure 8. General implementation scheme

Conclusion

The research software prototype implemented shows the practical value for the proposed approach. The open system architecture allows to extend it with new components adding functionality including, for example, generation of other analytical representation types to study different business process aspects not presented in this paper so far.

It has to be mentioned that one can use DSM-platform MetaLanguage [Sukhov, 2013; Sukhov, 2014, *b*] to develop visual business process models of other types and notations and to transform and investigate diagrams.

Bibliography

[Badreddin, 2010] O.B. Badreddin. Umple: a Model-Oriented Programming Language. In: Proceedings of ACM/IEEE 32nd International Conference “Software Engineering”. 2010, vol. 2, pp. 337-338.

[Bendraou, 2010] R. Bendraou, J.-M. Jézéquel, M.P. Gervais, X. Blanc. A comparison of six UML-based languages for Foundation software process modeling. In: Software Engineering. 2010, vol. 36 (5), pp. 662-675.

[Hansen, 2012, *a*] H.H. Hansen, J. Ketema, B. Luttik, M.R. Mousavi, J. van de Pol. Towards model checking executable UML specifications in mCRL2. In: Innovations in Systems and Software Engineering. 2012, vol. 6 (1-2), pp. 83-90.

[Hansen, 2012, *b*] H.H. Hansen, J. Ketema, B. Luttik, M.R. Mousavi, J. van de Pol, O.M. dos Santos. Automated verification of executable UML models. In: Formal Methods for Components and Objects: Lecture Notes in Computer Science. Springer Berlin Heidelberg. Vol. 6957, 2012, pp. 225-250.

[Kim, 2003] C.-H. Kim, R.H. Westonb, A. Hodgsonb, K.-H. Lee. The complementary use of IDEF and UML modelling approaches. In: Computers in Industry. № 50, 2003, pp.35-56.

[Kim, 2001] C.-H. Kim, D.-S. Yim, R.H. Weston. An Integrated use of IDEF0, IDEF3 and Petri net methods in support of business process modelling. In: Proceedings of the Institution of Mechanical Engineers. Part E: Journal of Process Mechanical Engineering. Vol. 215. 2001, pp. 317-330.

[Le Vie] S. D. Le Vie. Understanding Data Flow Diagrams.
Available: http://ratandon.mysite.syr.edu/cis453/notes/DFD_over_Flow charts.pdf [April 22, 2015].

[Lyadova, 2014] L.N. Lyadova, A.O. Sukhov, E.B. Zamyatina. An Integration of Modeling Systems Based on DSM-Platform. In: Advances in Information Science and Applications. Volumes I & II. Proceedings of the 18th International Conference on Computers (part of CSCC '14). Ed.: E. B. Zamyatina. Vol. 1-2. Santorini Island : CSCC, 2014. P. 421-425.

[Mellor, 2002] S.J. Mellor, M.J. Balcer. Executable UML: A Foundation for Model-Driven Architecture. In: Addison-Wesley Professional, 2002.

[Poryazov, 2009] S. Poryazov. The overlaying free terminal states concept. In: Proceedings of a Conjoint Seminar “Modeling and Control of Information Processes”. 2009, pp. 110-116.

[Poryazov, 2008] S. Poryazov. Towards Useful Overall Network Teletraffic Definitions. In: International Journal “Information Technologies and Knowledge”. 2008, vol. 2, pp. 193-199.

[Poryazov, 2005] S. Poryazov. What is Offered Traffic in a Real Telecommunication Network? In: Proceedings of ITC19/Performance Challenges for Efficient Next Generation Networks. 2005, pp. 707-718.

[Sterle] M. Sterle. Intelligent Assistant for Simulation Model Generation from IDEF3 Process descriptions. Available: http://grantome.com/grant/NSF/IIP-9060443 [June 10, 2015].

[Sukhov, 2013] A.O. Sukhov, L.N. Lyadova. Horizontal Transformations of Visual Models in MetaLanguage System. In: Proceedings of the 7th Spring/Summer Young Researchers’ Colloquium on Software Engineering, SYRCoSE 2013 / Ed.: A. Kamkin.; Ed. by A. Petrenko, A. Terekhov. Kazan, 2013, pp. 31-40.

[Sukhov, 2014, *a*] A.O. Sukhov, L.N. Lyadova. An Approach to Development of Visual Modeling Toolkits. In: Advances in Information Science and Applications. Volumes I & II. Proceedings of the 18th International Conference on Computers (part of CSCC '14). Ed.: E. B. Zamyatina. Vol. 1-2. Santorini Island : CSCC, 2014. P. 61-66.

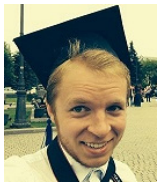
[Sukhov, 2014, *b*] A.O. Sukhov, L.N. Lyadova. Visual Models Transformation in MetaLanguage System. In: Advances in Information Science and Applications. Volumes I & II. Proceedings of the 18th International Conference on Computers (part of CSCC '14) / Ed.: E.B. Zamyatina. Vol. 1-2. Santorini Island, CSCC, 2014. pp. 460-467.

[Zhow, 2015] W. Zhow, F. Yang, Y. Zhu. A transformation method of OPM Model to CPN Model for System Concept Development. In: Proceedings of the First International Conference on Information Science and Electronic Technology (ISET), 2015, pp. 98-102.

[Dorrer, 2011] M.G. Dorrer. Algorithm for Transforming Models of Business Processes into Monochrome Petri Nets. In: Automatic Control and Computer Sciences, Vol. 45, No. 7, 2011, pp. 1-9.

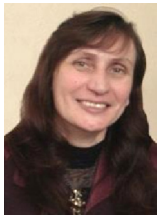
[Lantsev, 2013] Y.A. Lantsev, M.G. Dorrer. Creating agent-based model from the business process discrete-event model. In: St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunication and Control Systems. Vol. 3 (174), 2013, pp. 44-52.

Authors' Information



Roman Nesterov *– National Research University Higher School of Economics, Department of Information Technologies in Business, student; 38, Studenceskaia St., Perm, 614070, Russia; e-mail: RAnesterovHSE@gmail.com.*

Major Fields of Scientific Research: Software Engineering, Business Informatics, Business process analysis, Mathematical modeling, Domain specific modelling



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

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