VERSITA
Applied Computer Systems
doi: 10.2478/acss-2013-0004
2013/ 14

UML Sequence Diagram: Transformation from the Two-Hemisphere Model and Layout

Oksana Nikiforova[1], Ludmila Kozacenko[2], Dace Ahilcenoka[3], [1-3]*Riga Technical University*

***Abstract* – Modeling of the object interaction is one of the core tasks during system analysis and design, because it gives developer an ability to define responsibilities of class objects and to sketch general architecture of software components. In this task an ability of automatic generation of the UML sequence diagram becomes one of the most important activities. The two-hemisphere model contains enough information to define operations to perform by classes and therefore is investigated in this paper in the context of the UML sequence diagram generation.**

***Keywords* – two-hemisphere model, UML sequence diagram, model transformation, layout, BrainTool.**

I. Introduction

Model Driven Software Development (MDSD) is built on the principles of abstraction, modeling, reuse, and patterns, to provide software developers with an approach to identify and classify all of the system development activities and offer the usage of models and model transformations as a foundation for system development within every group of activities. Computation Independent Model (CIM) describes the system for its requirements analysis. System analysis results in platform independent model (PIM), which is then refined and transformed into platform specific model (PSM) to support the design activities in terms of software components. Then PSM is used to define code components and system implementation.

The primary benefit of MDSD is to give a big-picture view of the architecture of the entire enterprise. In order to use the MDSD approach, the developer should have a common modeling system. Nowadays, OMG's standard—Unified Modeling Language (UML) is widely used to represent system specification at different levels of system abstraction. UML defines a notation for a set of diagrams used for modeling of different aspects of the system (i.e., static and dynamic ones). The central part of static modeling in UML is class diagram, which defines the general structure of the system and serves as a basis for the development of software architecture. Class diagrams have been quite well studied in different researches. Problems associated with modeling of the system dynamic are the main reasons why software development still has not succeed in the model-driven way.

The central part of modeling system dynamic is the presentation of object interaction, where UML sequence diagram plays an important role and is used to present the system behavior. The problem, which is widely researched in the area of modeling of object interaction, is formal transition between the models presented at different levels of system abstraction. The transition from problem domain into system implementation expressed in terms of objects is required in the object-oriented software development using the principles of model-driven software development. Nikiforova in [1] proposes the way how classes and its object's operations can be defined based on so called two-hemisphere model and how this can be also potentially used for definition of the UML sequence diagram. This paper is the continuation of that research, where the authors are trying to perform a more profound analysis of the abilities given by two-hemisphere model for definition of the UML sequence diagram and focus also on the problem of automatic sequence diagram layout after its derivation from the two-hemisphere model. Since it is very important to ensure that the models and diagrams are well built not only in terms of their content but also how they visually represent the information, how they are layouted.

Layouting diagrams manually is a time consuming activity, and in case of large diagrams, it can be ineffective, therefore this paper is also about automatic UML sequence diagram layout, its problems and solutions. The authors discuss general diagram layouting criteria and criteria specially adapted for sequence diagrams. The authors also discuss the existing algorithmic layouting approaches and compare them. Since not all existing algorithmic approaches are suitable for the specific character of the sequence diagram, the author proposes a new algorithm taking into account the existing methods. As a result, definition of a two-hemisphere model received from the supporting BrainTool [2] is used to apply the authors' defined transformations for generation of the UML sequence diagram. Then the information of element placement and respective coordinates are added to the definition of the obtained sequence diagram, thus creating the complete XML file describing complete sequence diagram received directly from the two-hemisphere model specification. Visual representation of the received diagram is evaluated by importing the XML file into Sparx Enterprise Architect. In future, the algorithm can be implemented into BrainTool or in any UML compatible tool.

The paper is structured as follows. The next section describes notational conventions used in the UML sequence diagram and discusses several researches made for identification of the elements of the UML sequence diagram from different presentations of initial knowledge about the problem domain. The third section introduces an idea about using the two-hemisphere model for definition of the UML sequence diagram. The section provides theoretic information on the transformation itself, describes the source and target elements and explains the essence of transformation between

the two-hemisphere model and the UML sequence diagram. The fourth section focuses on the problems of the layout of the UML sequence diagram and describes the solution offered by the authors for placing objects, their lifelines and operations by taking into consideration several criteria for the sequence diagram layout. Practical experiment of using the transformations proposed and the layout algorithm offered in this paper is described in Section 5. The authors discuss the main contribution of the paper, highlight several conclusions and sketch the future direction for the research in the sixth section of the paper.

II. Background and Related Work

UML sequence diagrams allow describing interactions between system objects and actors of its environment. It presents sequences of communications that may occur in the course of a run of the system and traces the messages that are exchanged during this run. Sequence diagram is a popular notation to specify scenarios of the processing of operations as its clear graphical layout gives an immediate intuitive understanding of the system behavior. UML sequence diagram is stated as one of the ambiguous UML diagrams, with an implicit and informal semantics that designers can give to basic sequence diagram as a result of this conflict.

UML sequence diagram shows the objects, their lifelines, and messages to be sent by objects-senders and performed by objects-receivers. Sequence diagram is used to present the dynamic aspect of the system, which in object-oriented approach is expressed in terms of message transfer among objects. The dynamics of interactions is defined by ordering of the message sending and receiving actions. It serves the basis for definition of operations performed by objects to be grouped into classes, as well as to present and verify the dynamic aspect of class state transition.

There exist several methods and approaches for construction of the UML sequence diagram based on the information of problem domain, especially based on the system use case and their scenarios. For example, Visual Paradigm [3] offers semi-automated generation of the UML sequence diagram from the use case diagram; however user needs to add manually as much information about the use case as he would normally do creating UML sequence diagram from the beginning on his own. Also, Visual Paradigm [4] offers creating scenarios from activity diagram which can be later used to create UML sequence diagram. BPMN notation is used in a variety of tools to create business process models, however UML diagrams cannot be obtained from these models; user would need first to create UML diagrams and then link them to previously created BPMN models, e.g., Enterprise Architect [5] allows creating business domain models using BPMN; however, creation of the sequence diagram is manual. Several researches, e.g., [6], [7] offer analytical transformation of UML sequence diagram based on the data flow diagram and use case diagram. All these methods are semi-formal, semi-manual or manual and they do not support automatic generation of the UML sequence diagram directly from business level information.

As for the second problem discussed in the paper, namely, layout of sequence diagram, it is still an open question for the research, which needs to find how to achieve productivity in visual representation of sequence diagram. Fact, that the criteria for "good" UML and other diagrams are widely discussed in literature, proves the importance of diagram layout. Details on the related work are presented in Section 4 of this paper.

So far, the main goal of this paper is to try to solve both problems in construction of the UML sequence diagram. Authors suggest using elements of the two-hemisphere model to identify the elements of the UML sequence diagram and to define an algorithm for layout of these elements. Thus, automatic generation of the UML sequence diagram can be implemented inside the BrainTool for further importing into any MDSD support tool to complete the missing steps of the model transformation chain.

III. Definition of Transformation from the Two-Hemisphere Model into the UML Sequence Diagram

According to [8], transformation is the automatic generation of a target model from a source model, according to transformation definition. In our case as shown in Figure 1, the target model is the UML sequence diagram described in the first subsection, the source model is the two hemisphere model described in the second subsection, transformations are defined in the third subsection as is the transformation tool.

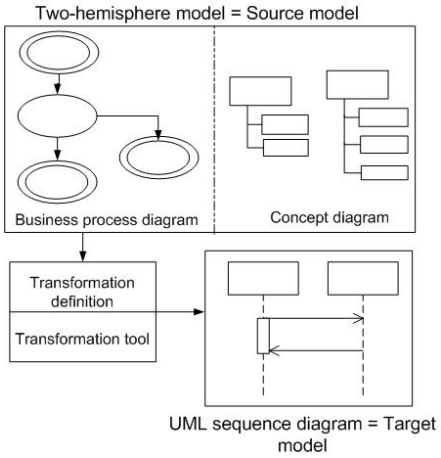


Fig. 1. The idea of transformation from the two-hemisphere model to the UML sequence diagram

A. Target model: the UML sequence diagram

Sequence diagram is one of the well-known UML diagrams, which represents system dynamic. Therefore, time plays the most important role and helps to organize messages in correct sequence. Vertical axis is used to display time, the beginning of the diagram is at the top and it is read downwards. Sequence diagram can consist of many different elements; however the authors will use only those, which can be acquired from the two-hemisphere model. Based on [9] some of the sequence diagram elements are:

- Object – is an instance of a class, which reflects real system object. In diagrams it is shown as rectangle.

- Lifeline – represents the time of object existence and participation in interactions. In the diagram lifeline it is shown as a dashed line (when the object is not active) or as a small rectangle (when the object is active) underneath the object rectangle.
- Actor - is a specific type of object, although it is not part of the constructing system, actor interacts with it. Graphically an actor is shown as a rectangle or as a human icon.
- Message - is used to show object/actor communication with another object/actor. It is displayed as an arrow; depending on message type an arrow line can be simple, dashed. Additionally, arrow shows the direction, indicating the sender and the receiver.
- Fragment - combines several messages into blocks. There are several fragment types, such as parallel, loop, alternative and etc. they define the way in which messages are executed.

A simplified sequence diagram metamodel presented in Figure 2 shows only those elements of the diagram and their dependences, which are being used in the transformation process, in other words, only those sequence diagram elements, which can be acquired from a two-hemisphere model.

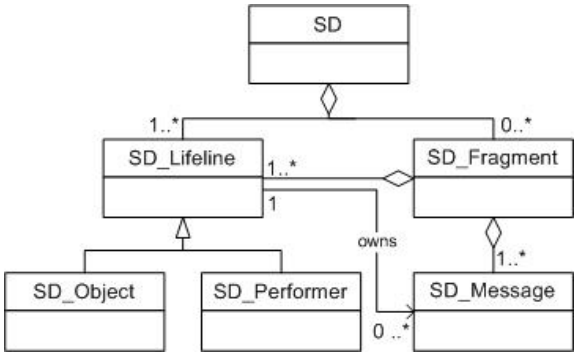


Fig. 2. Metamodel of the UML sequence diagram containing only transformable elements

B. Source model: the two-hemisphere model

For the first time the idea of how to use both the process and the concept models for sharing responsibilities between class objects was described in [10] and the title of the approach based on these two models as two-hemisphere model driven approach was announced in [11]. The main idea behind the two-hemisphere model driven approach is to represent all collected business level information in the formal way; therefore, a two-hemisphere model consists of the business process diagram and the concept diagram. Business process diagram shows the functional side of the system, while the concept diagram describes the structure of the system. The aim of two-hemisphere model is to use it as a basis for the next software development steps, i.e., design and implementation.

The use of the business process diagram as suggests [12] can be based on the researches, which prove that companies use different frameworks for business process control. Hence, enterprise employees are familiar with business process diagram notation. The author of [12] indicates that the concept diagram together with the business process diagram is used as a tool to check developers' understanding of semantic and procedural side of the domain. Besides, the concept diagram gives information about the system objects or entities and their attributes.

The author of [12] describes use of the two-hemisphere model driven approach to get the UML class diagram, later [13] proposes an idea to use the two-hemisphere model driven approach for creating the UML sequence diagram. Elaborating the idea of the two-hemisphere model driven approach further researchers' group [2] suggests the BrainTool, which realizes creation of the UML class diagram from the two-hemisphere model.

Business process diagram shows the processes that take place within the system that is being developed. The most important thing is that the diagram displays the processes in the correct order to achieve the goal. This research is connected with the BrainTool, therefore the BrainTool business process and concept diagram notations will be used. Figure 3 shows the metamodel of the two-hemisphere model defined during development of BrainTool.

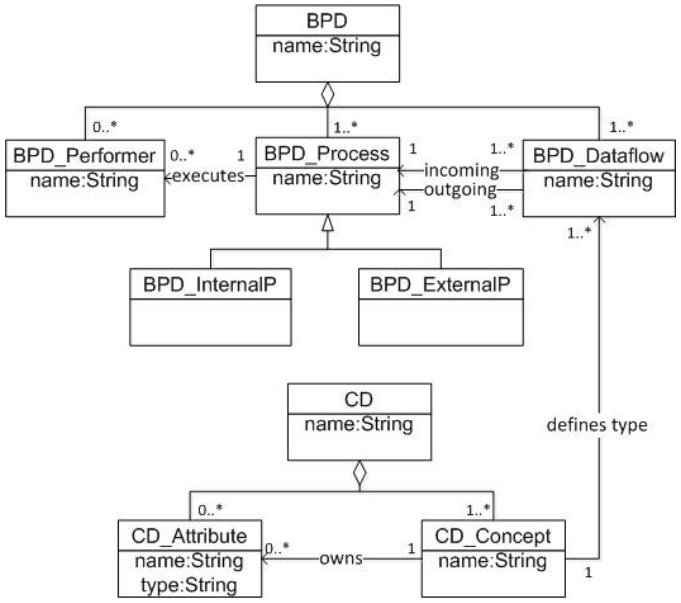


Fig. 3. Metamodel of the two-hemisphere model implemented in BrainTool

Based on the BrainTool business process diagram consists of:

- Internal process - the smallest business process unit. Business process consists of several internal processes; division into internal processes depends on the customer needs and the business process itself. Internal process in the BrainTool is displayed as an ellipse, with the name of the process written inside this ellipse.
- External process – the process, which occurs outside the business process, however it is necessary, because it delivers information to one or several internal processes. In the BrainTool external process is showed as an ellipse with 2

outlines (simple and dashed); the name of the process is also written in the ellipse.

- Data flow – shows information flow in the business process diagram. Data flow can connect internal or external process with other internal or external processes. Each process must have at least one incoming data flow to indicate, which kind of information is needed for the process to occur, and at least one outgoing data flow to indicate the result of the process. External processes are exceptions as they may not have incoming or outgoing data flows, e.g., the external process, which initiates internal process. BrainTool displays data flows as arrows.
- Performer – specifies agent of the process. BrainTool shows performers in brackets below the name of the process.

From the BrainTool concept model only the concept will be used in the transformation, which serves to reflect the potential class members. BrainTool shows concepts as named rectangles. Attributes also defined in the concept model are not used for definition of object interaction as they are modeled for system structure.

C. Transformation definition

The authors of [13] offer a rough idea of the transition from the two-hemisphere model elements to the UML sequence diagram elements, however there is no clear transformation definition. Table I proposes mapping between the two-hemisphere elements to UML sequence diagram elements.

UML sequence diagram contains information about object interaction in the right sequence, thus the two-hemisphere model should also have this information in order to get UML sequence diagram using transformation. The business process diagram of the two-hemisphere model is created in such manner that it becomes possible to determine, which process is at the beginning of the system and which process follows next. Every process has incoming and outgoing data flows; they can be attached to one or several concepts. Therefore, it is possible to define, which operation (process) should be send by which object (classified by concept) to which object (classified by concept). Still, neither business process diagram nor concept diagram provides information about the process (operation) type. Information on whether the message is synchronous or asynchronous cannot be retrieved from the process itself, and cannot be retrieved from the business process diagram. Thus, there will be no distinction between the message types in this discussion, however this problem stated to be solved in future researches.

TABLE I

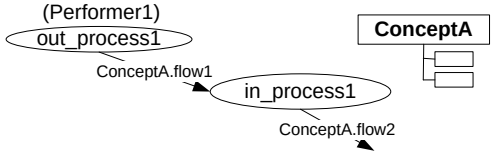
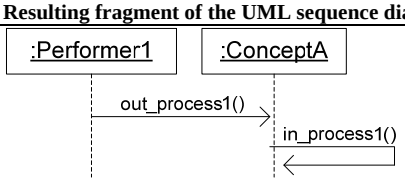
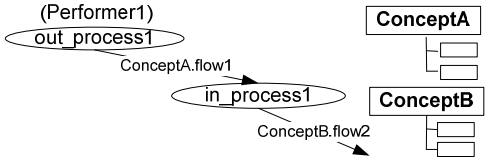
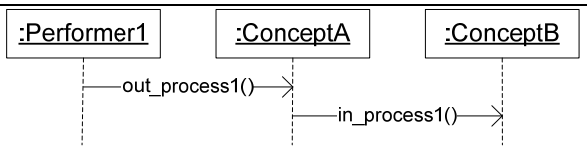
MAPPING OF THE SOURCE AND TARGET ELEMENTS

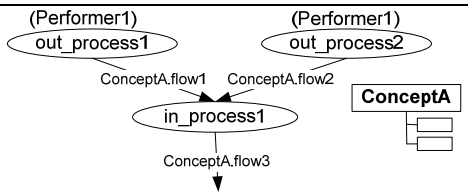
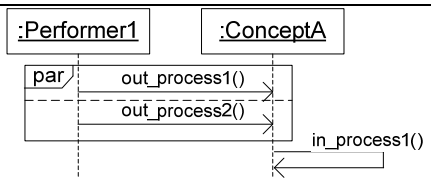
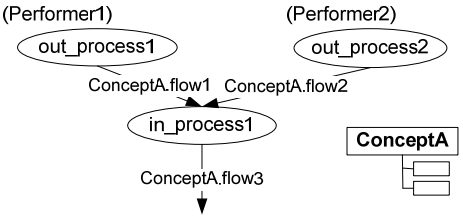
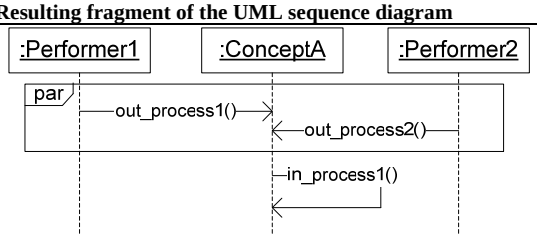
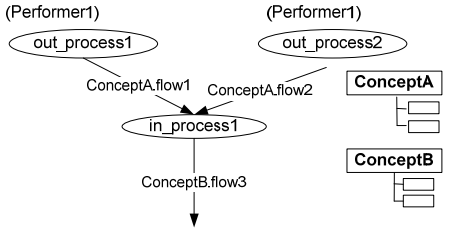
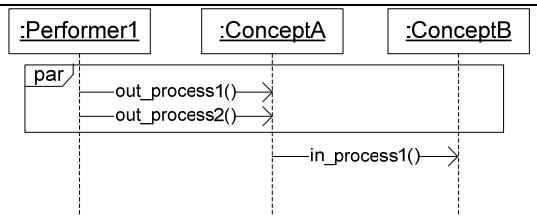
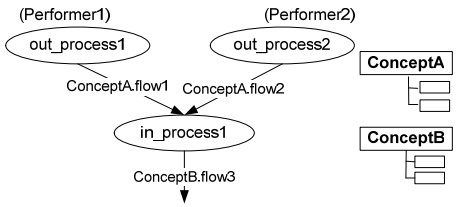
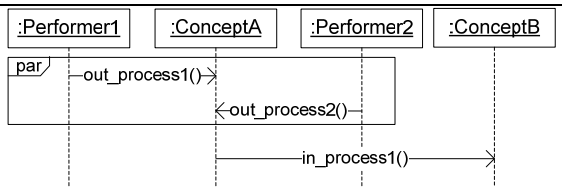
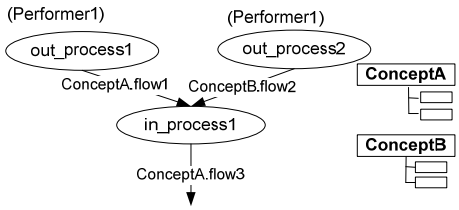
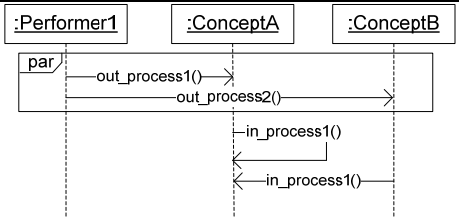
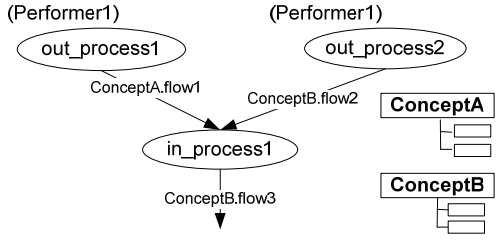
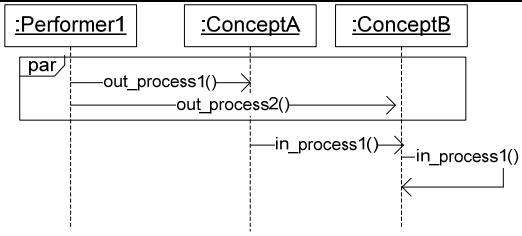
TWO-HEMISPHERE MODEL	UML SEQUENCE DIAGRAM
Internal process	Message
External process	Message
Performer	Actor
Data flow	Object
Concept	
Group of processes	Fragment

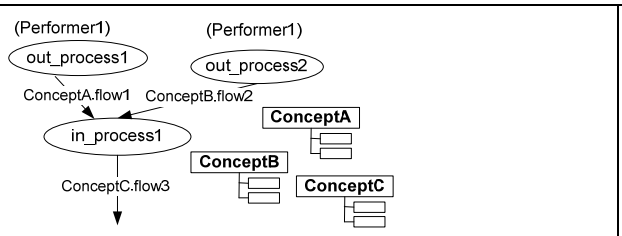
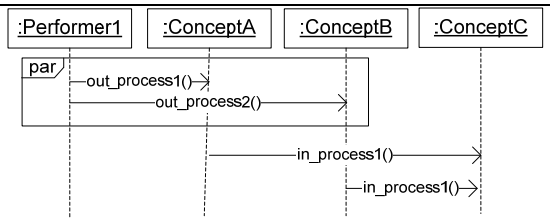
Table II adapted from [14] summarizes typical situations that may occur during creation of the two-hemisphere model and its subsequent transformation to the UML sequence diagram. Each row shows the transformation case of the fragment of the two-hemisphere model and the related fragment of the UML sequence diagram as a transformation result.

TABLE II

TRANSFORMATION CASES OF THE TWO-HEMISPHERE MODEL TO THE UML SEQUENCE DIAGRAM

Nr	Fragment of the two-hemisphere model	Resulting fragment of the UML sequence diagram
1.		
	Performer "Performer1" of the external process "out_process1" becomes an actor in the sequence diagram, which sends operation out_process1() to execute by object of class A, which is a concept of the concept model. Because flow1 and flow2 are related to the same concept A, they are expressed as A.flow1 and A.flow2 on the process model and the sender and receiver of the message "in_process1()" is the same object – object of the class ConceptA.	
2.		
	The difference from the first case is that incoming and outgoing data flows are related to different concepts. Hence message in_process1() is sent to another object – object of class ConceptB.	

3.		
	In case the two outgoing flows are from different external processes performed by the same performer the sequence of operations is defined in the parallel interaction fragment in respect to the UML notation.	
Nr	**Fragment of the two-hemisphere model**	**Resulting fragment of the UML sequence diagram**
4.		
	This use case is similar to the previous one. However, performers of the external processes are different, therefore the second actor Performer2 is displayed in the UML sequence diagram.	
5.		
	5th case is similar to the 3rd case. The difference is that flow1 and flow2 belong to the same concept, but flow3 belongs to another concept, therefore ConceptA sends message in_process1 to ConceptB.	
6.		
	This case is union of the 4th and 5th cases. There are different performers and also different concepts.	
7.		
	This case is similar to the 3rd case, however flow2 belongs to another concept. Message in_process1() is send two times according to the data flows.	
8.		
	8th case is similar to the 3rd and 5th cases, however there is some difference between senders and receivers of the message in_process1().	

9.		
	All data flows of this case belong to different concepts, therefore UML sequence diagram is much more complicated.	

The transformation provides only mapping of elements from source to target model. Layout of the model elements is another potential research problem domain, which is discussed in the next section of this paper.

IV. Several issues on the layout of the UML sequence diagram

Diagram is a convenient way to represent information and is much more comprehensible than textual information. Although diagrams can be used to present complex and difficult problems, they must be semantically and syntactically correct and well layouted to give a desirable result. A good diagram needs to satisfy different criteria int. al. aesthetic and layout criteria. General diagram criteria and specific UML diagram layout criteria have been studied by [15], [16], [17] , [18] and others. All diagrams should comply with general graph layout criteria.

General layout criteria result from the theory of perception. There are many such criteria (more than 20), but not all of them can be applied to all types of diagrams. Specific diagram like the UML sequence diagram has additional criteria, e.g., slidability. According to [15] there are six perceptual principles referring to organization of diagram elements, when the elements are considered as a group. These principles are acquired from Gestalt theory [19] and they are the following:

- Law of simplicity - the simpler the diagram, not containing too much information, the easier it is to understand .
- Law of similarity - elements of similar shape or color often can be perceived as a group.
- Law of continuation - elements are perceived together according to smoothest path. Those elements, which are connected with curved, complex lines are more difficult to perceive together.
- Law of proximity - close located elements are perceived as a group.
- Law of connectedness - if two elements are connected with a line or in other way physically they are perceived together as a group.
- Law of familiarity - elements, which seem to be familiar are perceived together.

There are three more principles related to perceptual element segregation, they are the following:

- Law of symmetry - diagram's symmetric parts are perceived as separated.
- Law of orientation - horizontally or vertically oriented elements are more likely to be perceived as separated figures than shapes in an angle.
- Law of contour - elements with contour are seen separately and all elements in this contour are perceived as a group.

All of these Gestalt theory principles are considered as aesthetic criteria. General aesthetic criteria are widely discussed in [20], [18], [15], [21], [22] and [17].

A sequence diagram is specific in its visual presentation. All the objects are allocated horizontally at the top of the diagram and the life lines are drawn vertically top-down. Therefore, the criteria for the UML sequence diagram should be carefully selected or even modified, so that they could be applied. E.g., one specific criterion for sequence diagram is correct sequence of messages, which is the meaning of this diagram. The authors of [18] and [15] have identified the criteria specific for sequence diagrams. Table III shows the list of these criteria in descending order of their importance. Criteria are marked with SD identifier.

TABLE III

Criteria for layout of the UML sequence diagram

ID	Name of the criteria	Description
SD0	Precise sequence of messages	Notational convention of the UML requires to display messages in the order they are being sent.
SD1	Avoid object and lifeline overlapping	When objects or lifelines are overlapping it is hard or sometimes impossible to read the diagram.
SD2	Elements need to be arranged orthogonally	Sequence diagram is an example of orthogonal diagram - message arrows are situated horizontally (typically) and lifelines - vertically.
SD3	Diagram flow	It is very important to layout elements by creating obvious flow - visible start and end of the diagram, easier to follow the elements and read the diagram. Usually the first message is located at the top left corner of sequence diagram.
SD4	Message arrow length minimization	To make the diagram more comprehensible and the area smaller, the message arrow length should be minimized
SD5	Reduction of long message arrow number	It is difficult to follow long message arrows, so they should be as few as possible.
SD6	Minimize longest message arrow length	The longest message arrow should be shortened if possible e.g. placing elements closer.
SD7	Minimize crossings	In the sequence diagram message arrows should not cross at all, therefore by crossings message arrow crossings over lifelines are understood and the number of this kind of crossings should be

		reduced.
SD8	Uniform message arrow length	Message arrows with similar length make diagram more understandable. Similar arrow length is also needed to fulfill the slidability criteria.
SD9	Improve slidability	Slidability is an aesthetic criteria for better clearness, particularly important in bigger sequence diagrams, where the whole diagram fails to fit in one screen. Slidability means that a fixed size window can be placed on the diagram and slid over it in such a way that all senders and receivers of messages, which are in this window fit in too.
SD10	Subset separation	Sequence diagram has subsets if there is one such message eliminating which, two unconnected sequence diagrams are formed.
SD11	Employ symmetry	[18] and [15] believe that symmetry should also be considered in sequence diagrams. However none of these authors define symmetry for sequence diagram, which is why this criterion will not be implemented.

Diagram layout plays an important role in the model driven approach because without correct layout system modeling is incomplete [16]. However diagram layout is not an easily executable task. Layouting diagrams manually is very time-consuming especially layouting large diagrams. There also are problems with automatic diagram layout, e.g., how good is the algorithm to fulfill different criteria and algorithm performance. The author of [23] believes that there are no suitable layout algorithms for more complex diagram types because there are problems with computing and communication.

The problem of automatic UML diagram layout still exists and it is widely discussed in relation to class and sequence diagrams. The cause of the layout problem is that algorithms are not well suited for each diagram type and there are many different aesthetic criteria to comply with. Some of the criteria are easier to implement than others, for example SD1, SD2, SD4-SD8, but there is no definition how to implement SD11 in algorithmic way [18]. Another problem in automatic layout is that many of the aesthetic criteria are conflicting, e.g., message arrow length minimization (SD4) and minimization of crossings (SD7), because reducing message arrow length is more likely to cause more crossings. The authors of [18] mention that the optimal layout is algorithmically complicated challenge, which is one more problem to automatic layout, for example an optimal linear layout problem is considered as NP-complete problem [24].

Since there are many different layout algorithms the author of [25] believes that a solution can be found by studying different possibilities to tailor algorithm to specific problem or combine several of them to get the expected results. Diagram layouting algorithms are based on graph theory and graph layouting algorithms [26]. The focus of this paper is on the layout algorithms looking appropriate for the UML sequence diagram. Algorithms can be divided into approaches, where the most used ones are topology-shape-metrics, hierarchical, visibility, divide and conquer, force-directed approaches; they are described in [20]. Genetic algorithms can also be used in diagram layouting.

Topology-shape-metrics approach is one of the most used ones [27]. Approach is suitable for orthogonal graphs and it supports many different aesthetic criteria [20]. Eichelberger in [23] mentions, that algorithms of this approach have been used to layout UML class diagrams and are implemented in such tools as GoVisual [28] and yWorksUML [29]. The approach has three main steps - planarization, orthogonalization and compaction, which are well described in [20].

Hierarchical approach, also called Sugiyama approach, is also used in UML class diagram automatic layout [30]. This approach is suitable for directed acyclic graphs - more or less hierarchic graphs, which is not the sequence diagram case.

Visibility approach is the general approach suitable for various types of graphs. It has been used in entity-relationship diagram layouting by [31]. The approach also has three main steps as mentioned in [23] and [20]. This approach can be put in the middle between both previously described approaches. Having studied this approach more closely the authors can conclude, that this approach is less suitable for the sequence diagram than topology-shape-metrics because of its second and third steps.

Divide and conquer approach first divides graph in parts, arranges elements and then merges these parts together [20]. Regarding to sequence diagram this approach is only suitable to diagrams with separable subsets therefore not suitable for all kinds of sequence diagrams.

Force-directed approach is suitable for undirected graphs [20]. Force-directed approach simulates physical system of forces, where a system tries to achieve the state of minimum energy. One of main criteria in this approach is minimization of crossings, which is not the most important criteria for sequence diagrams.

There is a wide range of genetic algorithms and they can be used for various purposes as was mentioned in [32]. Genetic algorithms simulate processes from nature, like mutations crossover and selection. Genetic algorithms were used in [17] for class diagram layouting and according to the research results these algorithms are time consuming (20 minutes for 17 class layoting).

Authors compared the relevance of each algorithm for the sequence diagram and genetic algorithms and topology-shape-metrics approach algorithms proved to be theoretically most suitable according to how they meet the sequence diagram criteria. Other approaches are not considered to be suitable at all because they either do not consider the right order of the priorities of criteria or they are not suitable for such diagram/graph type (e.g., they are tailored for undirected, acyclic types of graphs, but the sequence diagram is directed and cyclic).

There are several tools that provide automatic diagram layout, e.g. [15] mentioned tools Borland Together [33] (supports automatic UML sequence diagram layout, but uses lawless set of layout criteria) and "Rational Rose" [34] (supports UML class, but doesn't support sequence diagram layout). Sparx Enterprise Architect [5] is the tool that also provides automatic UML sequence diagram layout, however, it does not satisfy all the mentioned criteria of layout.

Considering the specificity of sequence diagram the authors propose to use an algorithm, which is based on topology-shape-metrics planarization step. Algorithm places elements as close as possible and tries to arrange communicating participants beside. Algorithm always provides SD0, SD1, SD2, SD3 and tries to fulfill SD4, SD5 and SD6 if possible.

V. Theory in use

Previous sections provide the summary of the theory on the transformation of the two-hemisphere model into the UML sequence diagram and discuss several layouting problems and their potential solution. The main idea is to implement transformation and layouting in order to get complete derivation and visualization of the UML sequence diagram. Figure 4 shows the conceptual framework of the steps being proposed by the authors.

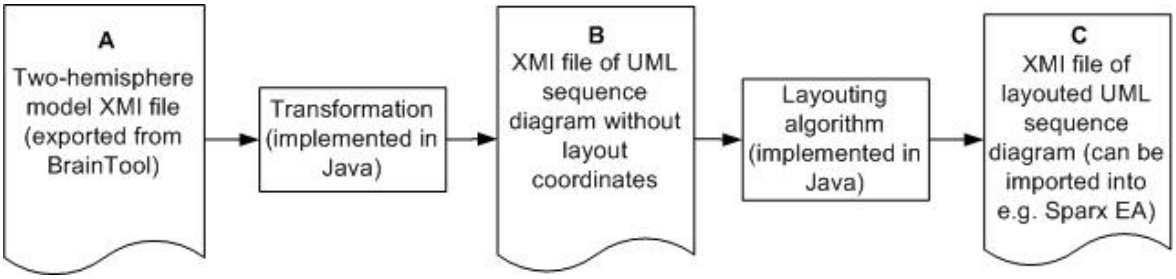


Fig. 4. Conceptual framework for proposed transformation and layout

First, one needs to take the two-hemisphere model expressed in the form of XMI. It is possible to create and to save the model by BrainTool [2].

Based on [35] transformation classification the direct-manipulation approach is used to transform the two-hemisphere model into the UML sequence diagram. Transformation definitions are written using general purpose programming language Java [36]. In order to apply the transformation proposed in Section 3 the XMI file of the two-hemisphere model is used as a source model and the result (or target model) is the XMI file of the UML sequence diagram.

The layout algorithm offered in Section 4 has been implemented also using the Java programming language. The algorithm is applied to XMI file describing the UML sequence diagram received at the previous step. Algorithm adds coordinates to the XMI file and any UML compatible tool, e.g., Sparx Enterprise Architect, can visualize the result. Figure 5 presents an example of the two-hemisphere model on the left side of the figure (marked by A similarly as in Figure 4) and the received sequence diagram on the right of the figure (marked by C in the same manner as in Figure 4).

Layout algorithm tries to satisfy as many criteria as possible. It calculates the distance between the elements considering lengths of messages and class object names. Algorithm places elements as close as possible by taking into account the diagram flow (e.g. interacting objects are being placed beside if possible). Pseudo code of the layout algorithm implemented is the following:

```
func layout_Elem(object_group obj[], message_group
msg[])
for i to object_count do
  element_beside=true
  for m to object_count do
    if obj[m].no_coordinates then
     for j to message_count do
       if (msg[j].sender=obj[m] AND msg[j].receiver=
       =last_added_obj)OR
       (msg[j].receiver=obj[m] AND msg[j].sender=
        =last_added_obj)then
          if(min_distance <msg[j].minLength)
            min_distance =msg[j].minLength
          element_beside=true
     if element_beside =true OR added_obj_count
     >obj_count*2 then
       if last_added_obj.right_dist > min_distance
       then
         min_distance = last_added_obj.right_dist
         min_distance = min_distance -
         -last_added_obj.right.dist
           if(min_distance <20) then
             min_distance =20
           obj[m].left =
           =last_added_obj.right+minBreak
           obj[m].right = obj[m].left+
           +obj[m].minWidth
           last_added_obj= obj[m]
           obj[m].has_coordinates
```

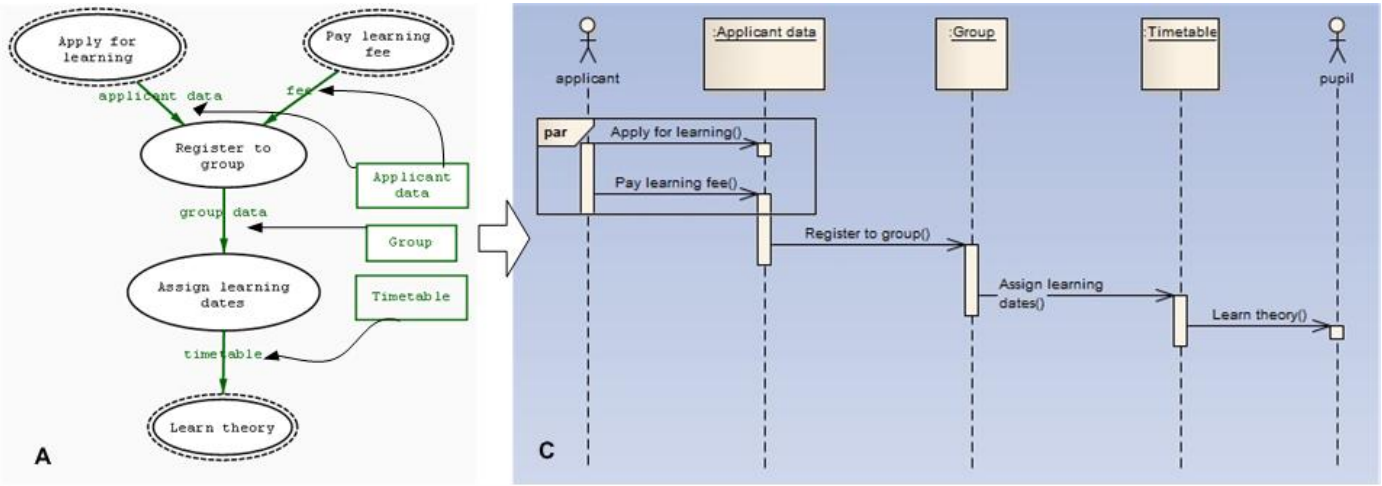


Fig. 5. Application of the transformation and layout algorithm

VI. Conclusion

In comparison with the traditional software engineering development methods the model driven approach provides software development based on models. Models are system abstraction; they are the main artifacts, which are used on each development step. Automatic model transformations are used to design and develop software systems in a more comfortable and faster way. Transformation takes the model created on one level of abstraction and converts it to the model on another level of abstraction. Numerous languages and tools exist, which support this kind of development process. However, it is still not possible to automate software implementation, because there exist several problems, which do not allow complete model transformation.

The research object of this paper is the UML sequence diagram. Both activities for its creation are investigated: they are element identification from the problem domain and visual representation (i.e., layout).

The contribution of the paper can thus be summarized as follows:

- Transformation cases for derivation of elements of the UML sequence diagram from the two-hemisphere model are defined in the formal way, thus they can be implemented in a modeling tool;
- A set of elements, which still are not transformable from the two-hemisphere model, is defined and allows the author to state the directions for the future research;
- An algorithm for the layout of the UML sequence diagram is developed and implemented, which pass the core requirements put forward to the object life-lines, messages and interaction frames.

The main conclusions of the research are the following:

- The two-hemisphere model contains sufficient amount of information about the problem domain to identify a variety of the elements of the UML sequence diagram.
- It is possible to define all the required transformations in the formal way; moreover, they can be implemented by general purpose programming language.
- The layout of the diagram is a complicated task due to large amount and diversity of the criteria that should be taken into consideration when placing elements in the diagram.
- Modeler cannot use convenient algorithms for graph presentation to layout the UML sequence diagram due to its specific structure, therefore some unique method should be applied.
- The quality of the layout algorithm strongly depends on the complexity of the diagram itself.

The transformations and layout algorithm offered in this paper can be introduced into BrainTool in order to expand its functionality in respect of the modeling of the UML sequence diagram. Analysis of mapping abilities of the two-hemisphere model with the UML sequence diagram indicates an ability to refine notational conventions of the two-hemisphere model in order to increase a variety of the elements of the UML sequence diagram. Both these topics can be stated as a direction for further research. Additionally, further research directions can include potential transformations from the two-hemisphere model to other types of UML diagrams, e.g. state charts, activity etc.

Acknowledgements

The research presented in the paper is partly supported by Grant of Latvian Council of Science No. 09.1269 "Methods and Models Based on Distributed Artificial Intelligence and Web Technologies for Development of Intelligent Applied Software and Computer System Architecture".

References

[1] O. Nikiforova, *Object Interaction as a Central Component of Object-Oriented System Analysis, International Conference „Evaluation of Novel Approaches to Software Engineering" (ENASE 2010), Proceedings of the 2nd International Workshop „Model Driven*

Architecture and Modeling Theory Driven Development" (MDA&MTDD 2010), Osis J., Nikiforova O. (Eds.), Greece, Athens, July 2010, SciTePress, Portugal, pp. 3-12

[2] O.Nikiforova, N. Pavlova, K.Gusarovs, O. Gorbiks, J. Vorotilovs, A. Zaharovs, D. Umanovskis, and J. Sejans, Eds., *Development of the Tool for Transformation of the Two-Hemisphere Model to the UML Class Diagram: Technical Solutions and Lessons Learned: Proceedings of the 5th International Scientific Conference „Applied Information and Communication Technology*, April 26-27, 2012, Jelgava, Latvia.

[3] Visual Paradigm, "Generate Sequence Diagram from Use Case Flow of Events", May 2011. [Online]. Available: http://www.visual-paradigm.com/product/vpuml/tutorials/gensdfromfoe.jsp [Accessed: Sept. 21, 2012].

[4] Visual Paradigm "Drawing activity diagrams". Available: http://www.visual-paradigm.com/support/documents/vpumluserguide/94/200/6713_drawingactiv.html [Accessed: Sept. 21, 2012].

[5] Sparx systems, "Enterprise Architect". Available: http://www.sparxsystems.com.au/ [Accessed: Sept. 24, 2012].

[6] A.A.A. Jilani, M.Usman, Z. Halim, " Model Transformations in Model Driven Architecture," *Universal Journal of Computer Science and Engineering Technology*, vol. 1, no. 1, pp. 50-54, October 2010. [Online] Available: UNICSE, http://www.unicse.org/. [Accessed May 28, 2012]

[7] M. Kardos, M. Drozdova, „Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA)," *Journal of Information and Organizational Sciences*, vol 34, no. 1, pp. 89-99, May 2010. [Abstract]. Available: JIOUS, http://jios.foi.hr/index.php/jios/article/view/163. [Accessed May 2005].

[8] A. Kleppe, J. Warmer and W. Ba, *MDA Explained: The Model Driven Architecture™: Practice and Promise* USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[9] K. Hamilton and R. Miles, *Learning UML 2.0*, USA: O'Reilly, 2006

[10] O.Nikiforova, Eds., *General framework for object-oriented software development process: Scientific Proceedings of Riga Technical University in series "Computer Science" vol.13.*, 2007, Riga, Latvia

[11] O.Nikiforova, M. Kirikova, Eds., *Two-hemisphere driven approach: Engineering based software development: Advanced Information Systems Engineering 16th International Conference, Series: Lecture Notes in Computer Science, Vol. 3084*, June 7-11, 2004, Riga, Latvia. Springer, 2004

[12] O.Nikiforova, "Two Hemisphere Model Driven Approach for Generation of UML Class diagram in the Context of MDA," *e-Informatica Software Engineering Journal*, vol. 3, no. 1. pp.60-74, 2009. [Online]. Available: e-Informatica Software Engineering Journal, http://www.informatik.uni-trier.de/~ley/db/journals/eInformatica/eInformatica3.html [Accessed September 28, 2012]

[13] O.Nikiforova, "System Modeling in UML with Two-Hemisphere Model Driven Approach," *Scientific Journal of Riga Technical University. Computer Sciences*, vol. 21. pp. 37-44, 2010. [Online]. Available: Versita, http://www.degruyter.com/view/j/rtucs.2010.41.issue--1/v10143-010-0022-x/v10143-010-0022-x.xml. [Accessed May 12, 2012]

[14] L. Kozacenko Research of Implementation Methods for Model Transformations, Bachelor Thesis, Riga Technical University, 2012

[15] K. Wong and D. Sun, *On evaluating the layout of UML diagrams for program comprehension: IWPC 2005, 13th International Workshop on Program Comprehension*, May 15-16, 2005, St. Louis, Missouri, USA. IEEE Computer Society 2005.

[16] A. Galapovs and O. Nikiforova, "UML Diagram Layouting: the State of the Art," *Scientific Journal of Riga Technical University. Computer Science. Applied Computer Systems*, vol. 47, pp. 101-108, 2011, [Online]. Available: Riga Technical University, https://ortus.rtu.lv/science/lv/publications/ [Accessed September 10, 2012].

[17] A. Galapovs and O. Nikiforova, *Several Issues on the Definition of Algorithm for the Layout of the UML Class Diagram*: *3rd International Workshop on Model Driven Architecture and Modeling Driven Software Development In conjunction with the 6th International Conference on Evaluation of Novel Approaches to Software Engineering*, June 8-11, 2011, Beijing, China. SciTePress Digital Library 2011.

[18] T. Poranen, E. Makinen and J. Nummenmaa, *How to Draw a Sequence Diagram*: *SPLST'03 Proceedings of the Eighth Symposium on Programming Languages and Software Tools*, June 17-18, 2003, Kuopio, Finland. University of Kuopio, Department of Computer Science 2003.

[19] B.E. Goldstein, *Sensation and Perception.* Wadsworth, 2002.

[20] [Battista u.c.1999] G. di Battista, P. Eades, R. Tamassia and I. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.

[21] H.C. Purchase, J-A. Allder and D. Carrington, "Graph Layout Aesthetics in UML Diagrams: User Preferences," *Journal of Graph Algorithms and Applications*, vol. 6, no. 3, pp. 255-279, 2002. [Online]. Available: Universitat Trier, http://www.informatik.uni-trier.de [Accessed Sept. 10, 2012].

[22] H. Eichelberger and K. Schmid, "Guidelines on the aesthetic quality of UML class diagrams," *Information and Software Technology*, vol. 51, no. 12, pp.1686-1698, 2009. [Abstract]. Available: ScienceDirect, http://www.sciencedirect.com. [Accessed Sept. 11, 2012].

[23] H. Eichelberger, "Aesthetics and Automatic Layout of UML Class Diagrams," PhD dissertation, Julius-Maximilians-Universität, Würzburg, Germany, 2005.

[24] M. Garey and D. Johnson, *Computers and intractability - A Guide To The Theory Of NP- Completeness*. W.H. FREEMAN AND COMPANY. - 1991.

[25] D. Ahilcenoka Research of the Problems and Solutions for the Layout of the UML Sequence Diagram, Bachelor Thesis, Riga Technical University, 2012.

[26] K. Freivalds, U. Dogrusoz and P. Kikusts, *Disconnected Graph Layout and the Polyomino Packing Approach:GD 2001 9th International Symposium on Graph Drawing*, September 23-26, 2001, Vienna, Austria. Lecture Notes in Computer Science, Springer-Verlag, 2002.

[27] J. Sun**,** *Automatic, Orthogonal Graph Layout.*, Project work, Hamburg University of Technology, 2007.

[28] Oreas optimization, research and software, GoVisual Diagram Editor, [Online]. Available: http://www.oreas.com/gde_en.php. [Accessed April 10, 2012].

[29] yWorks, Automatic Layout of Networks and Diagrams, 2012, [Online]. Available: http://www.yworks.com/en/products_yfiles_practicalinfo_gallery.html. [Accessed April 10, 2012].

[30] J. Seemann, *Extending the Sugiyama Algorithm for Drawing UML Class Diagrams: Towards Automatic Layout of Object-Oriented Software Diagrams*: *GD '97, Graph Drawing, 5th International Symposium*, September 18-20, 1997, Rome, Italy. New York: Springer Verlag, 1997.

[31] R. Tamassia, *New Layout Techniques for Entity-Relationship Diagrams: Entity-Relationship Approach: The Use of ER Concept in Knowledge Representation, Proceedings of the Fourth International Conference on Entity-Relationship Approach* October 29-30, 1985, Chicago, Illinois, USA. IEEE Computer Society and North-Holland, 1985.

[32] M. Mitchell, *An Introduction to Genetic Algorithms*. A Bradford Book, 1999.

[33] Borland a micro focus company, Borland Together, [Online]. Available: http://www.borland.com/products/Together/. [Accessed Sept. 2, 2012].

[34] IBM, Rational Rose product family, [Online]. Available: http://www-01.ibm.com/software/awdtools/developer/rose/. [Accessed Sept. 2, 2012].

[35] K. Czarnecki and S. Helsen, "Feature-Based Survey of Model Transformation Approaches," *IBM Systems Journal*, vol. 45, no. 3, p. 621, 2006. [Abstract]. Available: IEEE Xplore, http://ieeexplore.ieee.org. [Accessed: June 8, 2012].

[36] Java, Learn About Java Technology, [Online]. Available: http://www.java.com/en/about/. [Accessed May. 10, 2012].

Oksana Nikiforova earned her Ph.D. in information technologies (system analysis, modeling and design) in Riga Technical University, Latvia, in 2001.
At present, she is Full Professor at the Department of Applied Computer Science, Riga Technical University, where she has been studying and working since 1999. Her current research interests include the object-oriented system analysis and modeling, with special focus on the issues in the framework of Model-driven Software Development (MDSD). She has published widely in these areas and has been awarded several grants.

She participated and managed several research projects related to system modeling, analysis and design, as well as participated in several industrial software development projects.
She is member of RTU Academic Assembly, Council of the Faculty of Computer Science and Information Technology, RTU Publishing Board, RTU Scientific Journal Editorial Board, etc. She co-chairs the workshops focused on MDSD – MDA 2009 in conjunction with ADBIS, MDA&MTDD 2010 and MDA&MDSD 2011 in conjunction with ENASE. She was awarded a RTU Young Scientist of the Year 2009.
E-mail: oksana.nikiforova@rtu.lv

Ludmila Kozacenko took bachelor's degree in computer systems from Riga Technical University, Latvia, in 2012.
She is presently the first year master's student and scientific assistant at the Department of Applied Computer Science, Riga Technical University.
Her current research interests include transformation approach classification and realization of transformation using general purpose programming language Java.
E-mail: ludmila.kozacenko@rtu.lv

Dace Ahilcenoka took the bachelor's degree in computer systems from Riga Technical University, Latvia, in 2012.
She is presently the first year master's student and scientific assistant at the Department of Applied Computer Science, Riga Technical University.
Her current research interests include UML diagram layouting, algorithms of diagram layout.
E-mail: dace.ahilcenoka@rtu.com