Two-Hemisphere Model Based Approach to Modelling of Object Interaction

Oksana Nikiforova, Ludmila Kozacenko, and Dace Ahilcenoka
Faculty of Computer Science and Information Technology
Riga Technical University
Riga, Latvia
oksana.nikiforova@rtu.lv,ludmila.kozacenko @rtu.lv, dace.ahilcenoka@rtu.lv

***Abstract* — It is a modern trend to use automatic transformations of different type of models to develop a software system. Software engineers have quite enough notations to present models at different levels of abstraction and at different stages of software development project. UML is an industrial standard for system modeling and specification and offers notational conventions for presentation of both aspects of the system – dynamic as well as static one. Currently, the research focuses in the area of software system modeling and model transformation is turned exactly to the dynamic aspect of the system. We propose to use the so called two-hemisphere model for receiving a set of elements, which are used for modeling an object interaction as a central part of the system dynamic presentation. The paper describes the main principles of the two-hemisphere model transformation into the UML sequence diagrams, as well as compares it to other transformation approaches.**

Keywords - UML sequence diagram; two-hemisphere model; layouting algorithm; model transformation; BrainTool.

I. INTRODUCTION

As we had stated in a previous paper devoted to the two-hemisphere model-driven approach [1], tools to support models and modeling at the initial stage of software development is the modern trend in business process modeling and analysis. Therefore, the focus of the automation of software development is shifted from automatic code generation from the system model to the automatic modeling of the problem domain and further code generation from them. Here, the valuable notation became the Unified Modeling Language (UML) [2] and its class diagram, which specifies the structure of the developed system and static information about system behavior. An ability to generate elements of the UML class diagram from the two-hemisphere model by BrainTool is demonstrated in [1]. Currently, we consider the dynamic aspect of the system and are investigating an ability to generate elements to present an object interaction according to the UML notation [2].

In general, there are two ways of looking at any software system. One way is to consider just data, including variables, arguments, data structures and files where the operations are examined only within the framework of the data. And the other way of viewing the software system is to consider just the operations performed on the data where data are of the secondary importance. According to the object-oriented software development, data and operations are viewed at equal importance, in spite of the fact that sometimes data have to be stressed and other times operations are more critical.

The main attention during object-oriented software development is devoted to the definition of system objects which are the primary artifacts of the developed system and include the information about data and operations together. Therefore, one of the fundamental tasks during object-oriented software development is to define an object structure and to share the responsibilities of an object, i.e., to determine the operations for objects to perform.

The paper presents the way to solve the problem of sharing responsibilities between objects by using the two-hemisphere model supported by BrainTool. We illustrate the process creating a two-hemisphere model [4] for a business domain and then investigate construction of the UML sequence and communication diagrams. In order to solve this task we defined a set of transformation rules and also focused on the problem of automatic layout of the UML diagrams after their derivation from the two-hemisphere model. Since it is very important to ensure that the diagrams are well built not only in terms of their content, but also how they visually represent the information.

The paper is structured as follows. Section 2 explains the essence of the object-oriented software development and discusses the importance of the object definition and responsibilities shared between them during the object oriented system analysis and modeling. Section 3 defines the essence of the two-hemisphere model transformation to share responsibilities to perform operations by system objects. Section 4 demonstrates BrainTool supporting the proposed approach and discusses the problem of the UML sequence diagram layout and its solution. Section 5 compares BrainTool with other tools giving an ability to create the UML diagrams. In conclusion, we stress the main contribution of the paper and state the directions for the further research.

II. THE ROLE OF THE OBJECT INTERACTION MODELING IN THE SOFTWARE DEVELOPMENT

The object-oriented software development assumes that the main attention is to be devoted to identification of objects from the problem domain and to sharing responsibilities of operation execution between these objects. Therefore, the role of the system modeling becomes very important. In the object-oriented software development, the standard notation for the system modeling is the Unified Modeling Language

(UML) [2]. The UML diagrams give a possibility to present different aspects of software system, but UML is just a notation and does not contain methodological instructions on how to model the system. The developer needs the information about the system to be developed in the form, which gives an ability to transform this information into UML diagrams.

Basically, the software system development starts with the business information gathering and presenting it in the form suitable for further software system modeling. In classical approach, this information is presented as the processes to be performed and the information flows required for the process execution. Then, this presentation of business information has to be transformed into the model, which in object-oriented manner for software development requires to present objects to interact in the form of UML sequence diagram [2]. It shows the objects, their lifelines and messages to be sent by objects-senders and performed by object-receivers and is used to present dynamic aspect of the system, which in object-oriented approach is expressed in terms of message sending among objects. The dynamic of interactions is defined by an ordering of the messages. It serves as a basis for definition of operations performed by objects to be grouped into classes, as well as to present and to verify a dynamic aspect of class state transition. The problem, which is recently widely researched in the area of the object interaction analysis, is formal transitions among the models presented at the level of problem domain and system presentation expressed in terms of the object interaction, if we are dealing with the object-oriented software development and using a set of model transformation rules. For now, this transition is defined and is partly supported by UML modeling tools and some guidelines exist on how these transformations should be performed.

Loniewski et al. [3] show the result of analysis of different approaches to transformation of the problem domain description into the UML class diagram during the last 10 years, published in four digital libraries (IEEEXplore, ACM, Science Direct, Springerlink). The survey states that there exist enough approaches with different types of solutions for the generation of a UML class diagram and half of them are automated and supported by tools. However, the authors of [3] stress that these tools are not widely used practically and are created to approve the automation level of the approach offered by their vendors. Other researchers who are investigating the functionality of the UML modelling tools and model transformation tools raise the question about the ability to define the tool chain to cover all the necessary activities for software system development. For example, the lack of a conceptual view on the integration problem and appropriate reuse mechanisms for already existing integration knowledge, which forces the developer to define model transformation code again and again for certain recurring integration problems in an implementation-oriented manner, resulting in low productivity and maintainability of integration solutions. We consider that the maturity level of advanced modelling and model transformation tools is not enough to support the full chain of software system development. Thereby, despite the number of approaches to automatic creation of the system model and further code generation from it, the variety of tools supporting the system modelling at the initial stage of software development are reduced to UML editors and "tight" code generators.

The core of this paper is a hypothesis that our proposed notation of the two-hemisphere model supported by BrainTool contains enough information for sharing responsibilities among objects and can serve for automatic generation of the elements to present the UML sequence diagram. Whereas UML sequence is stated as an one of ambiguous UML diagrams [5], with the implicit and informal semantic that designers can give to basic sequence diagram as a result of this conflict [6], [7], [8]. The two-hemisphere model [4] contains information about business processes and concepts and has already been used for representation of object interaction with UML communication diagram [9], where only static view of the system is investigated and an ordering of message sending and receiving is missed. Currently, we define the mapping between elements of two-hemisphere model and elements of UML sequence diagram, especially in its timing aspect, solve the problem of sequence diagram layout and offer to use BrainTool for receiving of the UML sequence diagram.

III. Definition of transformation from the two-hemisphere model into the UML sequence diagram

A nature of transition from business information into the object interaction is found in the definition of which processes have to be performed in the system and which performer will execute exact process at the software level of system modeling. In order to identify a performer of the process at the software level of system presentation the process has to be analyzed with the aim to define a software operation to execute the process and to notice the object to perform this operation. So far two general steps can be defined for the object-oriented system analysis. The first one is to identify objects themselves. This task is solved by [10], [11]. In general, the analysis of entity relationship [12] can serve as a base for the object identification of the software system. Further, the second activity of object-oriented system analysis is so called "sharing of responsibilities" among the objects, which is not so trivial and is stated for solving by the author of the paper. The main task to be defined is which operation will be executed by which object and in which time sequence.

In UML models, objects interact to implement behavior. UML has two kinds of diagrams to reflect object interaction – communication and sequence diagrams. Communication diagram allows observing the common interaction of objects in the system mainly focused on associations between objects and time aspect is not stressed in the communication diagram. The UML sequence diagram shows interaction of objects for execution of concrete use case or business function expressing time aspect as a main focus of the modeling. We analyze the possibility to generate all the necessary information for object interaction (especially time component of that) in terms of the UML sequence diagram.

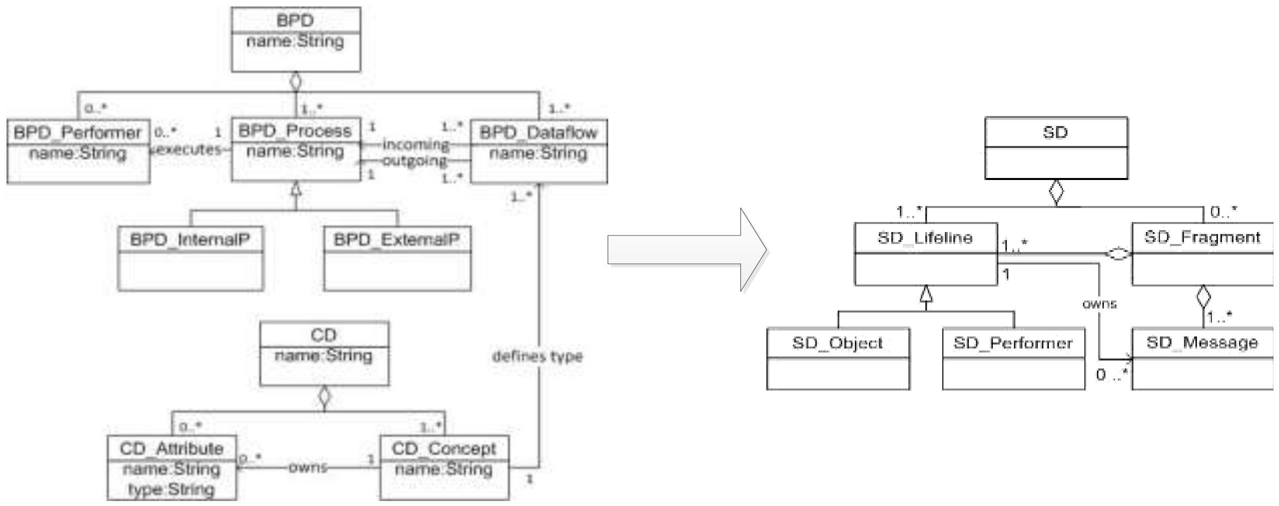

Figure 1. Elements of the two-hemisphere model used in transformation rules to generate elements of the UML sequence diagram.

The definition of elements of the sequence diagram needs an examination of elements of two-hemisphere model, which is presented as a business process model related with concept model. The sequence diagram consists of objects, their lifelines and messages which they have to send to other objects.

A simplified sequence diagram metamodel [13] presented at the right side of Fig. 1 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. The left side of Fig. 1 shows the metamodel of the two-hemisphere model [13].

Object identification is based on the analysis of noun phrases in the problem domain description [10], where it is presented in the form of two-hemisphere model and contains the information about the problem domain, where the noun phrases are defined for the events (arcs) of business process model and concepts of the concept model (see Fig. 2). Therefore, it is possible to suggest that description of an event in business process with its defined data structure in concept model can serve as a basis for identification of an object in the sequence diagram.

The transformation of the two-hemisphere model into the UML communication diagram is performed in a direct way of graph transformation, where arcs (i.e., information flows in Fig. 2) of graph of business processes are transformed into the nodes of graph of object communication. E.g., “Applicant data” as an information flow in process model becomes a class “Applicant data” in communication and sequence diagrams. Process “add applicant to group” in process model becomes a method “add_applicant_to_grop()” sent by object “Applicant data” to object “Group blank” presented on the interaction diagrams. As for UML sequence diagram, the description of an event in business process with its defined data structure in concept model can serve as basis for definition of the object, which is a node of its lifeline.

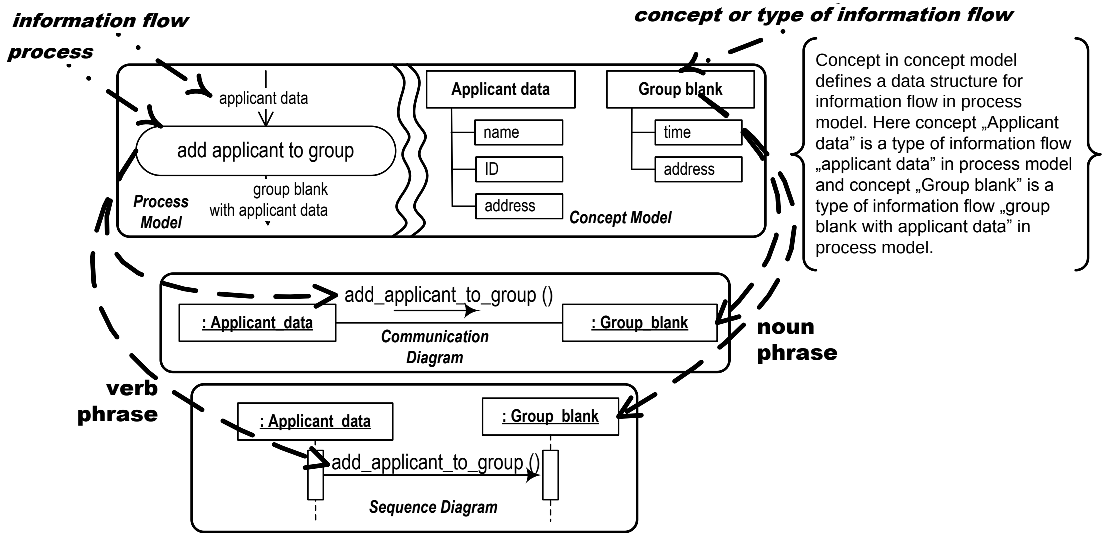
information flow
process
concept or type of information flow
applicant data
add applicant to group
group blank with applicant data
Process Model
Applicant data
name
ID
address
Group blank
time
address
Concept Model
Concept in concept model defines a data structure for information flow in process model. Here concept „Applicant data” is a type of information flow „applicant data” in process model and concept „Group blank” is a type of information flow „group blank with applicant data” in process model.
add_applicant_to_group ()
: Applicant_data
: Group_blank
Communication Diagram
noun phrase
verb phrase
: Applicant_data
: Group_blank
add_applicant_to_group ()
Sequence Diagram

Figure 2. Analysis of verb and noun phrases in two-hemisphere model and related object interaction.

The analysis of a verb phrase (see Fig. 2 [14]) makes it possible to suggest that the name of a business process has to be the base for the definition of a message of a sequence diagram to be performed by the object-receiver of this message. And if the name of the business process is defined in the form, where the first word is a verb, we can assume that the name of the exact message will be the same as the name of business process. According to the notation of the sequence diagram in [2], a message has the object-sender and the object-receiver of the message, which has to perform the action defined in the message. Direct transformation of graph of business processes into the graph of object communication defined in [9] solves the problem of identification of the object-sender and the object-receiver of the exact message by the application of several outlines of graph theory [15], where nodes of the graph of business processes have to be transformed into the arcs of the object communication and the arcs of the graph of the business processes have to be transformed into the nodes of object communication. The same assumption can be applied for the definition of objects in the sequence diagram – the object sender will be defined by an incoming arc of exact process in the model of business processes and the object-receiver will be defined by an outgoing arc of exact process in the model of business processes (see an example in Fig. 2 [14]). Therefore, the message defined to execute an exact process in the business process diagram will be sent by the object defined in the incoming arc of exact business process and received by the object in the outgoing arc of exact business process.

IV. The Two-hemisphere Model Driven approach Support by BrainTool

BrainTool [16], developed by the researchers of the Riga Technical University, is a step forward in the area of automation of the modeling process. There exist a number of tools, which generate different UML diagrams. Some of them enable to define several elements of class structure based on a data presentation of the problem domain, e.g. Sparx Enterprise Architect or Rational Software Architect. Others generate the system model from the existing source code, to display the structure or the dynamic of the developed system, e.g. MS Visual Studio 2010. However, the problem of automatic generation of the UML diagrams from the formal and still customer-friendly presentation of the problem domain is not solved yet. Nikiforova et al. proposed to use BrainTool to generate UML class diagram from the two-hemisphere model [1]. Currently, the research group is working on a set of transformation rules for the generation of the UML interaction diagrams to built-in them into BrainTool and to expand a spectrum of the diagram supported by the tool. The essence of the transformations is described in the previous section. But the transformation provides only mapping of elements from a source to a target model. Layout of the model elements is another potential research problem to be solved to complete the task of supporting the automatic generation of the diagrams by BrainTool.

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, among them aesthetic and layout criteria. General diagram criteria and specific UML diagram layout criteria have been studied by [17], [18], [19], [20] and others. All diagrams should comply with general graph layout criteria as a result from the theory of perception [17].

The UML communication diagram in the task of its layout can be accessed as usual graph, containing nodes connected by edges. Therefore, it is possible to use layout principles for usual graph layout. The UML sequence diagram, otherwise, is very specific in its visual presentation. All the objects are allocated horizontally at the top of the diagram and the lifelines 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. Poranen et al. [20] and Wong et al. [17] have identified the criteria specific for sequence diagrams, which are taking into consideration implementing the layout algorithm for the UML sequence diagram in BrainTool.

The 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). The pseudo code of the layout algorithm implemented is presented in Fig. 3. The possible result of the transformation of the two-hemisphere model into the elements of the UML sequence diagram is shown in Fig. 4.

```
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
```

Figure 3. Pseudo code of the layout algorithm.

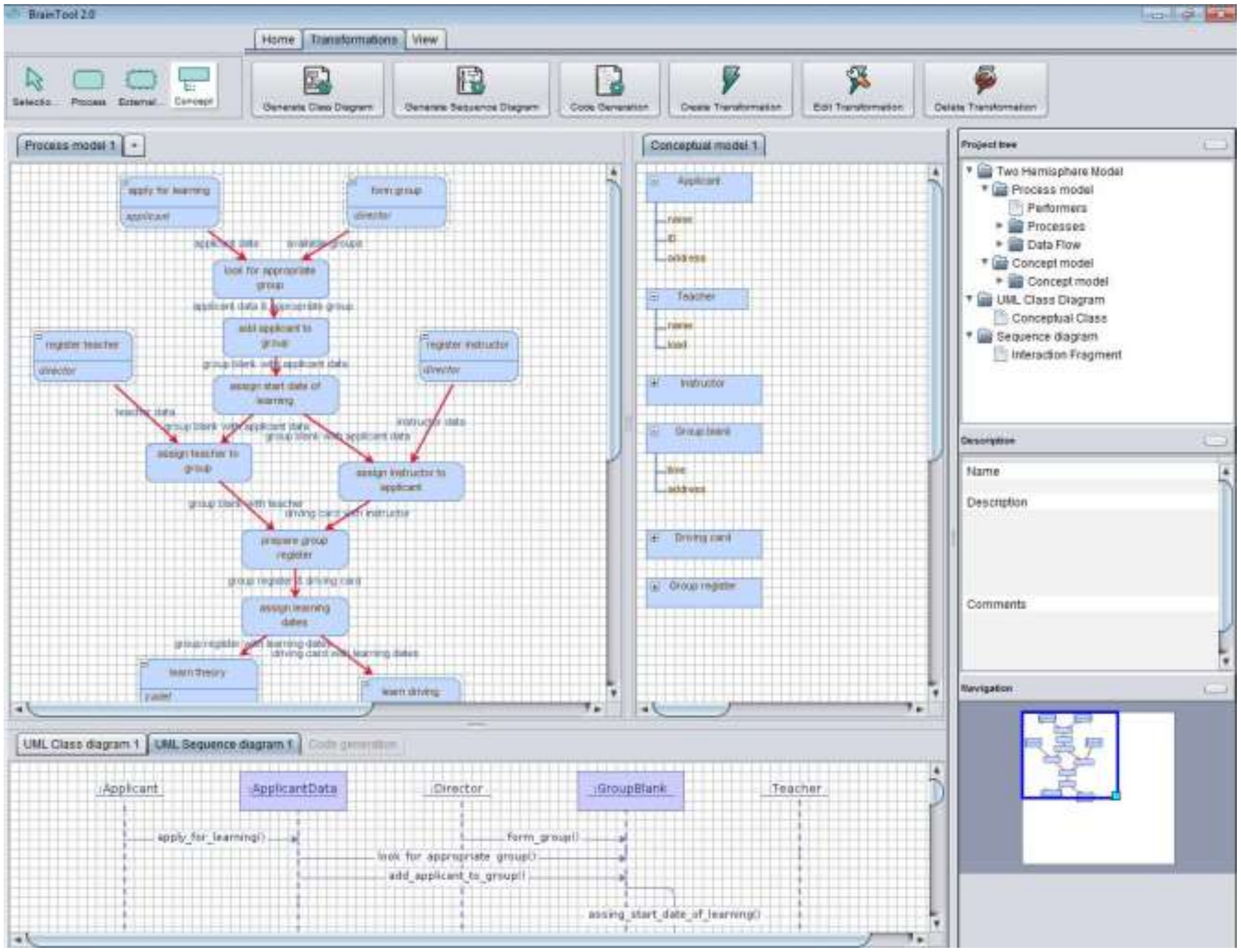

Figure 4. General view of BrainTool.

V. Comparison of the BrainTool with other UML compatible Tools

We have listed several tools offering creation of the UML interaction diagrams in Table 1, but they are mainly UML editors, where a developer creates all the diagrams manually with limited ability to generate new elements. Tools, like Sparx Enterprise Architect [21], Visual Paradigm [22] or Rational Software Architect [23] gives the ability to reflect to the existing UML diagram elements, if they are already created in other UML diagrams, but still, initially, these elements are identified manually.

Attempts to receive UML interaction diagrams from the requirements in natural language are one of the popular lines of research. For example, ReDSeeDS [24] supporting tool proposes linguistic analysis of system requirements and generates several elements of the UML sequence diagram, based on predefined format of requirements specification. But the tool has no graphical presentation of the resulting diagram and exports the result to Sparx Enterprise Architect.

On the other hand, Visual Studio supports the ability to generate the UML sequence diagram from the source program code. This is different direction from the approach offered in this paper and the tool can be interesting for comparison only in diagram presentation aspect, like as the diagram layout implementation, or export to other UML compatible tools.

There are several tools that provide automatic diagram layout, e.g., Borland Together [25] (not listed in Table 1) supports automatic UML sequence diagram layout, but uses lawless set of layout criteria). Sparx Enterprise Architect [21] is the tool that also provides automatic UML sequence diagram layout, however, it does not satisfy all the mentioned criteria of layout.

Thereby, we appreciate that currently abilities for the generation of the UML interaction diagram offered by the two-hemisphere model driven approach and supported by BrainTool are the most expansive, but we still have to refine the tool with additional functionality expected by users in popular UML editors.

TABLE I. COMPARISON OF BRAINTOOL TO OTHER TOOLS PROVIDING THE POSSIBILITY TO GENERATE THE UML INTERACTION DIAGRAM

Tool / Criteria	**Visual Paradigm**	**Sparx EA**	**IBM RSA**	**Visual Studio**	**ReDSeeDS**	**BrainTool**
Initial information for generation of the UML interaction diagrams	System req-ts & use-case diagram	System req-ts & use-case diagram	System req-ts & use-case diagram	Program code	System req-ts	Two-hemisphere model
Actors	Borrowed from use-cases	Borrowed from use-cases	Borrowed from use-cases	No	Automatically	Automatically
Objects	Manually	Manually	Manually	Automatically	Automatically	Automatically
Lifelines	Manually	Manually	Manually	Automatically	Automatically	Automatically
Operations	Manually	Manually	Manually	Automatically	Automatically	Automatically
Operation ordering	Manually	Manually	Manually	Automatically	Automatically	Automatically
Interaction frames	Manually	Manually	Manually	Automatically	Automatically	Automatically
Operation parameters	Manually	Manually	Manually	Automatically	Automatically	No
Links between objects (in communication diagram)	Manually	Manually	Manually	No	Automatically	Automatically
Transformation base	Linguistic analysis	Linguistic analysis	Linguistic analysis	Formal transformation text-to-model	Linguistic analysis	Formal transformation model-to-model
Model editor for initial information	Text editor	Text editor	Text editor	Text editor	Text editor	Graphical editor
Graphical representation of the UML sequence diagram	Yes	Yes	Yes	Yes	No	Yes
Graphical representation of the UML communication diagram	Yes	Yes	Yes	No	No	Not yet
Automatic layout	Not for UML sequence diagram	Lawless ordering of objects in the top of diagram	Not for UML sequence diagram	Yes	No	Yes
Export abilities to UML compatible tools	Has special export format	Has special export format	Has special export format	No	Yes (at least to Sparx EA	Defined by XMI and importable in the tools supporting the standard specification

VI. CONCLUSION

In comparison with the traditional software engineering development methods the model-driven approaches provide 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. A 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 are several problems, which do not allow completing the model transformation.

The research object of this paper was the generation of the UML interaction diagrams, based on the two-hemisphere model. Both activities for that are being investigated: they are element identification from the problem domain and the visual representation (i.e., layout).

Thus, the contribution of the paper can be summarized as follows:

- A set of transformation rules for derivation of elements to present object interaction in terms of the UML diagrams are defined and implemented in BrainTool;
- 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 lifelines, messages and interaction frames.
- The tool supporting the transformations presented in this paper is compared to other tools giving an ability to create UML diagrams.

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 for object interaction presentation.
- 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 a large amount and diversity of the criteria that should be taken into consideration when placing elements in the diagram.

• A 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 are implemented in BrainTool [16] in order to expand the functionality of its first version presented in [1] with respect to 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. This can be stated as a direction for a 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.

ACKNOWLEDGMENT

The research presented in the paper is supported by Accenture Latvian Branch, project No. L7950 “Development of Model Transformation Tool Prototype”, and by Latvian Council of Science, No. 342/2012 "Development of Models and Methods Based on Distributed Artificial Intelligence, Knowledge Management and Advanced Web Technologies".

REFERENCES

[1] O. Nikiforova, K. Gusarovs, O. Gorbiks, and N. Pavlova “BrainTool. A Tool for Generation of the UML Class Diagrams”, Proc.: The Seventh International Conference on Software Engineering Advances, 2012, pp. 60-69.

[2] Unified modeling language: superstructure v.2.2, OMG. Available: http://www.omg.org/spec/UML/2.2/Superstructure. [retrieved 09/ 2013]

[3] G. Loniewski, E. Insfran, and Abrahao S. “A Systematic Review of the Use of Requirements Techniques in Model-Driven Development”. In: 13th Conference, MODELS 2010, Model Driven Engineering Languages and Systems, Part II, Oslo, Norway, pp. 213—227.

[4] O. Nikiforova and M. Kirikova, “Two-hemisphere model drivenapproach: engineering based software development,” in The 16th International Conference Advanced Information Systems Engineering, A. Persson and J. Stirna, Eds. BerlinHeidelberg: Springer-Verlag, LNCS 3084, 09.2004, pp. 219-233.

[5] C. Sibertin-Blanc, O. Tahir, and J. Cardoso, “Interpretation of UMLsequence diagrams as causality flows,” in Advanced DistributedSystems, 5th International School and Symposium. Heidelberg:Springer, LNCS, vol 3563, 2005, pp. 126-140.

[6] R. Alur, K. Etessami, and M. Yannakakis, “Inference of messagesequence charts,” in Proceedings of the 22nd International Conferenceon Software Engineering (ICSE). New York: ACM Press, 2000, pp. 304-313.

[7] S. Uchitel, J. Kramer, and J. Magee, “Detecting implied scenarios inmessage sequence chart specifications,” in Proceedings of the 9thEuropean Software Engineering Conference and 9th ACM SIGSOFTInternational Symposium on the Foundations of Software Engineering(ESEC/FSE’01). New York: ACM, 2001, pp. 74-82.

[8] B. D. Aredo, “A framework for semantics of UML sequence diagrams inPVS,” JUCS, vol. 8, no. 7, 2002, pp. 674-697.

[9] O. Nikiforova, Object Interaction as a Central Component of Object-Oriented System Analysis, (ENASE 2010), Proc. International Conference Evaluation of Novel Approaches to Software Engineering, WS Model Driven Architecture and Modeling Theory Driven Development, Osis J., Nikiforova O. (Eds.), SciTePress, Portugal, 2010, pp. 3-12.

[10] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object Oriented Modeling and Design. New Jersey, Englewood Cliffs:Prentice-Hall, Inc, 1991.

[11] C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, 3rd ed. New Jersey: Prentice Hall, 2005.

[12] P. Chen, “The entity relationship model – towards a unified view of data,” ACM Trans. Database Systems, vol. 1, 1976, pp. 9-36.

[13] O. Nikiforova, L. Kozacenko, and D. Ahilcenoka “UML Sequence Diagram: Transformation from the Two-Hemisphere Model and Layout”, *Applied Computer Systems*. Vol.14, 2013, pp. 31-41, doi: 10.2478/acss-2013-0004

[14] O.Nikiforova, "System Modeling in UML with Two-Hemisphere Model Driven Approach," Scientific Journal of Riga Technical University. Computer Sciences, vol. 21., 2010, pp. 37-44

[15] J. Grundspenkis, “Causal domain model driven knowledge acquisitionfor expert diagnosis system development,” in Lecture Notes of theNordic-Baltic Summer School on Applications of AI to ProductionEngineering, Kaunas, Lithuania, K. Wang and H.Pranevicius, Eds. Kaunas: Kaunas University of Technology Press, 1997.

[16] BrainTool. Availablet http://braintool.rtu.lv/ [retrieved 09/ 2013]

[17] K. Wong and D. Sun, On evaluating the layout of UML diagrams for program comprehension: IWPC 2005, 13th International Workshop on Program Comprehension, St. Louis, Missouri, USA. IEEE Computer Society, 05.2005.

[18] 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.

[19] 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 conjinction with the 6th International Conference on Evaluation of Novel Approaches to Software Engineering, Beijing, China. SciTePress Digital Library, 06.2011.

[20] 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, Kuopio, Finland. University of Kuopio, Department of Computer Science, 06.2003.

[21] Sparx systems, "Enterprise Architect". Available: http://www.sparxsystems.com.au/ [retrieved 09/ 2013]

[22] Visual Paradigm, "Generate Sequence Diagram from Use Case Flow of Events", May 2011. Available: http://www.visual-paradigm.com/product/vpuml/tutorials/gensdfromfoe.jsp [retrieved 09/ 2013]

[23] IBM, Rational Software Architect. Available: http://www.ibm.com/developerworks/rational/products/rsa/ [retrieved 09/ 2013]

[24] ReDSeeDS. Available: http://www.redseeds.eu/ [retrieved 09/ 2013]

[25] Borland a micro focus company, Borland Together, Available: http://www.borland.com/products/Together/ [retrieved 09/ 2013]