BrainTool

A Tool for Generation of the UML Class Diagrams

Oksana Nikiforova, Konstantins Gusarovs, Olegs Gorbiks, Natalja Pavlova
Faculty of Computer Science and Information Technology
Riga Technical University
Riga, Latvia
e-mail: oksana.nikiforova@rtu.lv, konstantins.gusarovs@rtu.lv, olegs.gorbiks@rtu.lv, natalja.pavlova@rtu.lv

***Abstract*—Object-oriented system modeling enables the sharing of responsibilities between system objects at a high level of system abstraction. The UML class diagram is the central part of the object-oriented system model and serves as a "bridge" between the information about the problem domain at the customer's side and the software components at the developer's side. However, UML is not a methodology for how to model the system, but just a notation for "drawing" of model elements. This paper demonstrates the functionality of the BrainTool, which enables the generation of the UML class diagram from the so called two-hemisphere model, where the problem domain is presented as a concatenation of the problem domain processes, incoming and outgoing information flows and their types. BrainTool is developed using Visual Studio .NET for modeling of the two-hemisphere model, the Python programming language for definition of transformation rules and XMI for model interchange with Sparx Enterprise Architect.**

Keywords-BrainTool v1.0; UML class diagram; two-hemisphere model; model transformation.

I. INTRODUCTION

The object-oriented approach is widely used in software development. One of the tasks of software development is to present different aspects of the system for the implementation of the software solution for the required system. In solving this task, system modeling became an important activity in software development. The goal of system modeling is to represent the system graphically, in a form understandable to analysts, developers and at least partly understandable to the customer. A systematic approach to the derivation of the system model from information about the problem domain plays an important role in completing the task of system modeling.

K. Vollmer, C. Richardson, and Clair C. research [1] confirms that tools to support models and modeling at the initial stage of software development are 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 UML diagrams to the automatic modeling of the UML diagrams and further code generation from them. Here, the valuable diagram became the UML class diagram, which specifies the structure of the developed system and static information about system behavior.

Moreover, the increasing interest towards software development within the framework of Model Driven Development (MDD) [2] turns the focus again to the area of model transformation at different levels of abstraction. Unified Modeling Language (UML) [3] is an industry standard for software specification and modeling in an object-oriented manner. The UML class diagram is used to model class specification and serve as a "bridge" between the information about the problem domain and the information required for definition of the software components and their architecture. Researchers are trying to achieve a high enough level of automation in creation of the UML class diagram and derivation of the diagram from information about the problem domain. Currently, an increasing number of developers admits the necessity to model system at the initial stage of the software development project, and the models are increasingly used to specify the system and its processes at the business level [1], [4].

BrainTool [5], developed by 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 the UML class diagram. Some of them enable to define several elements of class structure based on data presentation of the problem domain. Others generate a class diagram from existing source code, to display the structure of the developed system. However, the problem of automatic generation of the UML class diagram from the formal and still customer-friendly presentation of problem domain is not solved yet. Authors of BrainTool propose to generate UML class diagram from the so-called two-hemisphere model [6] of the problem domain, which presents information about processes, information flows between these processes and pre-defined types of these information flows.

In 2004, when the main idea of the two-hemisphere model were published, the lack of the appropriate languages and standards eliminated the ability to support the two-hemisphere modeling of the system and to implement the transformations defined for it by tool. The evolution of the idea of model driven software development and appearance of different techniques for development of such modeling environment with embedded abilities for implementation of

 ISBN: 978-1-61208-230-1

transformations gave us a powerful means to create such the tool.

The goal of this paper is to solve the task of tool development to support generation of the UML diagram from the initial model of the problem domain expressed in terms of the two-hemisphere model and to discuss about technical abilities of current solutions to create such a tool.

The paper is structured as follows. The next section describes the related work in the area of UML class diagram generation and tools supporting this generation. Section 3 explains the main principles of the transformations used for generation of the UML class diagram from the two-hemisphere model. The essence of the two-hemisphere model is also described in the third section. Section 4 gives several explanations on the solutions used for implementation of BrainTool and shows its main components. Several conclusions on the application of BrainTool and directions for future research are stated in the fifth section.

II. RELATED WORK

Since the beginning of the 1980s, a great number of modeling tools and model generating software systems have been offered to attack problems regarding software productivity and quality [7]. Modeling tools developed since that time were oversold on their "complete code-generation capabilities" [8]. Nowadays, similar situation is observed in modeling tools, using and integrating UML models at different levels of abstraction and automation of software development [9].

Most of today's tools combine a number of modeling and code generation functions in a more or less open fashion. The traditional modeling tools provide a model editor and a model repository. A code generator based on a scripting language and plugged into a modeling tool provides the transformation tool and transformation definition editor. In that case, the transformation repository is simply text files [10].

The variety of the "model-driven" tools can be divided into tools created for defining the system model itself and tools to support code generation from the UML model. The first group of tools is so-called "UML editors", where the developers of these tools provide different levels of automation of the actual model creation. BrainTool demonstrated in this paper can be classified as a tool for automatic creation of UML class diagrams, where the result of the generation – the UML class diagram – is importable either into UML editors for further refinement and working with model or into code generation tools for further generation of software components.

Loniewski et al. in [11] describe the results of a survey about different approaches used for transformation of system requirements to system design and implementation. The survey shows 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 many approaches with different types of solutions for the generation of a UML class diagram. Moreover, the authors analyze the approach based on several criteria, one of them is tool support. Analysis of the automation level in these approaches shows that 25 out of 71 approaches described in corresponding papers are supported with a tool. However, Loniewski et al. stress that these tools are academic tools and are not widely practically used as far as they are created to approve the automation level of the approach offered by their vendors [11].

One more kind of the related tools are tools that generate the class diagram from a data structure or a data model. These tools require a solid contribution from a software specialist to define all these structures. It is already the modeling of the UML class diagram itself. In contrast to these tools, BrainTool generates the class diagram from initial information about the system, which is understandable for the business analyst and doesn't require software knowledge for its modeling. Therefore, a tool that generates the class diagram in the initial stages of the project is very useful. It allows to automatically create a static structure of the developed system and serves as a base for further code, avoiding mistakes and mismatches between requirements and implementation.

As far as for the evolution of the two-hemisphere model, which is a base model for generation of the UML class diagram in BrainTool, the main idea of displaying the initial information about the system with two interrelated models– the business process model and the concept model – and the hypothesis about how to use two interrelated model to share the responsibilities between object classes was demonstrated on the abstract example in [6] and later in several real projects. In all cases, the two-hemisphere model was created manually, in the first one by authors and in others by an independent problem domain expert. Successful application of two-hemisphere model transformation into the UML class diagram served as a motivation to support these transformations by software system. The first software prototype of tool supporting two-hemisphere model based transformation was introduced in 2008 [12]. The prototype used textual information in special format as a source and produced a text file containing description of the resulting UML class diagram as a specification, where classes, attributes, methods and relationships were listed in pre-defined format. Analysis of these generated text files gave authors an ability to refine transformations for definition of relationships between classes; the results are published in [13]. Currently, the ability to apply the two-hemisphere model for generation of the UML sequence diagram with attention to the timing aspect is investigated and preliminary results are published in [14]. So far, the continuing research in the area of model-driven software development and an increasing demand in the industry for automation of the ability to bridge the gap between problem domain and software components, can serve as a motivation to develop the first version of BrainTool, which gives an ability to draw the two-hemisphere model in the manner suitable for the problem domain expert and to generate the UML class diagram from it.

 ISBN: 978-1-61208-230-1

Moreover, instead of manual creation of the UML class diagram directly from information about problem domain based on principles of object-oriented analysis, the proposed BrainTool gives an ability to use already existing business artifact – a business process diagram is widely used in many enterprises, and the structure of information flows between processes is definable under description of user stories. A lot of organizations are using different tools for business process analysis and therefore they have complete and consistent models of their organizational structure, employer responsibilities, business processes and the structure of documentation flows, in other words, well-structured initial business knowledge, which can serve as a basis for even automatic creation of the two-hemisphere model.

The main benefit of the two-hemisphere model is that it can be created and often already is created by the business analyst at the customer's side. A Standish group survey shows that about 83% of companies are engaged in business process improvement and redesign. This implies that many companies are very familiar with business process modeling techniques or at least they employ particular business process description frameworks [4], [15]. On the other hand, the practice of software development shows that functional requirements can be derived from the problem domain description as much as 7 times faster than if trying to elicit them directly from users [16]. Both facts mentioned above and the existence of many commercial and open source business modeling tools are a strong motivation to base software development on the business process model, rather than on any other soft or hard models.

Therefore, with minimal efforts, the two-hemisphere model, which is created and intuitively understandable by the customer, can be used to automatically generate class diagram prototypes that can be later reviewed and used in software development.

III. TRANSFORMATION OF TWO-HEMISPHERE MODEL TO THE UML CLASS DIAGRAM

The two-hemisphere model driven approach uses the transformation of graphs, where nodes of one graph become the edges of the other graph, and edges of the first graph become the nodes of the other. These two initial interrelated graphs are: business process model (shortly – process model), which displays behavior of the system and the model of conceptual classes (shortly – concept model), which displays a skeleton of system's static structure. The meaning of objects in an object-oriented philosophy gives a possibility to share responsibilities between objects based on the direct graph transformation, where the data flow outgoing from the internal process in the process model becomes the owner of this process for performing it as an operation in object communication.

The essence of the transformation is illustrated on the left side of Figure 1. The business process model (graph G1 in Figure 1) is interrelated with the concept model (graph G2 in Figure 1) as follows. A certain concept in the concept model defines the data type for one or several data flows between business processes. The business process model is transformed into an object communication diagram (graph G3 in Figure 1), where edges (i.e., data flows) of the business process model became nodes (i.e., objects) of communication diagram, and nodes of business process model (i.e., processes) became edges (i.e., messages to perform the operation) of the communication diagram. The communication diagram itself serves as a base for the definition of classes-owners of methods in the UML class diagram. Details about the application of the two-hemisphere model are expressed in [12].

The right side of Figure 1 shows the interpretation of the transformations defined by the approach after the transformations have been studied for the implementation.

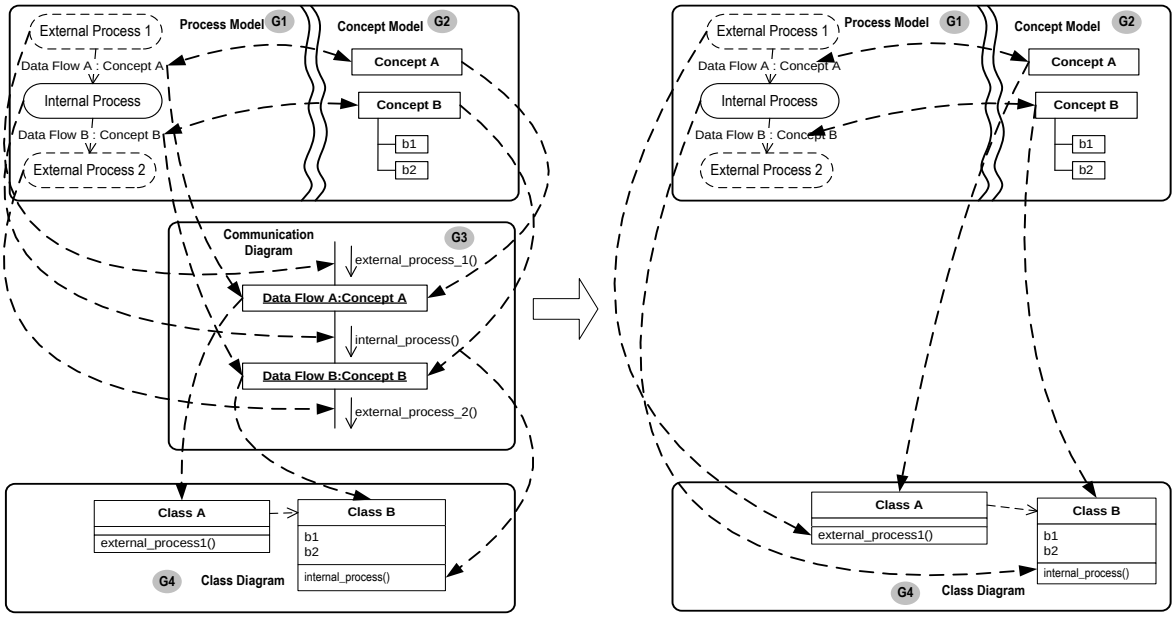

Figure 1. Interpretation of the transformations from two-hemisphere model to the UML class diagram.

Therefore, the left side of Figure 1 shows the transformations defined by the approach of the theoretical investigation of the sharing responsibilities among objects and the right side of Figure 1 shows the situation as it is simplified for tool development. The elements of the business process model are transformed into the UML class diagram directly. The edges of the business process model became the nodes of the UML class diagram. The nodes of the business process model became the methods of the classes, which were the outgoing data flows of the exact business process.

The analysis of different situations, which may appear in drawing the process model, i.e., a number of incoming and outgoing data flows, a variety of their types and so on, has given a possibility to define various transformation cases depending on the number of input and output processes and cardinality (a set of different concepts assigned to a certain data flow). These transformation cases are implemented according to the definition of relationships between generated classes, which are expressed in [13].

IV. IMPLEMENTATION OF TWO-HEMISPHERE MODEL IN BRAINTOOL

The goal of the prototype tool implementation in 2008 [12] was to examine the efficiency of the proposed method and to confirm that transformations offered by the two-hemisphere model driven approach can be automated. The current version of the implementation of the two-hemisphere model driven approach can be stated as a standalone tool entitled BrainTool in correspondence with the title of the approach, which in turn is derived from cognitive psychology [17] by analogy with human brain consisting of two interrelated hemispheres.

According to [10], a modern trend in system modeling tools is having the components to implement a model editor, a repository, its validation and transformation to another model. BrainTool gives a possibility to create the two-hemisphere model, to save it in the defined repository, to apply all the defined transformations for generation of the UML diagram and to export it in XMI format.

The Model Editor is a part of the tool providing model creation and modification possibilities. Model Repository is the "database" for models, where they are stored. The Transformation Definition Editor is used for transformation definition construction and modification. Currently, the Python interpreter is being used to support this component. However, it is possible to define the transformation in any programming language.

Finally, The Model Validator is a component used to check if the model is well-enough defined and has no potential problems that can affect the transformation result. This component is implemented as a standalone transformation using the Python programming language. The next subsections describe the main components of BrainTool and technical solutions for their implementation in detail.

A. *Model Editor*

The two-hemisphere model can be designed and then transformed using BrainTool model editor shown in Figure 2.

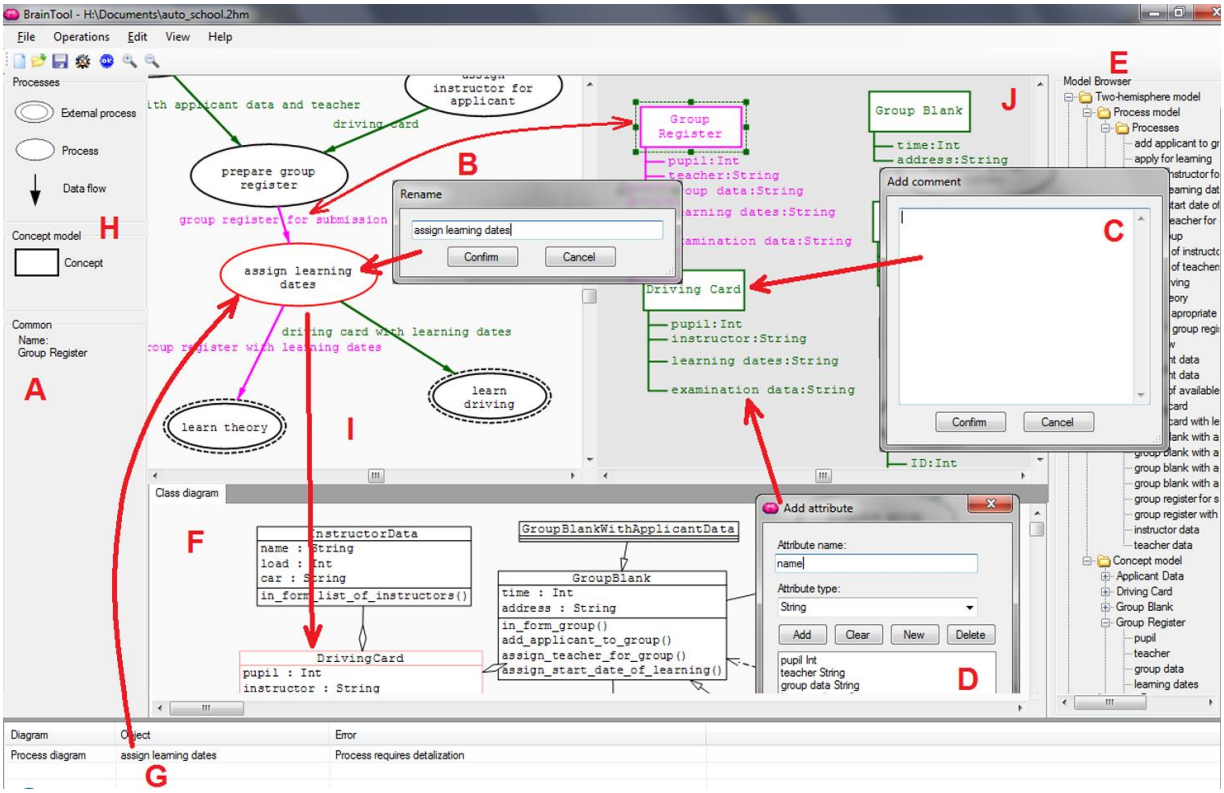

Figure 2. Model editor view in BrainTool.

The screen of BrainTool is divided into three parts. The information panel (highlighted as H in Figure 2 on the left side of the screenshot) shows the list of elements defined by the two-hemisphere model, where panel (A) provides basic information on the currently selected element.

The central part is divided into three drawing frames – the process model (I), the concept model (J) and the resulting class diagram (F). Any element of the two-hemisphere model can be entitled (highlighted as B for processes and D for attributes in Figure 2) or commented (C). The tree view of all objects defined in all models (including the resulting model) is shown in the right part of the screen shot under the letter E in Figure 2. The diagram elements causing transformation problems are listed at the bottom of the screen (G). Such problematic elements are also highlighted in the model perspective.

The simplified version of the business process for the driving school is reflected in Figure 3, where the process model is presented on the top side and the model of conceptual classes (so called concept model) is presented on the bottom side of the figure, which presently is the screen shot from BrainTool.

B. *Transformation Definition*

According to the transformation definition – a transformation is the automatic generation of a target model from a source model [10]. In the case of BrainTool, the source model is presented as a two-hemisphere model consisting of the process diagram, a set of concepts and concepts assigned to data flows. The target is the UML class diagram, which is a set of classes, class methods, class attributes, interfaces and relationships between classes and interfaces. The first transformation task is to generate classes of the resulting UML class model. Classes are created from concepts and retain their attributes. Cardinalities (number of different concepts linked to separate data flows) of process' inputs and outputs are used to determine different types of the relationships between classes in the UML class diagram.

For example, outgoing multiple data flows assigned to the same concept give an ability to define the generalization. The transformation rules give a possibility also to define aggregation, dependency or at least simple association. The following high-level pseudocode expresses the idea of the transformation for class creation:

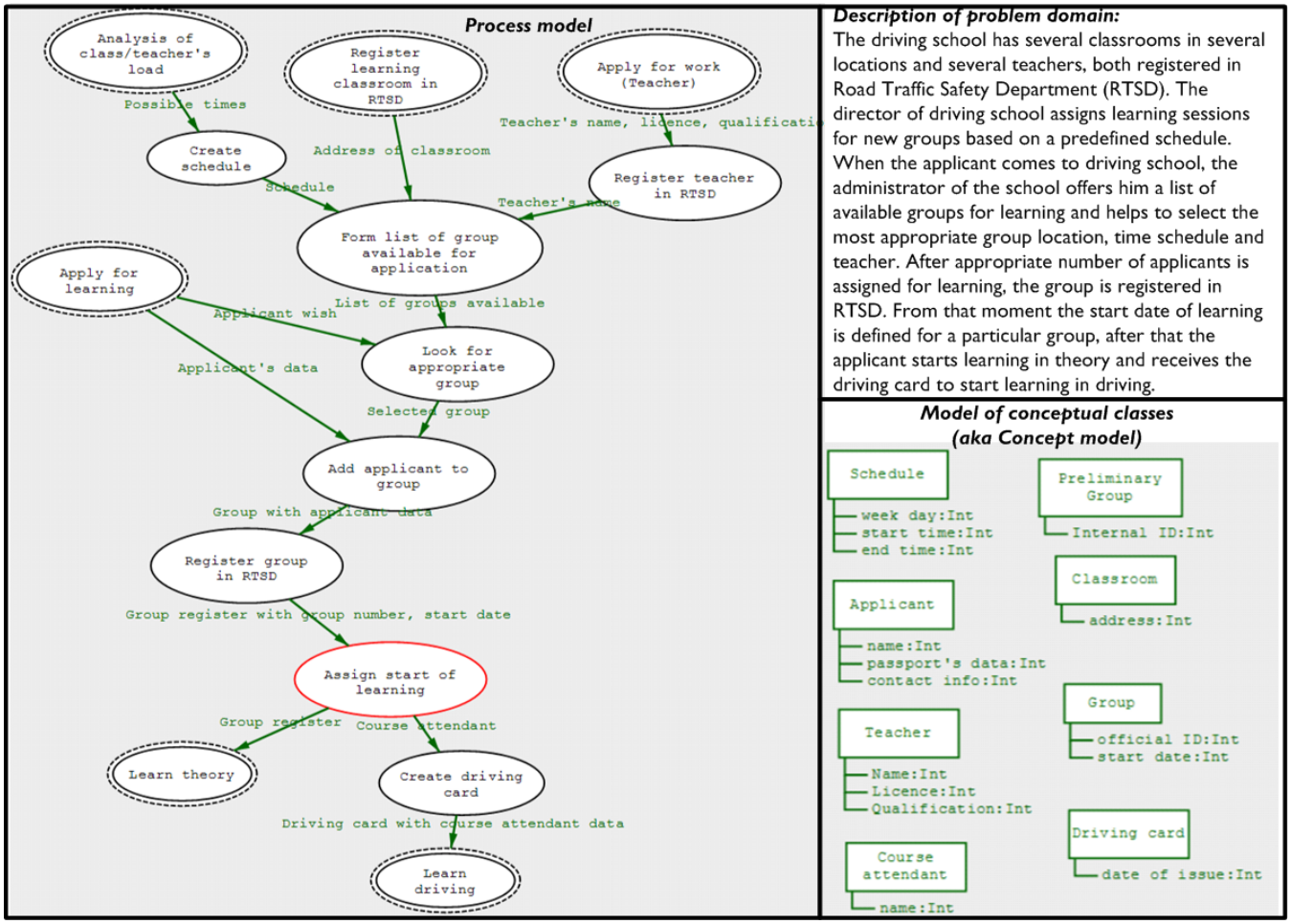

Figure 3. Two-hemisphere model of a driving school.

```
func generate_classes(process_model pm,
concept_model cm, class_model clm)
  for each concept in cm do
    clm.create_class_from(concept)
  for each process in pm do
    u_inputs = node.input_set().cardinality
    outputs = len(node.output_set())
    u_outputs = node.output_set().cardinality

    if u_inputs = 1 and u_outputs = 1 and outputs != 1 then
      for each output in node.output_set() do
        clm.create_class_from(output)
        clm.define_generalization(output, node.input_set())
```

Processes from the process model become the class methods as a result of the transformation expressed in such pseudocode fragment:

```
func assign_methods(process_model pm,
concept_model cm, class_model clm)
  for each process in pm do
    classes = node.get_classes(clm)
    inter = null
    if len(classes) > 1 then
      inter = clm.create_or_get_interface(classes, node)

    for each c in classes do
      c.add_method(node)
      if inter != null then
        clm.define_realisation(c, inter)
```

The method assignment to classes allows to also define interfaces and realization relationships in the UML class diagram. So, as a result, the target model consists of classes with methods and attributes, interfaces with methods and five kinds of relationships: generalization, dependency, aggregation, association and realization. An example of generated UML class diagram for the problem domain of a driving school described in Section 4.1 is shown in Figure 4.

C. *Export of the UML Class Diagram to the UML Compliant Tool*

After elements of the two-hemisphere model are transformed into the class diagram, BrainTool gives the possibility to export it in XMI format to be later used in other UML editor or code generators that are able to import UML class diagrams in XMI format. Currently, most UML compatible tools use their own modifications of the XMI format and a developer cannot be sure about the result of import/export [18]. Therefore, the authors were forced to adjust the exported XMI for the requirements of a specific corresponding tool. The Sparx Enterprise Architect [19] is selected for the experiment, and the result of the implemented chain is shown in Figure 5. It is not a problem to define the elements of the UML class diagram according to the specific requirements for import in any other UML tool.

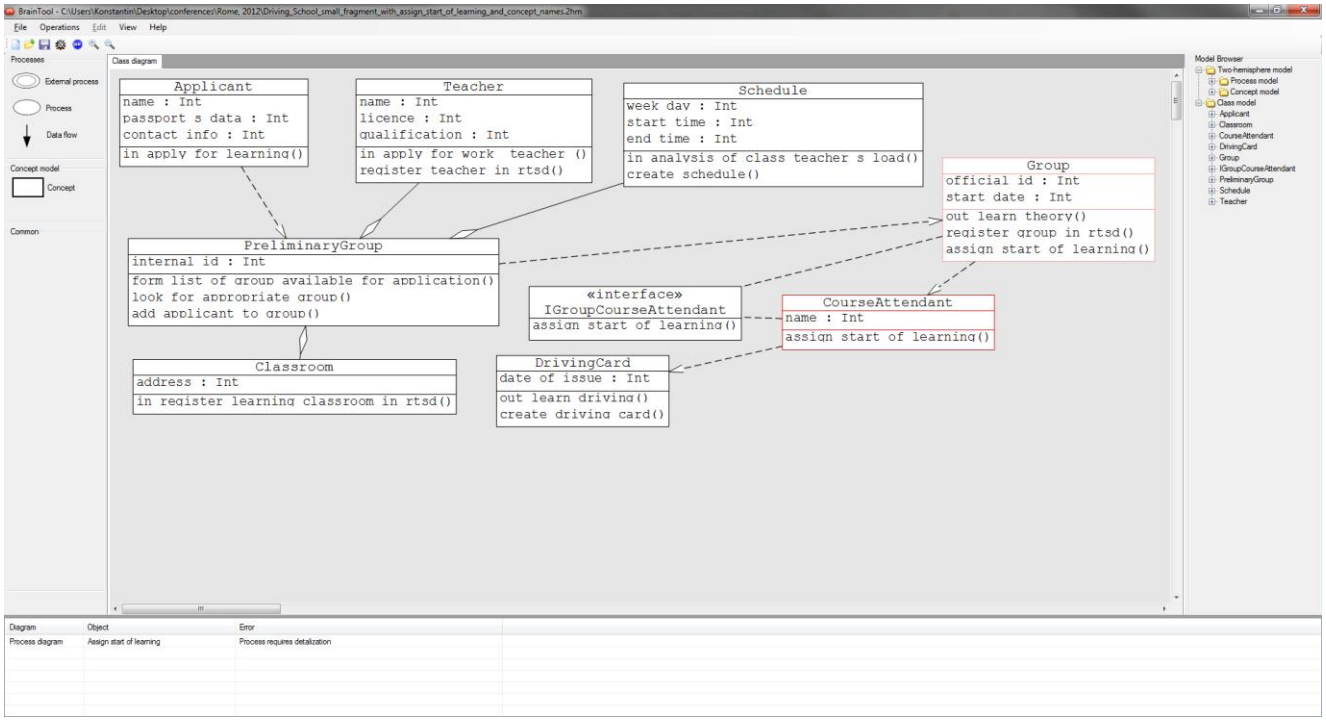

Figure 4. Resulting UML class model for driving school.

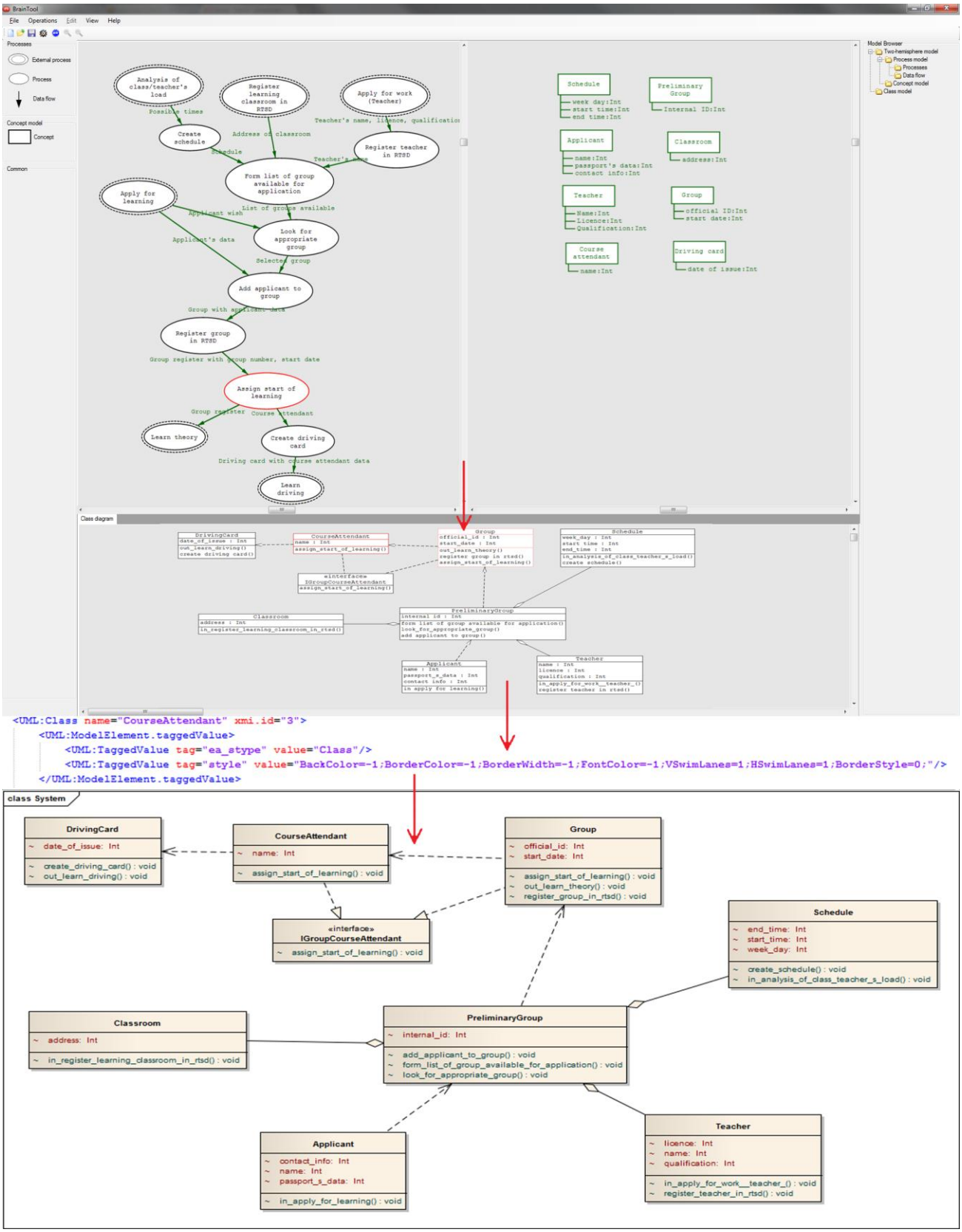

Figure 5. Export to SPARX Enterprise Architect.

D. *Model Validator*

The two-hemisphere model driven approach has several limitations in definition of the UML class diagram. They are expressed in several combinations of incoming and outgoing data flows of certain process. In this case, BrainTool highlights the problematic process in the two-hemisphere model and the potential owner in the UML class diagram.

The modeler is then required to create the sub-process diagram for the highlighted process, as it is shown in Figure 6.

The problematic process is being detailed in the following manner:
A – identify the problem.
B – receive the working area for creation of sub-process diagram
C – create sub-process diagram.
D – confirm sub-process diagram.
E – transfer sub-process diagram into main model.

For example, if there are at least two data flows outgoing from the process, which are typified by different conceptual classes and are differently typified from incoming data flows, the two-hemisphere model driven approach offers to refine the problematic process by dividing it into sub-processes. In order to support these treatments BrainTool gives the possibility to validate the two-hemisphere model developed by the modeler and to define processes, which do not give the clear ability to define a corresponding method's owner for the UML class diagram.

What is more the preliminary structure of the sub-process diagram already contains the incoming and outgoing information flows derived from the main model and the concerned external processes. The modeler is asked to divide the problematic process into a number of separated sub-processes and to define outgoing information flows more precisely (see area B in Figure 6).

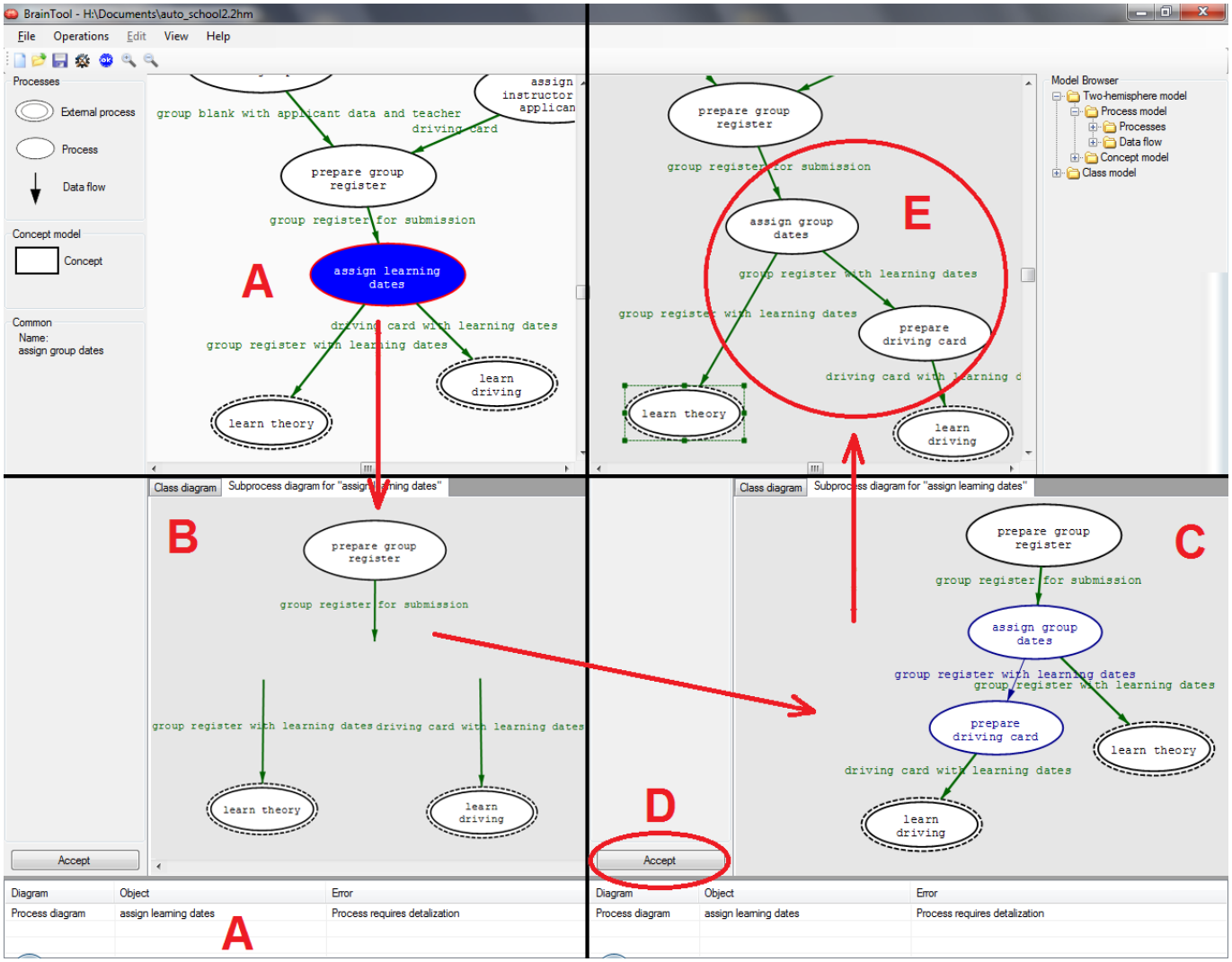

Figure 6. Model validation for the necessity to define the sub-process diagram.

V. CONCLUSION

Nowadays, the usage of model transformations has become a widespread practice and tools supporting such transformations have become increasingly popular. The main goal of the research presented in this paper was to implement a tool, which can generate the UML class diagram from the initial presentation of problem domain. BrainTool gives a possibility to automatically generate the UML class diagram and to export it to any UML compliant modeling environment supporting the XMI [20] format for model interchange. Due to the fact that modern modeling tools use their own variations for model interchange, the authors this time have chosen SPARX Enterprise Architect for integration with BrainTool in XMI import domain and have tuned the output of BrainTool suitable to import it into SPARX Enterprise Architect. The wide use of the unified standard for model interchange in modeling tools will increase the number of tools that can be integrated with BrainTool.

When the essence of the transformation from the source model to the target model is clearly defined, the creation of a tool supporting such a transformation becomes a programming task and it can be solved in two ways. The first way is to use a special transformation language and environments supporting model-driven software development. Another way, which was applied by authors of this paper, is to use general purpose programming languages for regular software implementation and consider the task of developing such a tool as a software development task. The tool specification, including description of the required use cases, was defined.

The definition of source and target models was used to define corresponding data structures; transformation definition was automated using a scripting language. As a result, the authors have implemented a tool that supports creation, editing and validation of the two-hemisphere model and its transformation into the UML class model, where the generated class diagram can further be imported into some UML editor or code generator.

Despite the successful project and the expected result of having a working tool, which enables to draw the initial information about problem domain in the form of the two-hemisphere model and to generate from it the UML class diagram, several problems are left unresolved. One of these problems is a cross-tool model exchange. Various modeling tools support XMI export and import, but, unfortunately, in most cases, the tool defines its own XMI-based format and thus common model interchange standard needs to be defined and implemented. As for BrainTool, the problem was temporarily solved by choosing one concrete tool, namely, Sparx Enterprise Architect, and adjusting the exported XMI schema in correspondence with its requirements.

Another unresolved problem is the layout of the generated diagram. There is no complete algorithm for automatic layout of the UML class diagram, therefore for now BrainTool requires manual layout of the resulting UML class diagram. However, this problem is not tool-specific and the layout algorithm can be integrated with BrainTool at any moment.

The main contributions of the research in comparison with authors' previous papers in the area are as follows:

1) BrainTool has a standalone modeling editor for the two-hemisphere model. We improved the lack of supporting software prototypes developed in 2008, which required import of text files describing the elements of the model.

2) A set of transformations for identification of the elements of the UML class diagram from the two-hemisphere model was refined during the programming to simplify the transition from processed in the process model into operations in classes. Several corrections of the transformation for relationships identification were made during implementation. The approach was improved by the implementation of transformations into the tool.

3) BrainTool has its own model validator, which had not been implemented in the software prototype. It allows identifying processes needed to be refined to complete transformations. This ability gave authors a base for further research of transformation capacity from the two-hemisphere model.

4) An import of the developed UML class diagram into the UML compatible tools bridges the gap between computation independent modeling of the system and software components could be generated from the UML class diagram.

Within the development process, several new possibilities of two-hemisphere approach were investigated. Authors believe that current transformation rules can be improved in order to reduce the number of limitations currently existing in the two-hemisphere model driven approach, to generate a more precise UML class model and more complete set of the class diagram elements for further using this model for code generation. In turn, several new facilities of code generation directly from the two-hemisphere model were also stated.

Authors' future work will be focused on the implementation of a refined version of BrainTool with respect to creation of two-hemisphere model of BrainTool itself and generating the UML class diagram for its development. We expect interesting results in comparison of the UML class diagram "as is" in the current version of BrainTool with the one generated by the tool.

ACKNOWLEDGMENT

The research presented in the paper is partly supported by Grant of Latvian Council of Science No. 09.1245 "Methods, models and tools for developing and governance of agile information systems".

The travel costs and participation fee to conference was supported by the European Regional Development Fund project «Development of international cooperation projects and capacity in science and technology Riga Technical University», Nr. 2DP/2.1.1.2.0/10/APIA/VIAA/003

 ISBN: 978-1-61208-230-1

REFERENCES

[1] K.Vollmer, C. Richardson, and C. Clair, "The importance of matching BPM tools to the process," 2010. Available at: http://www.forrester.com/ (last access in March, 2012).

[2] T. Stahl and M. Volter, "Model-Driven Software Development," Wiley & Sons, 2006.

[3] OMG "UML Unified Modeling Language Specification". 2011. Retrieved from: http://www.omg.org (last access in March, 2012).

[4] P. Rittgen, "Quality and perceived usefulness of process models," 25th Symposium On Applied Computing. ACM, 2010, pp. 65-72.

[5] Website of BrainTool. Available at http://brain-tool.org/ (last access in March, 2012).

[6] O. Nikiforova and M. Kirikova, "Two-Hemisphere Model Driven Approach: Engineering Based Software Development," CAiSE 2004 16th International Conference on Advanced Information Systems Engineering June 7-11, Proceedings, 2004, pp. 219-233. Riga, Latvia.

[7] R. Balzer, "A 15 year perspective on automatic programming," IEEE Transactions on Software Engineering, 11 (No 11), 1985, pp. 1257-1268.

[8] J. Krogstie, "Integrating enterprise and IS development using a model driven approach," 13th International Conference on Information Systems Development – Advances in Theory, Practice and Education. Vasilecas O. et al. (Eds). 2005. Springer Science+Business media, Inc. pp.43-53.

[9] S. J. Mellor, K. Scott, A. Uhl, and D. Weise, "MDA Distilled. Principles of Model-Driven Architecture," Addison-Wesley, 2004.

[10] A. Kleppe, J. Warmer, and W. Bast, "MDA Explained: The Model Driven Architecture – Practise and Promise," Addison-Wesley, 2003.

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

[12] O. Nikiforova and N. Pavlova, "Development of the Tool for Generation of UML Class Diagram from Two-Hemisphere Model," Proceedings of The Third International Conference on Software Engineering Advances (ICSEA), International Workshop on Enterprise Information Systems (ENTISY). Mannaert H., Dini P., Ohta T., Pellerin R. (Eds.), Published by IEEE Computer Society, Conference Proceedings Services (CPS), 2008, pp. 105-112.

[13] O. Nikiforova and N. Pavlova, "Foundations on generation of relationships between classes based on initial business knowledge," Information Systems Development: Towards a Service Provision Society. Springer US, 2009, pp. 289–297.

[14] 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, 2010, pp. 3-12. SciTePress.

[15] H. Peyret and D. Miers, "The Shifting Market For Business Process Analysis Tools," Forester research, 2010.

[16] W. M. P. van der Aalst, "Trends in business process analysis: From validation to process mining," International Conference on Enterprise Information Systems, 2007.

[17] J. Anderson, "Cognitive psychology and its implications," W.H. Freeman and Company, New York, 1995.

[18] O.Nikiforova, N. Pavlova, A. Cernickins, and T. Jakona, "Certification of Model-Driven Architecture Tools: Vision and Application," Proceedings of the ICSEA 2011 - The Sixth International Conference on Software Engineering Advances. Lavazza L. et al (eds.), IARIA ©, Barcelona, Spain, October 23-29, 2011, pp. 393-398.

[19] Sparx, 2012. Sparx Enterprise Architect. Official page of Sparx Enterprise Architect tool. Available at: http://www.sparxsystems.com (last access in March, 2012).

[20] Information Technology, 2005. XML Metadata Interchange (XMI), International Standard ISO/IEC 19503:2005(E), ISO/IEC.

 ISBN: 978-1-61208-230-1