Development of BRAINTOOL for generation of UML diagrams from the two-hemisphere model based on the Two-Hemisphere Model transformation itself

Oksana Nikiforova, Olegs Gorbiks, Konstantins Gusarovs, Dace Ahilcenoka, Andrejs Bajovs, Ludmila Kozacenko, Nadezda Skindere, Dainis Ungurs

Riga Technical University, Faculty of Computer Science and Information Technology, Meza street 1/3, Riga, LV-1048, Latvia
oksana.nikiforova@rtu.lv; olegs.gorbiks@rtu.lv; konstantins.gusarovs@rtu.lv; dace.ahilcenoka@rtu.lv; andrejs.bajovs@rtu.lv; ludmila.kozacenko@rtu.lv; nadezda.skindere@rtu.lv; dainis.ungurs.@rtu.lv

Abstract: *The Unified Modeling Language (UML) is an industrial standard for object-oriented software specification, which offers notational conventions for system modeling at the initial stage of software development. An actual problem is to develop a tool available for automatic generation of UML diagram from some form of the problem domain description. Authors have been introduced such a tool, called BrainTool and developed by a research group in Riga Technical University. BrainTool supports transformation of the problem domain presented as two-hemisphere model into several kinds of UML diagrams and give an ability to export these diagrams for further software development into UML compatible modeling tools and IDEs. Present paper demonstrates the comparison results of the manual UML class diagram creation during the development of the tool, automatic generation of the UML class diagram received by application of BrainTool for the BraintTool itself and so called "to be" class diagram for BrainTool development. These results give a basis to discuss about possible improvements of the transformations offered by two-hemisphere model and potentially "richer" usage of the model for generation of the UML diagrams suitable for further software development.*

Keywords: model transformation tool, two-hemisphere model, UML class diagram, BrainTool.

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. Moreover, the increasing interest towards software development within the framework of Model Driven Development (MDD) (Stahl and Volter, 2006) turns the focus again to the area of model transformation at different levels of abstraction. Unified Modeling Language (UML) (OMG's UML) 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. Currently, 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.

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 have been presented the tool, called BrainTool at the previous AICT conference (Nikiforova et al., 2012a). BrainTool (Website of BrainTool), developed by researchers of the Riga Technical University, is a step forward in the area of automation of the modeling process. Authors of BrainTool propose to generate UML class diagram from the so-called two-hemisphere model (Nikiforova and Kirikova, 2004) of the problem domain, which presents information about processes, information flows between these processes and pre-defined types of these information flows.

Present paper demonstrates the comparison results of the manual UML class diagram creation during the development of the tool, automatic generation of the UML class diagram received by application of BrainTool for the BraintTool itself and so called "to be" class diagram for BrainTool development.

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 remains the main components of BrainTool and the transformations used for generation of the UML class diagram from the two-hemisphere model. Section 4 gives comparison results and approves the correspondence of the generated class diagram to the “to-be” solution. Several conclusions and directions for future research are stated in the fifth section.

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 (Balzer, 1985). Modeling tools developed since that time were oversold on their "complete code-generation capabilities" (Krogstie, 2005). Nowadays, similar situation is observed in modeling tools, using and integrating UML models at different levels of abstraction and automation of software development (Mellor et al., 2004).

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 (Kleppe et al., 2003).

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, Insfran and Abrahao (2010) 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, Insfran and Abrahao (2010) 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.

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 – was introduced by Nikiforova (2002). The title of the approach as the two-hemisphere model driven was defined by Nikiforova and Kirikova (2004), where the hypothesis about how to use two interrelated model to share the responsibilities between object classes was demonstrated on the abstract example and later in a real project (Nikiforova et al., 2006). In both cases the two-hemisphere model was created manually, in the first one by authors and in the second one by an independent problem domain expert. Successfull 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 (Nikiforova et al., 2008). 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 (Nikiforova and Pavlova 2009). 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 (Nikiforova 2010). 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.

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 (Rittgen, 2010), (Peyret and Miers, 2010). 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 (Aalst, 2007). 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 created and intuitively understandable by customer two-hemisphere model can be used for automatically generating class diagram prototype that can be later reviewed and used in software development.

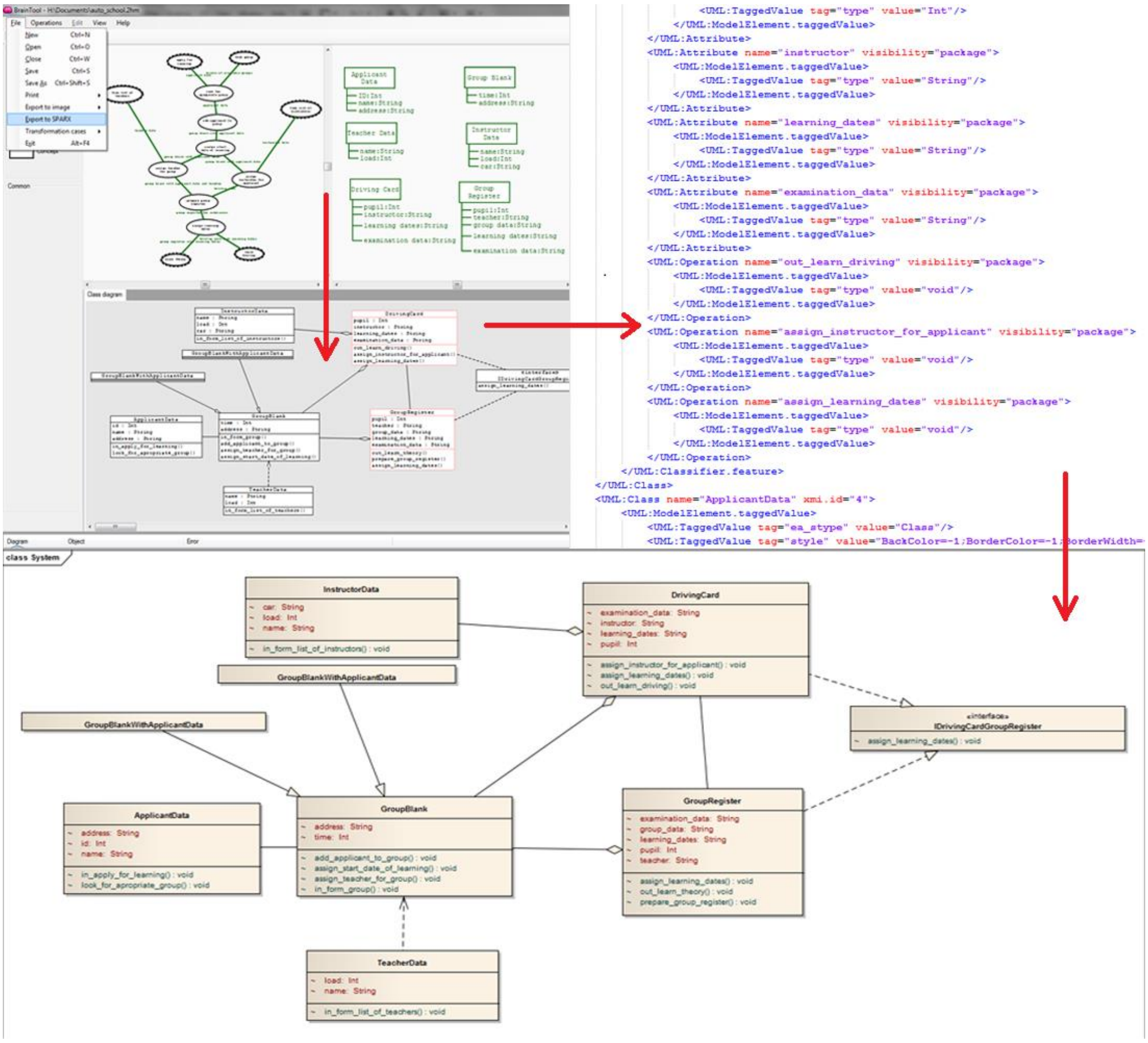
Implementation of two-hemisphere model driven approach in BrainTool

The two-hemisphere model driven approach is based on the transformation of two interrelated models into the UML class diagram. These two initial interrelated models 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 an owner of this process for performing it as an operation in object communication and further is mapped into class responsible for this operation in class diagram.

The current version of the implementation of the two-hemisphere model driven approach can be stated as a standalone tool titled as BrainTool in correspondence with the title of the approach, which in turn is derived from cognitive psychology (Anderson 1995) by analogy with human brain consisting of two interrelated hemispheres. According to Kleppe, Warmer and Bast (2003), 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. Fig.1 demonstrates the general view of BrainTool, transition to XMI file of the generated UNL class diagram, and its import into Sparx Enterprise Architect tool.

Model Editor is a part of the tool providing model creation and modification possibilities. BrainTool is based on a C# implementation using .NET framework 4. Model Repository is the "database" for models, where they are stored. Model Repository is implemented as an XML specification used by both the model editor and the transformation component. An XMI export feature is present, implemented as a Python program. 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. The Transformation Definition Repository is storage for transformation definitions, where a set of Python scripts is used for the current implementation. However, the usage of any console based application understanding BrainTool's XML schema is acceptable. And, 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.

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 (Nikiforova et al., 2011). Therefore, the authors were forced to adjust the exported XMI for the requirements of a specific corresponding tool. The Sparx Enterprise Architect is selected for the experiment, but 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.



```
            <UML:TaggedValue tag="type" value="Int"/>
        </UML:ModelElement.taggedValue>
    </UML:Attribute>
    <UML:Attribute name="instructor" visibility="package">
        <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="String"/>
        </UML:ModelElement.taggedValue>
    </UML:Attribute>
    <UML:Attribute name="learning_dates" visibility="package">
        <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="String"/>
        </UML:ModelElement.taggedValue>
    </UML:Attribute>
    <UML:Attribute name="examination_data" visibility="package">
        <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="String"/>
        </UML:ModelElement.taggedValue>
    </UML:Attribute>
    <UML:Operation name="out_learn_driving" visibility="package">
        <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="void"/>
        </UML:ModelElement.taggedValue>
    </UML:Operation>
    <UML:Operation name="assign_instructor_for_applicant" visibility="package">
        <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="void"/>
        </UML:ModelElement.taggedValue>
    </UML:Operation>
    <UML:Operation name="assign_learning_dates" visibility="package">
        <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="void"/>
        </UML:ModelElement.taggedValue>
    </UML:Operation>
  </UML:Classifier.feature>
</UML:Class>
<UML:Class name="ApplicantData" xmi.id="4">
  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="ea_stype" value="Class"/>
    <UML:TaggedValue tag="style" value="BackColor=-1;BorderColor=-1;BorderWidth=
```

File
Operations
Edit
View
Help
New Ctrl+N
Open Ctrl+O
Close Ctrl+W
Save Ctrl+S
Save As Ctrl+Shift+S
Print
Export to image
Export to SPARX
Transformation cases
Exit Alt+F4
Common
Class diagram
Diagram
Object
Error
class System
InstructorData
~ car: String
~ load: Int
~ name: String
~ in_form_list_of_instructors() : void
DrivingCard
~ examination_data: String
~ instructor: String
~ learning_dates: String
~ pupil: Int
~ assign_instructor_for_applicant() : void
~ assign_learning_dates() : void
~ out_learn_driving() : void
GroupBlankWithApplicantData
GroupBlankWithApplicantData
«interface»
IDrivingCardGroupRegister
~ assign_learning_dates() : void
ApplicantData
~ address: String
~ id: Int
~ name: String
~ in_apply_for_learning() : void
~ look_for_apropriate_group() : void
GroupBlank
~ address: String
~ time: Int
~ add_applicant_to_group() : void
~ assign_start_date_of_learning() : void
~ assign_teacher_for_group() : void
~ in_form_group() : void
GroupRegister
~ examination_data: String
~ group_data: String
~ learning_dates: String
~ pupil: Int
~ teacher: String
~ assign_learning_dates() : void
~ out_learn_theory() : void
~ prepare_group_register() : void
TeacherData
~ load: Int
~ name: String
~ in_form_list_of_teachers() : void

Fig. 1. **General view of BrainTool and import of the generated UML class diagram into the Sparx Enterprise Architect tool.**

Evaluation of the generated UML class diagram

Development of BrainTool gives a possibility to create the two-hemisphere model for different problem domains in turn to analyze possible improvements of the two-hemisphere model driven approach and to approve the correspondence of the generated UML class diagram to the practical usage for software system developments. Authors have been developed the two-hemisphere model, where the problem domain is the two-hemisphere model driven approach itself in order to develop BrainTool. The modeler of the problem domain should in first to model any process of the operating system; then some of the other processes have to be placed to unable the definition of the information flow between any two processes. Information flow, at first, should be created, linked to processes and then it is possible to define the data type for this information flow, etc. Fig. 2 demonstrates a class diagram generated by BrainTool from this two-hemisphere model. Fig.3 demonstrates a class diagram created during development of BrainTool itself and using iterative software development process and best practices.

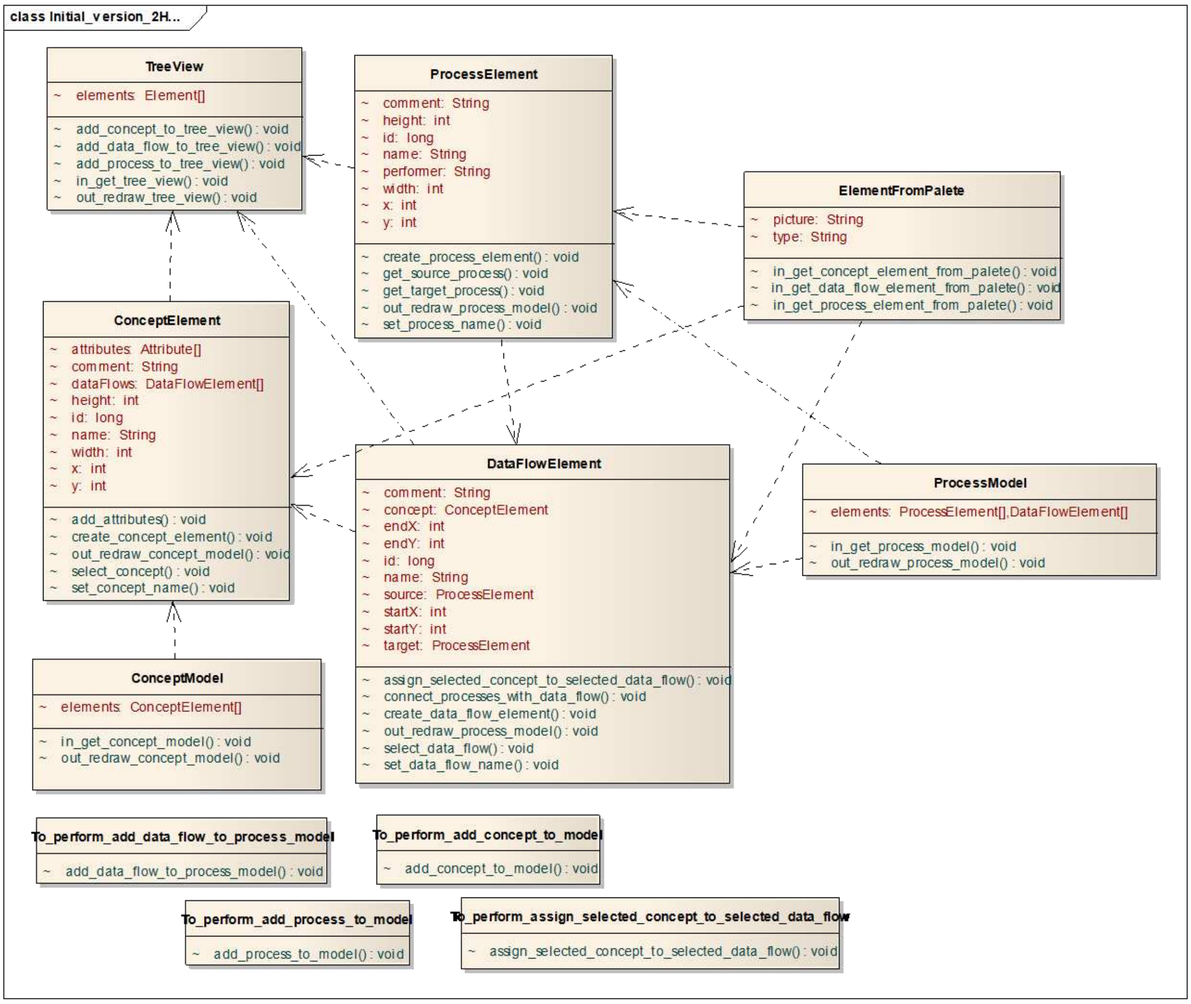

class Initial_version_2H...
TreeView
~ elements: Element[]
~ add_concept_to_tree_view() : void
~ add_data_flow_to_tree_view() : void
~ add_process_to_tree_view() : void
~ in_get_tree_view() : void
~ out_redraw_tree_view() : void
ProcessElement
~ comment: String
~ height: int
~ id: long
~ name: String
~ performer: String
~ width: int
~ x: int
~ y: int
~ create_process_element() : void
~ get_source_process() : void
~ get_target_process() : void
~ out_redraw_process_model() : void
~ set_process_name() : void
ElementFromPalete
~ picture: String
~ type: String
~ in_get_concept_element_from_palete() : void
~ in_get_data_flow_element_from_palete() : void
~ in_get_process_element_from_palete() : void
ConceptElement
~ attributes: Attribute[]
~ comment: String
~ dataFlows: DataFlowElement[]
~ height: int
~ id: long
~ name: String
~ width: int
~ x: int
~ y: int
~ add_attributes() : void
~ create_concept_element() : void
~ out_redraw_concept_model() : void
~ select_concept() : void
~ set_concept_name() : void
DataFlowElement
~ comment: String
~ concept: ConceptElement
~ endX: int
~ endY: int
~ id: long
~ name: String
~ source: ProcessElement
~ startX: int
~ startY: int
~ target: ProcessElement
~ assign_selected_concept_to_selected_data_flow() : void
~ connect_processes_with_data_flow() : void
~ create_data_flow_element() : void
~ out_redraw_process_model() : void
~ select_data_flow() : void
~ set_data_flow_name() : void
ProcessModel
~ elements: ProcessElement[],DataFlowElement[]
~ in_get_process_model() : void
~ out_redraw_process_model() : void
ConceptModel
~ elements: ConceptElement[]
~ in_get_concept_model() : void
~ out_redraw_concept_model() : void
To_perform_add_data_flow_to_process_model
~ add_data_flow_to_process_model() : void
To_perform_add_concept_to_model
~ add_concept_to_model() : void
To_perform_add_process_to_model
~ add_process_to_model() : void
To_perform_assign_selected_concept_to_selected_data_flow
~ assign_selected_concept_to_selected_data_flow() : void

Fig. 2. **Class diagram generated from BrainTool.**

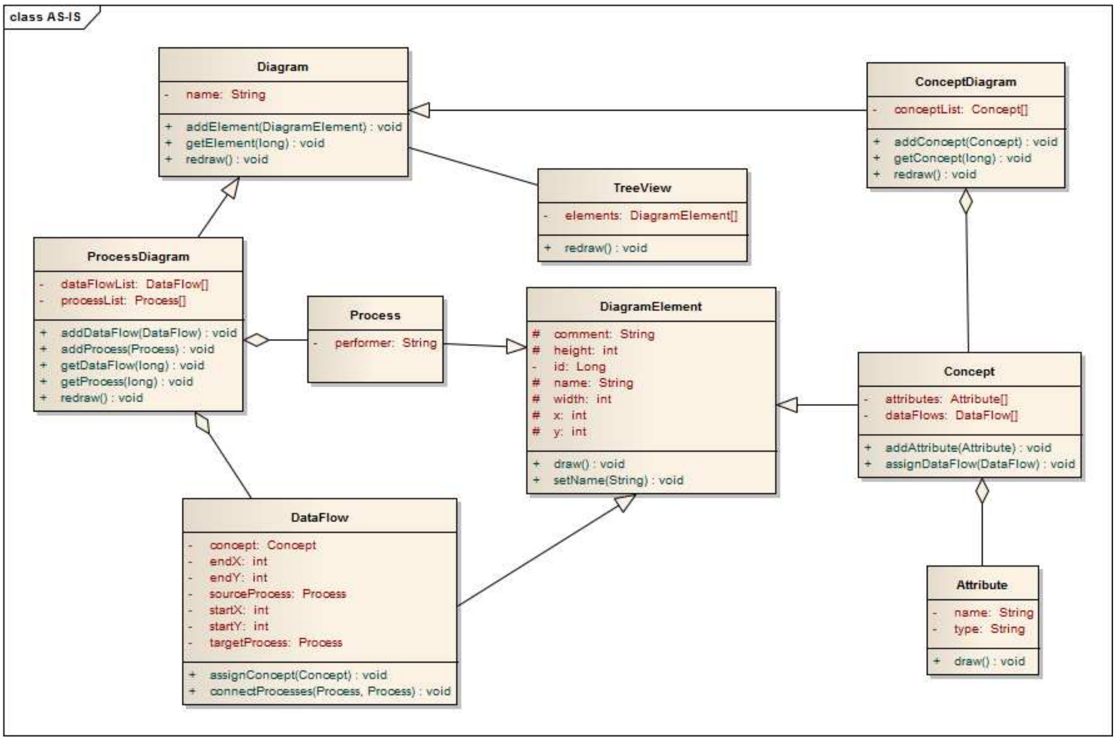


Fig. 3. **Class diagram created during development of BrainTool.**

To perform analysis of the correspondence of the approach supported by BrainTool, three class diagrams are compared each to other. They are the following:

1. The UML class diagram generated by BrainTool. It is shown in Fig. 2.
2. The class structure of BrainTool itself crated by BrainTool's developers manually and can be stated like "AS-IS" model. It is shown in Fig. 3.
3. The UML class diagram representing how should class model look like, which can be stated as "TO-BE" model. This model were obtained based on BrainTool development analysis and contains the improved AS-IS model with purpose to remove logical problems in it. It is shown in Fig. 4.

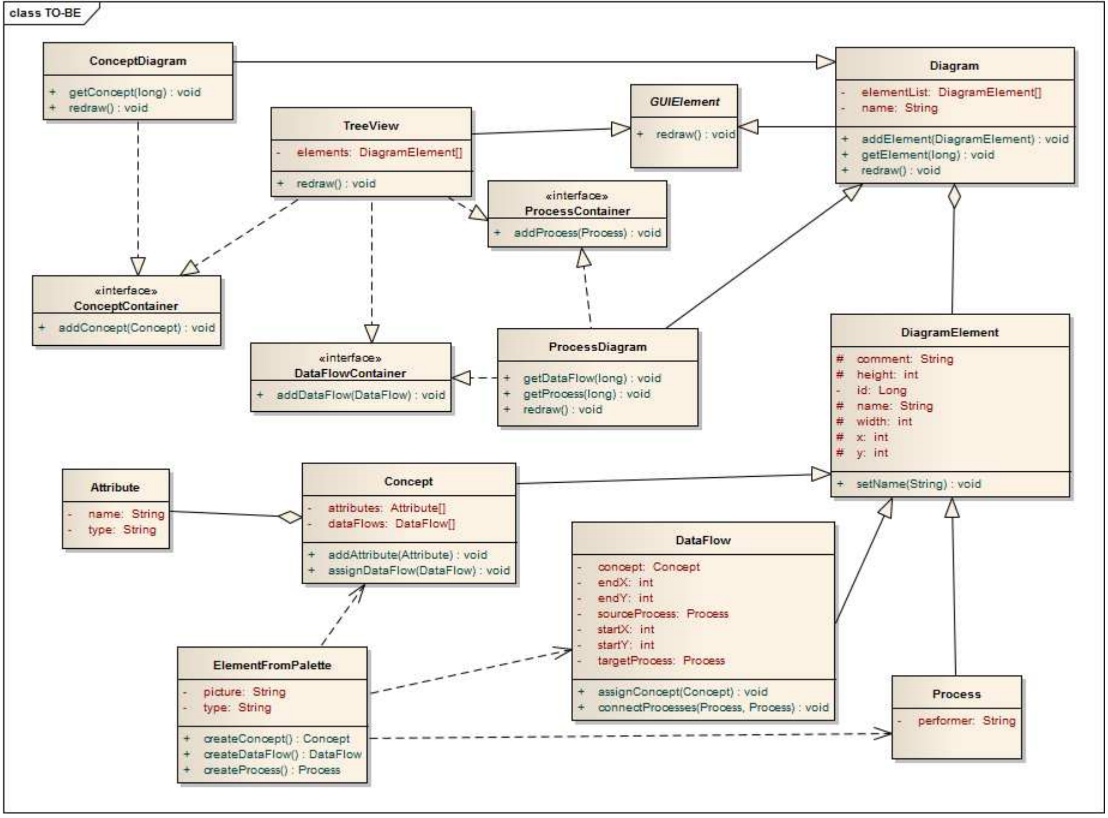


Fig. 4. **TO-BE class diagram.**

To estimate the obtained result, the first two models are compared with TO-BE model. Authors have been selected four comparison criteria describing the core set of the UML class diagram:

1. Classes (how many were generated right classes and interfaces).
2. Associations (how many between obtained associations are right comparison to the TO-BE model).
3. Attributes (how many attributes in each class are the same as in the TO-BE model).
4. Methods (how many methods in each class are the same as in the TO-BE model).

The names, that are in the TO-BE model and in the obtained models may be different, so the comparison were done manually based on obtained elements' semantics. If the meaning of the different named elements is the same, in the comparison process these elements are used as the same elements.

Concerning the classes criteria, authors can claim, that this criteria is good in the Two-Hemisphere approach. All the classes, that is need for the working software were obtained by the automatic transformations in BrainTool. This is expected result, because the class elements is the key elements in the object-oriented software development and reflects the main meaning of the business processes in the software system. However, the current version of BrainTool doesn't allow to generate interface objects for the UML class diagram. These elements might be valuable for the design process result (UML class diagram in our case), but interfaces are not mandatory. As shown on the Fig. 3, the AS-IS diagram doesn't contain interface elements, too. This means, that the resulting code (C# code) isn't so good-looking, but it is working code anyway.

The next criterion is the associations' quality. Authors compared these three models associations. As the result of this analysis, a number of lacks were detected. The main lack is that the generated by BrainTool UML class diagram doesn't contains the generalization association. The ability to create superclasses is quite critical for design process. The next lack is the aggregation correct detection. Though BrainTool provides the possibility to detect aggregation association in the automatic way, the result may be not monosemantic. In this experiment the aggregation associations didn't detect correctly.

The attributes are quite better included in the resulting diagram. Actually, all necessary attributes are including in the BrainTool generated UML class diagram. This means, that all necessary data can be successfully stored in the software objects. However, the picture is a little differs in all three diagrams. Indeed, because of the lack of generalization association, this attributes may be located in the generated UML class diagram in the different classes, than in the AS-IS or TO-BE models. However, the meaning of the attributes, are they located in the class itself, or in the superclass, is the same.

Analyzing the methods, authors can claim, that the main meaning of the methods in all three diagrams is quite close. Of course, because of the some lacks in the BrainTool's UML class diagram generation process the methods can be located differently, and because of the difference in the design process the names of the methods significantly differs. However, these methods must perform the same actions and they have the same meaning in all presented diagrams.

As the result of the comparison of the automatically generated UML class diagram by BrainTool, manually created during the development process and TO-BE model, authors can claim, that BrainTool is able to generate quite good UML class diagram. This means, that the resulting UML class diagram, produced by BrainTool, have many lacks, however, the data amount and quality stored in the BrainTool's diagram is quite close to data amount and data quality stored in the AS-IS model. In the current state, the BrainTool's UML class diagram can be used in the software development and these results give hope that with the improving of the current BrainTool's lacks, the resulting UML class diagram will be better than the AS-IS diagram. In the current state, BrainTool's result is a little worst than the AS-IS diagram in comparison to the TO-BE diagram. However, the deference is not critical and this diagram (obtained by BrainTool) was obtained in the automatic way, which means, that it is cheaper, than AS-IS model, obtained manually.

However, the necessity to improve the existing transformation process is quite clear. The ideas to imrove the approach to increase the quality of the resulting UML class diagram is given in (Nikiforova et al., 2013). These ideas give the possibility to develop the second version of the BrainTool.

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 verify a tool, which can generate the UML class diagram from the initial presentation of problem domain and to export it to any UML compliant modeling environment supporting the XMI format for model interchange. Authors have been implemented BrainTool 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 (Nikiforova et al., 2012a), (Nikiforova et al., 2012b).

The main contributions of the research is a ccomparison of the generated UML class diagram to so called "As-is" and "To-be" diagrams. It approves, that the generated diagram is not worse than the diagram created during the development project. The generated structure has the same classes to support all the processes and conceptual classes defined in the problem domain. And the main benefit is that the structure is received automatically from problem domain presentation. Several distinctions are detected in class' associations. This gives to authors a bases to refine transformations, may be by introducing new elements to the two-hemisphere model.

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 the UML class diagram generated from the two-hemisphere model of BrainTool itself and taking into consideration new statements for transformation improvement.

Acknowledgements

We thank the organizers of conference "Engineering for Rural Development" from Technical faculty of Latvia University of Agriculture for permission to use their publication template for our needs. 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".

References

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

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

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

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

Krogstie, J., 2005. 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). Springer Science+Business media, Inc. pp.43-53.

Loniewski, G., Insfran, E. and S. Abrahao, 2010. 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, pp. 213—227.

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

Nikiforova., O., 2002. General framework for object-oriented software development process In: Scientific Proceedings of Riga Technical University (13), pp. 132–144.

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

Nikiforova, O., Kirikova, M., Pavlova, N., 2006. Two-Hemisphere Driven Approach: Application for Knowledge Modeling, In proceedings of the Seventh IEEE International Baltic Conference on *DB and IS* (BalticDB&IS'2006), O. Vasilecas, J. Eder, A. Caplinskas (Eds.), Vilnius, Lithuania, pp. 244-250

Nikiforova, O., Pavlova, N. and Grigorjev, J., 2008. Several Facilities of Class Diagram Generation from Two-Hemisphere Model In: 23rd International Symposium on *Computer and Information Sciences* (ISCIS 2008). Istanbul, Turkey, October 27-29, [Online]. Available: http://ieeexplore.ieee.org/.

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

Nikiforova, O., 2010. Object Interaction as a Central Component of Object-Oriented System Analysis. In: 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, SciTePress. pp. 3-12.

Nikiforova, O., Pavlova, N., Cernickins, A. and Jakona, T., 2011. 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, pp. 393-398.

Nikiforova, O., Pavlova, N., Gusarovs, K., Gorbiks, O., Vorotilovs, J., Zaharovs, A., Umanovskis, D., Sejans, J., 2012a. 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 Internation Scientific Conference *Applied Information and Communication Technology*, April 26-27, Jelgava, Latvia, pp. 11-19.

Nikiforova, O., Gusarovs, K., Gorbiks, O., Pavlova, N., 2012b. BrainTool. A Tool for Generation of the UML Class Diagrams. Proceedings of the Seventh International Conference on Software Engineering Advances, Mannaert H. et al. (Eds), IARIA ©, Lisbon, Portugal, November 18-23, 2012, pp. 60-69.

Nikiforova, O., Gusarovs, K., Gorbiks, O., Pavlova N., 2013. Improvement of the Two-Hemisphere Model-Driven Approach for Generation of the UML Class Diagram. Scientific Journal of Riga Technical University: Applied Computer Systems, Grundspenkis J. et al. (Eds), No. 2013/14, Riga Technical University © (in press)

OMG UML Unified Modeling Language Specification. 2011. Retrieved from: http://www.omg.org (last access in January, 2013).

Peyret, H. and Miers, D., 2007. The Shifting Market For Business Process Analysis Tools, *Forester research*.

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

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

Website of BrainTool. Available at https://webbraintool.org/ (last access in January, 2013).