e-Informatica Software Engineering Journal, Volume 3, Issue 1, 2009

Two Hemisphere Model Driven Approach for Generation of UML Class Diagram in the Context of MDA

Oksana Nikiforova*

*Faculty of Computer Science and Information Technology, Riga Technical University

oksana.nikiforova@rtu.lv

Abstract

The Model Driven Architecture (MDA) separates the system business aspects from the system implementation aspects on a specific technology platform. MDA proposes a software development process in which the key notions are models and model transformation, where the input models are platform independent and the output models are platform specific and can be transformed into a format that is executable. In this paper principles of MDA and model transformations are applied for generation of UML class diagram from two hemisphere model, which is presented in the form of business process model related with concept model. Two hemisphere model is developed for the problem domain concerned with an application for driving school and UML class diagram is generated using the approach offered in the paper.

1. Introduction

One of the modern research goals in software engineering is to find a software development process, which would provide fast and qualitative software development. Most of currently proposed methodologies and approaches try to make the development process easier and still more qualitative. For achievement of this goal the role of explicit models becomes more and more important. Lately, the most popular approach is Model Driven Architecture [18]. MDA is the central component in the OMG's strategy for maximizing return on investment, reducing development complexity and future-proofing against technological change [29]. MDA tools do not support the complete code-generation capabilities from the initial business information, and the most problematic stage is system modelling based on knowledge about problem domain [22].

The main idea of MDA is to achieve formal system representation at the highest level of abstraction. Nowadays MDA tools support translation of platform independent system presentation into software components and code generation and researchers try to "raise" it as high as possible to fulfill the main statement of the MDA [13]. One of the most important and problematic stages in MDA realization is derivation of PIM elements from a problem domain and PIM construction in the form that is suitable for the PSM. It is necessary to find the way to develop PIM using formal representation, so far keeping the level of abstraction high enough. PIM model should represent system static and dynamic aspects. Class diagram shows static structure of the developed system. But UML is a modelling language and does not have all the possibilities to specify context and the way of modelling, which is always required to be defined in a methodology. Therefore, the construction of class diagram has to be based on well defined rules for its elements generation from the problem domain model presented in the suitable form.

Class diagram discussed in the paper contains classes, relations among them, attributes and operations of classes. Dynamic aspects, which are another meaningful component of system presentation at the platform independent level is not the object of the current research. To obtain the class diagram the initial business knowledge represented with two hemisphere model may be used. The transformation of this model into class diagram is discussed in the paper. The transformation should be defined in formal way and should be acceptable for use in transformation tool. The structure of a transformation tool is discussed in [13] with definition of models, necessary for transformation and transition between these models. Transformation tools take a source model as an input, and create another model, called target model, as an output [13]. Therefore, implementation of transformation needs well-defined set of notational elements of source and target models and definition for transformation of elements of one model into elements of another one. The paper describes class diagram development based on two hemisphere model. Therefore, according to Kleppe's definition source model is defined in terms of two hemisphere model (business process and concept model) and target model is defined in terms of UML class diagram [28]. The structure of the paper is as follows. The next section presents main principles of model driven architecture, defines models to be developed within it, describes transformations to be formalized to be able to develop the tool for support of that transformations. Section 3 presents an information about using of two hemisphere model to fulfil the main statement of MDA corresponds to formal transformations between models. An essence of two hemisphere model is shown in several aspects of its historical evolution and refinement and transformations of two hemisphere model into elements of class diagram are described according to the present state of author's investigations. The transformations presented in the paper are verified in section 4, where the approach offered in the paper is applied for several problem domains. Due to limitations on volume of the paper the section shows only general results on these applications. Section 5 concludes on the research presented in the paper and gives several remarks on author's future work in the area of model transformations.

2. Main Principles of Model Driven Architecture

MDA introduces an approach to system specification that separates the views on three different layers of abstraction: high level specification of how system is working (Computation Independent Model or CIM), the specification of system functionality, i.e. of what the system is expected to do (Platform Independent Model or PIM) and the specification of the implementation of that functionality on a specific technology platform (Platform Specific Model or PSM). In OMG Model Driven Architecture these models are primary artefacts in software developments process and all the activities are concentrated on going from CIM to PIM, from PIM to PSM and from PSM to code. The very important role there is played by the quality of PIM, i.e. its capability to adequately represent system under development [18].

2.1. Models within the MDA

CIM presents the requirements for the system to be modelled in a platform independent model, describing the situation in which the system will be used. Such a model is sometimes called a domain model or a business model. It may hide much or all information about the use of automated data processing systems. A CIM is a model of a system that shows the system in the environment in which it will operate, and thus it helps in presenting exactly what the system is expected to do. It is useful, not only as an aid to understanding a problem, but also as a source of a shared vocabulary for use in other models [18]. PIM is describing that part of information system specification, which is close to code, but is independent of platform specific features. PIM is representing information system in that way that will remain un-

changed on any programming platform. Nevertheless PIM usually is accommodated to specific architecture style [18]. Platform Independent Model is the model that resolves business requirements through purely problem-space terms and it does not include platform specific concepts. The PIM provides formal specification of the structure and functionality of the system that abstracts away technical details. There has to be rules for PIM checking if it defines all problem domain concepts in the correct way [18]. Platform Specific Model is a solution model that resolves both functional and non-functional requirements through the use of platform specific concepts. The platform definition can include wide range of conceptions in the context of MDA. It can be operation system, programming language, any technological platform, such as CORBA, Java 2 Enterprise Edition, also any specific vendor platform (for example, Microsoft .NET) [18]. Platform can imply any of engineering and technological characteristics, which are not important for program unit fundamental business functionality [18].

2.2. Model Transformations within the MDA

Generally, system model refinement and evolution in the framework of MDA is presented in Figure 1.

CIM presents specification of the system at problem domain level and can be transformed into initial elements of PIM. PIM provides formal specification of the system structure and functions that abstracts from technical details, and thus presents solution aspects of the system to be developed. Development of the solution domain model is based on derivation of all the necessary elements from problem domain description (Transformation 1 in Fig. 1). The PIM received as a result of Transformation 1 has to be refined (Transformation 2 in Fig. 1) to get a form suitable for PSM generation, i.e. PIM-refined enables model transformation (Transformation 3 in Fig. 1) to the platform level, named Software Domain in Figure 1.

An MDA idea is promising – raising up the level of abstraction, on which systems are developed, we could develop more complex systems more qualitatively. Core of solution domain development strategies focuses on the transformation of system model from the aspects of business level into the application level (Transformation 2 in Fig. 1). The main idea of MDA is to achieve formal system representation at the as high level of abstraction as possible. Nowadays MDA tools support translation of solution elements into software components (Transformation 3 in Fig. 1) and code generation (Transformation 4 in Fig. 1), and researchers try to "raise" it up as high as possible to fulfil the main statement of the MDA [18].

Transformations 1 and Transformation 2 are defined within different solutions [33], [36], [16], [30], [12], but there is no any solution, where complete transformation CIM $\rightarrow$ PIMinitial $\rightarrow$ PIMrefined would be defined [22].

One of the most important and problematic stages in MDA realization is derivation of PIM elements from a problem domain, and PIM construction in the form that is suitable for the PSM. Solutions that are focused on Transformation 1 can't insure that a PIM contains all the necessary information, and that the presentation of the PIM is formal enough to be able to transform it into the correct PSM, that is to support already the Transformation 3 [22]. It is necessary to find the way to develop PIM using formal representation, so far keeping the level of abstraction high enough, i.e. to implement Transformation 2 in formal way. The central element of PIM is the presentation of system structure, which would be independent from further implementation and usually is presented in the form of class diagram in UML notation [28], as well as adequate presentation of system dynamics. Different modelling tools are used for that. The paper discusses the class diagram development aspects, which satisfies the main statement of MDA and are based on transformation from two hemisphere model into elements of class diagram defined in UML.

Currently, transformations between UML models are still a subject of intensive investiga-

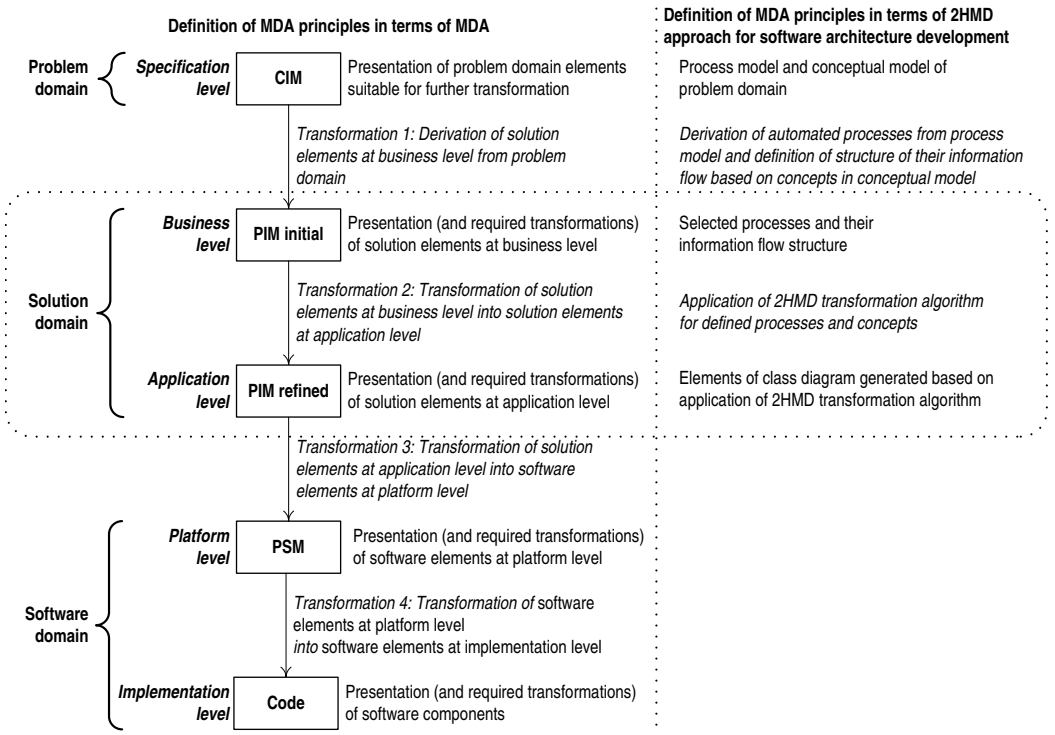

Figure 1. General structure of model transformation in the framework of MDA

tion. Principles of simple language for transformations are presented in [13]. Several proposals [6] are made in response to OMG request for proposals to MOF Query/View/Transformation [6]. The great attention is devoted to UML class diagram development, because class diagram in UML-based CASE systems serves as a main source of knowledge for development of software system: database specification, graphical user interface, application code, etc. [35].

Class diagram is the most often used model for visual representation of static aspects of classes [35]. Class diagrams in object-oriented software development are typically used: as domain models to explore domain concepts; as conceptual/analysis models to analyse requirements; as systems design models to depict detailed design of object-oriented software [1]. But UML is a modelling language and does not have all the possibilities to specify context and the way of modelling, which is required always to be defined in a methodology. There fore the construction of class diagram has to be based on well defined rules for its elements generation from the problem domain model presented in the form suitable for that.

2.3. Structure of a Tool for Model Transformation

The MDA process shows the role that the various models, PIM, PSM, and code play within the MDA framework. A transformation tool takes a PIM and transforms it into a PSM. A second (or the same) transformation tool transforms the PSM to code. These transformations are essential in the MDA development process. The transformation tool takes one model as input and produces a second model as its output. There is a distinction between the transformation itself, which is the process of generating a new model from another model, and the transformation definition. The transformation tool uses the same transformation definition for each transformation of any input model. A transformation is defined in [13] as the automatic generation of a target model from a source model, according to a transformation definition.

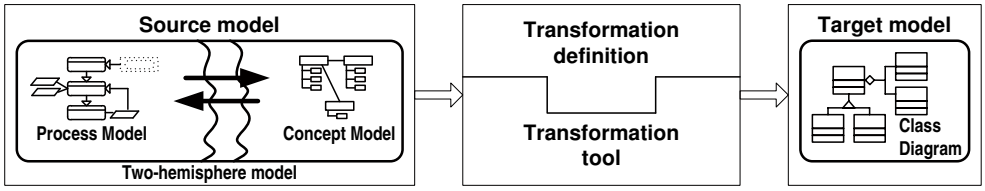

Figure 2. Schema of model transformation tool

And a transformation definition is defined in [13] as a set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language.

The recent tendency of automation of information handling process is essential in industry of information technology [9]. It gives a possibility to spare human and time resources. The implementation of tool, which automates transformation into class diagram, gives a possibility to receive static structure of the system without spending of a lot of time on design. For any transformation the initial data and needed result should be defined before. A transformation tool or approach takes a model on input, so called source model, and creates another model, so called target model, on output, see Fig. 2 [13]. The two hemisphere model has been marked as input with mapping rules, the class diagram and transformation trace has been received on output. Transformation trace shows the plan how an element of the two hemisphere model is transformed into the corresponding element of the class diagram, and which parts of the mapping are used for transformation of every part of the two hemisphere model [18]. Figure 2 shows how a transformation tool takes input – the two hemisphere model and receives output – the class diagram. Therefore implementation of model transformation (in our case transformation from two hemisphere model into class diagram) needs well-defined set of notational elements of source model, well-defined set of notational elements of target model and definition for transformation of elements of one model into elements of another one.

According to key notes of the paper the language for description of source model is defined as a notation for construction of two hemisphere model [21] and the language for description of target model is defined as a notation for construction of UML class diagram (see Fig. 2).

3. Models and Model Transformations in terms of Two Hemisphere Model

According to [32] the significant aspect of real world behaviour seen from the process point of view, where process is understood as the collection of actions, chronologically ordered and influencing objects and is more then "just an amorphous heap of the action". Similarly to the structural modelling of the real world [32]. Two hemisphere model corresponds to both fundamental things – functional aspects of the system defined in terms of business processes and the structural ones defined in terms of concept model. The details in the right column of the table in Figure 1 correspond to the two hemisphere approach, which addresses the construction of information about problem domain by use of two interrelated models at problem domain level, namely, the process model and the conceptual model. The conceptual model is used in parallel with process model to cross-examine software developers understanding of procedural and semantic aspects of problem domain.

3.1. Essence of Two Hemisphere Model

Two hemisphere model driven approach [21] proposes using of business process modelling and concept modelling to represent systems in the platform independent manner and describes how to transform business process models into UML models. For the first time the strategy was proposed in [20], where the general framework for object-oriented software development had been presented and the idea about usage of two interrelated models for software system develop-

ment has been stated and discussed. The strategy supports gradual model transformation from problem domain models into program components, where problem domain models reflect two fundamental things: system functioning (processes) and structure (concepts and their relations). The title of the proposed strategy [21] is derived from cognitive psychology [2]. Human brain consists of two hemispheres: one is responsible for logic and another one for concepts. Harmonic interrelated functioning of both hemispheres is a precondition of an adequate human behaviour. A metaphor of two hemispheres may be applied to software development process because this process is based on investigation of two fundamental things: business and application domain logic (processes) and business and application domain concepts and relations between them. Two hemisphere approach proposes to start process of software development based on two hemisphere problem domain model, where one model reflects functional (procedural) aspects of the business and software system, and another model reflects corresponding concept structures. The co-existence and inter-relatedness of these models enables use of knowledge transfer from one model to another, as well as utilization of particular knowledge completeness and consistency checks [21]. Figure 3 shows the essence of two hemisphere model for an example of an application for driving school.

A notation of the business process model, which reflects functional perspectives of the problem and application domains, is optional, however, it must reflect the following components of business processes: processes; performers; information flows; and information (data) stores [21]. For current research is used business process model constructed with GRAPES [11] notation. Current functional requirements always are present in the business process model that helps to maintain their consistency. As a result sophisticated models are used without disturbing software developers' and business experts' natural ways of thinking [21]. Some recent surveys show that about 80 percent of companies are engaged in business process improvement and redesign [10]. This implies that many companies are common with business process modelling techniques [10] or at least they employ particular business process description frameworks [31]. On the other hand practice of software development shows that functional requirements can be derived from problem domain task descriptions even about 7 times faster than if trying to elicit them directly from users [17]. Both facts mentioned above and existence of many commercial business modelling tools are a strong motivation to base software development on the business process model rather than on any other soft or hard models [21]. The concept model (graph G2 in Fig. 3) is used in parallel with business process model to cross-examine software developers understanding of problem and platform independent models. According to Larman [16] real-world classes with attributes relevant to the problem domain and their relationships are presented in concepts model. It is a variation of well known entity relationship (ER) diagram notation [4] and consists of concepts (i.e. entities or objects) and their attributes. Application of two hemisphere model for generation of class diagram gives a possibility to avoid relations between classes in concept model at business level (of problem domain). Due to transformation of process model into elements of object communication expressed in terms of UML communication diagram it becomes possible to define the relations between classes already according system realization at software level (of implementation domain). Therefore relations between concepts are not shown in concept model in Fig. 3). The notational conventions of the business process diagram gives a possibility to address concepts in concept model to information flows (e.g. events) in process model (see Fig. 3). All elements of the two-hemisphere model stated as source model in Figure 2 are as follows:

– Business process diagram/ Process – business process usually means a chain of tasks that produce a result which is valuable to some hypothetical customer. A business process is a gradually refined description of a business activity (task). Task is an atomic business process unit, which actually de-

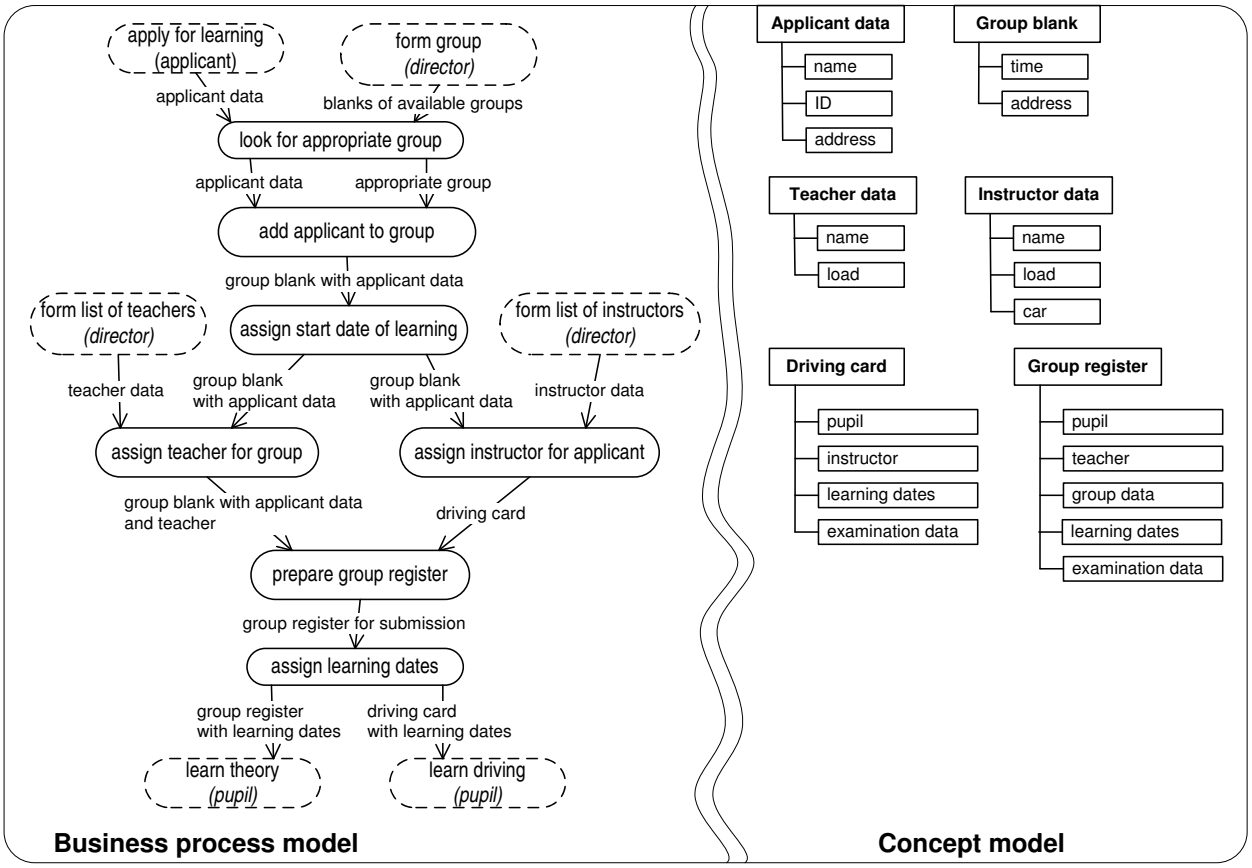

Figure 3. Example of two hemisphere model (application for driving school)

scribes some step or function and is done by a Performer [11].

- Business process diagram/Performer – performer is an attribute of a task of business process and serve as a resources required to perform the activities [11].
- Business process diagram/Event – events are an input/output object (or more precisely – the arrival of an input object and departure of an output object) of certain business process. These objects can be material things or just information [11].
- Concept model/Concept – conceptual classes that are software (analysis) class candidates in essence. A conceptual class is an idea, thing, or object. A conceptual class may be considered in terms of its symbols – words or images, intensions – definitions, and extensions – the set of examples [16].
- Concept model/Concept/Attribute – an attribute is a logical data value of an object [16].

The investigation of two hemisphere model driven approach under the MDA framework in [25] shows that approach could be applied for generation of several elements of class diagram. This paper shows the strategy of two hemisphere model application for generation of UML class diagram in a more precise way. The elements of the class diagram stated as target model in Figure 2 are as follows (only the main elements of the class diagram are listed here):

- Class diagram/Class – a class is the descriptor for a set of objects with similar structure, behaviour, and relationships [28].
- Class diagram/Actor – an actor specifies a role played by a user or any other system that interacts with the subject [28].
- Class diagram/Class/Attribute – an attribute is a logical data value of an object [28].
- Class diagram/Class/Operation – an operation is a specification of a transformation or

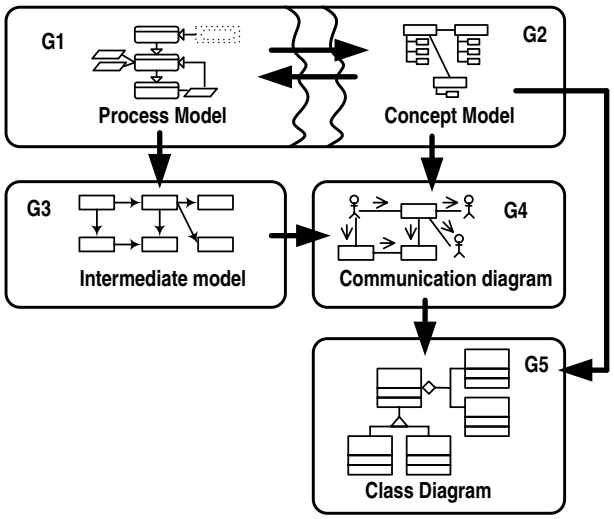

Figure 4. Transformations from two hemisphere model into class diagram under two hemisphere model driven approach

query that an object may be called to execute [28].

– Class diagram/Relationship – a relationship between instances of the two classes. There is an association between two classes if an instance of one class must know about the other in order to perform its work [28].

It is necessary to find the way how source model elements can be transformed into target model elements according to the definition of transformations in the framework of MDA.

3.2. Description of Transformations between Models within Two Hemisphere Model Driven Approach

The two hemisphere model driven approach [20], [21], [27] proposes to apply transformations from business process model into scenarios for object interactions by using so called intermediate model, which is received in a direct transformation way from process model. Appropriate interacting objects are extracted from concept model. Class diagram is based on concept model and is formed according to information about object interaction. All defined transformations from two hemisphere model into elements of class diagram are shown in Figure 4. The schema presents the way how elements of business process model (graph G1 in Fig. 4) and concept model (graph G2 in Fig. 4) are transformed into elements of class diagram (graph G5 in Fig. 4), using intermediate model (graph G3 in Fig. 4) and UML communication diagram (graph G4 in Fig. 4) [25].

Analysis of two hemisphere model proposed in [22] and application of two hemisphere model for knowledge architecture development in the task of study program development presented in [22] makes to think that notational conventions of UML communication diagram is more suitable for definitions of formal transformations of two hemisphere model into object interaction and then into class diagram, than using of UML sequence diagram. Although the aspect of time sequence, which is a component of UML sequence diagram and is not shown in communication diagram, is missed in this case. And author of the paper now is investigating the possibility to save time aspect in transition from two hemisphere model into class diagram through the defined transformations [26].

Intermediate model (graph G3 in Fig. 4 and Fig. 5) is used to simplify the transition between business process model and model of object interaction, which is presented in the form of UML communication diagram (graph G4 in Fig. 4 and Fig. 5).

Intermediate model is a graph generated from business process models using methods of

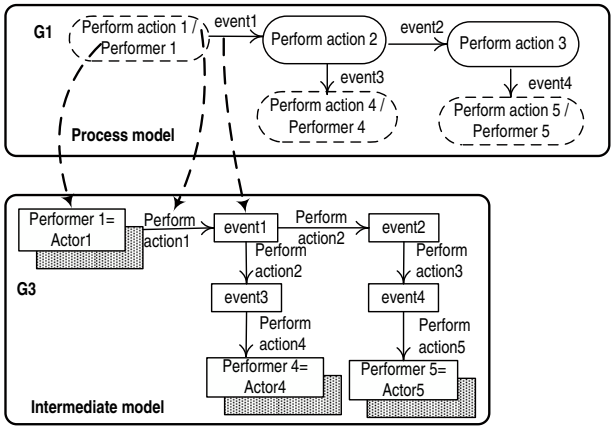

Figure 5. Transformations from business process model into intermediate model

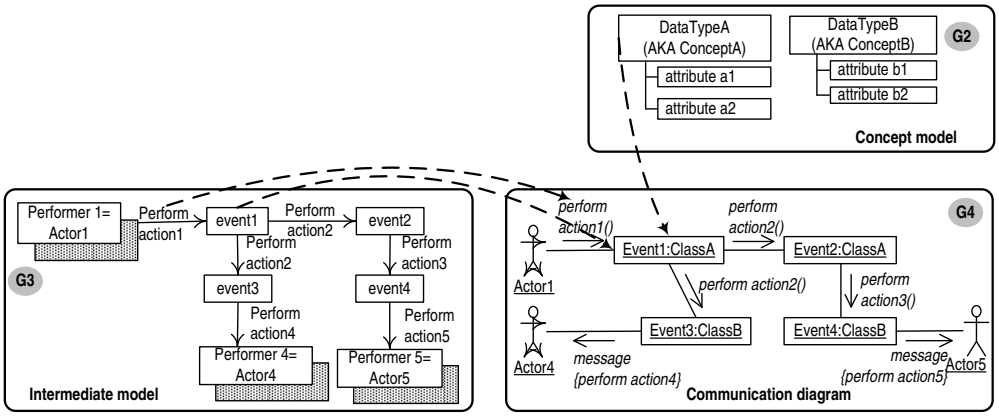

Figure 6. Transformations from intermediate model and concept model into object communication diagram

direct graph transformations based on principles of graph theory [8]. The nodes of the graph G1 in Figure 5 are transformed into the arcs of the graph G3 of Figure 5, and the arcs of the graph G1 in Figure 5 are transformed into the nodes of the graph G3 of Figure 5 [25].

In a case of abstract names of arcs and nodes of graphs in Figure 5 business process "perform action 1" is transformed into an arc "perform action 1" of intermediate model (graph G3 on Fig. 5) and events are transformed into nodes of intermediate model. Constructed intermediate model serves as a base for communication model. The communication diagram is represented as a graph G4 in Figure 4 and Figure 6.

The next transformation defines the method "perform action 1()" in communication diagram (graph G4 on Fig. 6) from the same arc of intermediate model, where the class-receiver of this method is defined as ClassA, because ConceptA defines a data type for event1 in concept model. Therefore if each process is examined as a message, and each data flow as an object, a draft communication diagram could be received by replacing all events of intermediate model with concerned class exemplars and the actions of intermediate model with messages or operations.

The last transformation of this business process defines the responsible class of this method in class diagram (graph G5 in Fig. 4 and Fig. 7) based on information that the type of the event "event 1" is defined by class in. The element "performer 1" is transformed as a node of intermediate model, and as "actor 1" of communication model. This element is defined as "actor 1" in class diagram. Data types for elements "event 1"

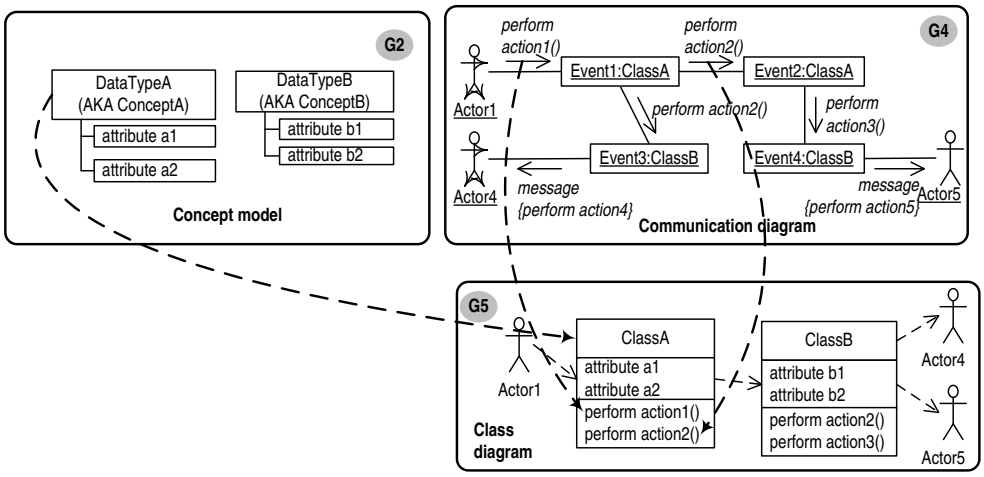

Figure 7. Transformations from intermediate model and object communication diagram into class diagram

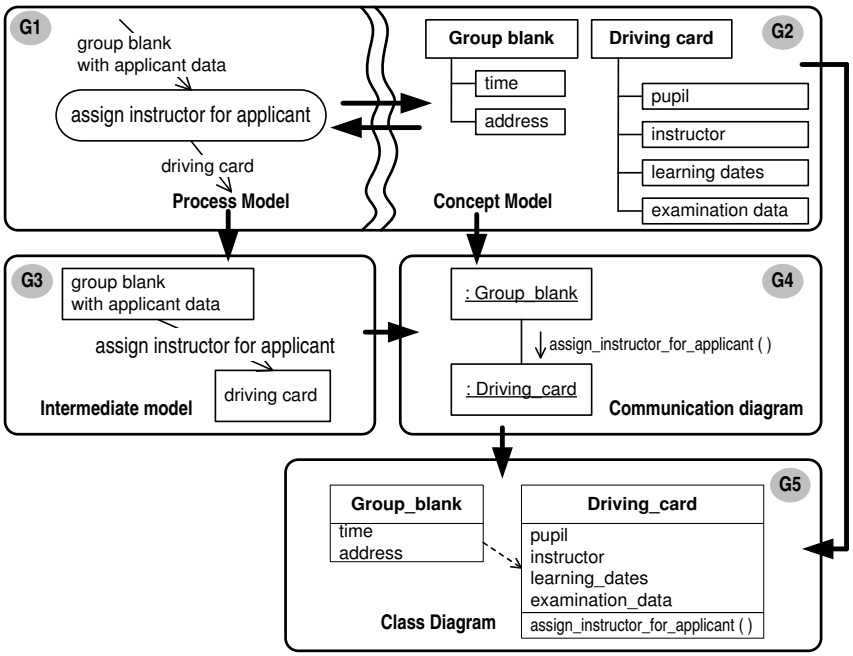

Figure 8. An example of transformation of process and concept elements into class elements

and "event 3" is defined as "DataType A" or "Concept A" of concept model.

These transformations are based on the hypothesis that elements of the class diagram (graph G5 in Fig. 7) can be received from the two hemisphere model by applying defined techniques of graph transformation [8]. The next step of transition is a class diagram. Here all messages of object communication (graph G4 in Fig. 7) are encapsulated as operations of classes using main principles of class diagram development based on information of object communication and all events and concepts defined as objects in graph G4 are defined as classes in class diagram (graph G5 in Fig. 7). Class diagram presents the set of attributes based on attributes defined in concept model.

In a very simple example, transformation described above looks like it is shown in Figure 8, where transformation of fragment of two hemisphere model for driving school into a fragment of the exact class is represented.

There is one process "assign instructor to applicant", which has an input event "group blank with applicant data", where concept "Group blank" with its attributes defines data type for the event, and an output event "driving card", where concept "Driving card" with its attributes defines data type for the event. These interrelated elements define a two hemisphere model, which serve as a base for transformation into intermediate model, with the same names of elements, but different position – arcs of process model are transformed into nodes of intermediate model, and nodes of business process are transformed into the arcs of intermediate model. Intermediate model allows to define communication diagram, where initial process "assign instructor to applicant" is defined as a method. And object-sender and object-receiver are defined in accordance with discussion above. Based on a communication of objects defined, it is possible to construct class diagram according to rules of object-oriented system modelling [20].

Experiments with different combinations of of incoming and outgoing arcs in the model of business process and a variety of different data types defined in a concept model, where the data type can be the same for incoming and outgoing events or different, give a possibility to define different types of relationships between classes. The results of full investigations of all the possible combinations of two hemisphere model, which gives a possibility to define relationships between classes, are shown in [24]. The paper has a discussion of a possibility to share class responsibilities and to encapsulate class attributes and methods according defined transformations from arcs and nodes of two hemisphere model. The paper offers the description of tool for the usage of two hemisphere model for class diagram generation based on the defined transformations.

4. Verification of Transformation within Two Hemisphere Model Driven Approach

When the structures of input and output data are known, it is possible to automate a process of input data transformation into output data.Class diagram generation should consist of four steps according to the application of two hemisphere model (see Fig. 9):

1. construction of two hemisphere model;
2. generation of model elements and their interrelations in some structured form;
3. application of the transformation rules defined (processing algorithm);
4. definition of class specification in well structured form suitable for class diagram construction (for example, XML format).

One of the tools for business process modelling, which gives a possibility to construct two interrelated models (business process and the concept ones), is GRADE [7]. Indeed, GRADE generates text descriptions of model with permanent structure, therefore it is chosen as a tool for development of two hemisphere model and further generation of textual files. It defines all the elements of the model and their relations from one into another. Generated text files serves as an input information into the tool developed and described in [27] in order to support the processing algorithm and XML file, which contains structure of the class diagram required. XML format of class specification gives a possibility to receive visual representation of class diagram in any tool, which supports import from XML for class diagram development. An ability to develop an automated tool for generation of class structure in XML format demonstrated in [27] proves, that transformations discussed in the paper are enough formal for programming. The tool is applied for generation of class diagram from two hemisphere model developed for problem domain of pupil application for learning in a driving school shown in Figure 3. Classes, attributes, operations and relations among classes, which could be determined from the business process diagram and the data structure, were defined applying discussed transformations from business process and concept model to class diagram. Figure 9 represents the structure of class diagram obtained.

One of the limitations of the approach is an impossibility to define the full specification of methods with its arguments. This could be one of the potential directions for future investiga-

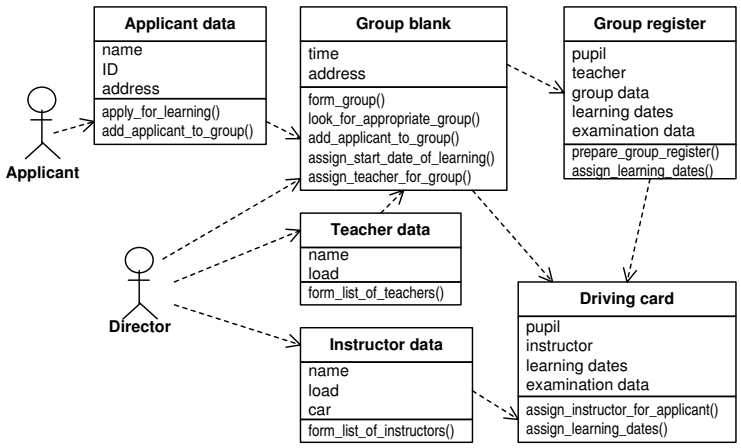

Figure 9. Initial structure of class diagram defined based on transformations from two hemisphere model

tion of application of two hemisphere model for generation of class diagram elements. In order to verify the transformations offered in the paper, the transformations defined for two hemisphere model driven approach in addition to an example of driving school are applied in some more examples: (1) problem area of hotel room reservation, where initial business process model is constructed in GRAPES notation [11] with CASE tool GRADE [7], the two hemisphere model is developed by authors and results of these experiments are shown in [23]; this case approve the possibility to define class diagram by application of the transformations offered in the paper; (2) problem area of insurance system, where initial model is constructed in GRAPES notation with CASE tool GRADE [7], the two hemisphere model is constructed by developers of GRADE tool as demo example and results of these experiments are shown in [25]; this case approve an independence of the transformations offered in the paper from the constructor of two hemisphere model. The problem area of hotel room reservation also is approbated by construction of initial business process model in IDEF0 [14] notation with CASE tool BPWin. Unfortunately, it does not provide construction of concept or object model. Therefore attributes of classes, received for hotel room reservation system with initial information in IDEF0 notation are missing in the class diagram. Even in this case automation of distributing methods among classes is important contribution within software development. Experiments on applying discussed transformations from two hemisphere model into class diagram in different problem domain prove that transformations are independent from problem domain. Experiments on applying transformation from two hemisphere model, constructed in various CASE tools and notations, prove that transformations from two hemisphere model into class diagram are independent from used notation of business process modelling, as well as CASE tool, used for initial model creation.

5. Conclusion

The Model Driven Architecture is the central component in the OMG's strategy for maximising return on investment, reducing development complexity and future-proofing against technological change [29]. But still the "complete code-generation capabilities" are no supported in MDA tools and the more problematic stage is exactly platform independent system modelling based on knowledge about problem domain. Since beginning of eighties a numerous accounts of model generated software systems have been offered to attack problems regarding software productivity and quality [3]. CASE tools developed up that time were oversold on their "complete code-generation capabilities" [15]. Nowadays, similar arguments are exposed to Object Management Group (OMG) Model Driven Architecture (MDA) [34], using and integrating Unified Mod-

elling Language (UML) models [28] at different levels of abstraction. Manipulation with models enables software development automation within CASE tools supported by MDA [5], [13], [19]. The paper discusses abilities on usage of problem domain knowledge presentation in terms of two hemisphere model, which contains two interrelated models of system aspects – process and concept presentation. The proposed transformations are applied to two hemisphere model of application for driving school and classes with attributes and different kinds of relationships are identified based on elements of process and concept models. The ability to define all the types of transformations in a formal way gives a possibility to automate the process of class diagram development from correct and precise two hemisphere model. On one hand, it enables knowledge representation in a form understandable for both business users and system analyst, moreover cover complete and consistent presentation of different system aspects. And on the another hand, it supports the formal transformations of model elements into elements of UML class diagram, which often is a starting point during software development by using nowadays CASE tools, especially in the ones following an idea of MDA. The central hypothesis of this research is that it is possible to apply the graph theory technique for model transformation in the framework of MDA, where the source model is defined in terms of a business process model, associated with a concept model, and the target model is defined in terms of a class diagram. Analysis of the system models presented as a set of graphs developed on the basis of the initial two hemisphere model enables derivation of the class diagram, which is the central component of PIM and is sufficiently detailed in order for the PSM to be generated. Two hemisphere model gives a possibility to define classes with attributes and operations they have to perform, as well as different types of relations can be defined between classes, based on analysis of different combinations of type definition for incoming and outgoing flows of processes of two hemisphere model. At the moment authors try to investigate the possibility of exact definitions of method's arguments based on information in two hemisphere model and to investigate abilities of usage of two hemisphere model for dynamic component of platform independent model expressed in terms of object interaction and state transition. A deeper analysis of notational conventions of nowadays available notations for business process modelling is required and could be stated as one of further researches directions.

Acknowledgements The research reflected in the paper is supported by the research grant No. FLPP-2009/10 of Riga Technical University "Development of Conceptual Model for Transition from Traditional Software Development into MDA-Oriented." And partially the research reflected in the paper is supported by Grant of Latvian Council of Science No. 09.1245 "Methods, models and tools for developing and governance of agile information systems".

References

[1] S. Ambler. *The Elements of UML Style*. Cambridge University Press, 2003.

[2] J. Anderson. *Cognitive Psychology and Its Implications*. W.H. Freeman and Company, New York, 1995.

[3] R. Balzer. A 15 year perspective on automatic programming. *IEEE Transactions on Software Engineering*, 11(11):1257–1268, 1985.

[4] P. Chen. The entity relationship model – towards a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[5] D. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. Woley Publishing, Inc., Indianapolis, Indiana, 2003.

[6] T. Gardner and C. A. Griffin. *Review of OMG MOF 2.0 Query/Views/Transformations Submissions and Recommendations Towards the Final Standards*. Object Management Group. OMG documents ad/03-08-02, available at http://www.omg.org.

[7] GRADE Development Group. *GRADE tools*.

[8] J. Gross and J. Yellen. *Graph Theory and Its Applications*. Discrete Mathematics and Its Applications. Chapman and Hall/CRC, 2nd edition, 2006.

[9] M. Guttman and J. Parodi. *Real-Life MDA: Solving Business Problems with Model Driven Architecture*. San Francisco, CA: Morgan Kaufmann Publishers, 2007.

[10] P. Harmon. Business process management. In *Lecture Notes in Computer Science*, volume 5240/2008. Springer Berlin/Heidelberg, 2008.

[11] INFOLOGISTIK GmbH. *GRADE Business Modeling, Language Guide*, 1998.

[12] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 2002.

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

[14] Knowledge Based Systems Inc. *IDEF Integrated Definition Methods*. Available at http://www.idef.com/.

[15] J. Krogstie. Integrating enterprise and is development using a model driven approach. In *Proceedings of 13th International Conference on Information Systems Development-Advances in Theory, Practice and Education*, pages 43–53. Springer Science+Business media, New York, 2005.

[16] C. Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Prentice Hall, New Jersey, 2000.

[17] S. Lausen. Task descriptions as functional requirements. *IEEE Software*, 20:58–65, March/April 2003.

[18] *MDA Guide Version 1.0.1*, June 2003. Available at http://www.omg.org/docs/omg/03-06-01.pdf.

[19] S. Mellor and M. Balcer. *Executable UML. A Foundation for Model-Driven Architecture*. Addison-Wesley, Boston, 2002.

[20] O. Nikiforova. General framework for object-oriented software development process. *Scientific Proceedings of Riga Technical University*, 13:132–144, 2002.

[21] O. Nikiforova and M. Kirikova. Two-hemisphere driven approach: Engineering based software development. *Advanced Information Systems Engineering*, pages 219–233, June 2004.

[22] O. Nikiforova, M. Kirikova, and N. Pavlova. Principles of model driven architecture in knowledge modeling for the task of study program evaluation. *Databases and Information Systems IV*, pages 291–304, 2007.

[23] 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, International Workshop on Enterprise Information Systems*, pages 105–112, October 2008.

[24] O. Nikiforova and N. Pavlova. Foundations on generation of relationships between classes based on initial business knowledge. *Proceeding of the 17th International Conference on Information Systems Development, Information Systems Development: Towards a Service Provision Society*, August 2008. In press.

[25] O. Nikiforova and N. Pavlova. Open work of two-hemisphere model transformation definition into uml class diagram in the context of mda. *Preprint of the Proceedings of the 3rd IFIP TC 2 Central and East Europe Conference on Software Engineering Techniques, CEE-SET 2008*, pages 133–146, October 2008.

[26] O. Nikiforova and N. Pavlova. Modeling of object interaction with two-hemisphere model driven approach. 2009. Submitted to the 13th East-European Conference on Advances in Databases and Information Systems.

[27] O. Nikiforova, N. Pavlova, and J. Grigorjevs. Several facilities of class diagram generation from two-hemisphere model in the framework of MDA. *Proceedings of 23rd IEEE International Symposium on Computer and Information Sciences*, pages 1–6, 2008. Available at http://ieeexplore.ieee.org/.

[28] Object Management Group. *Unified Modeling Language Specification*. Available at http://www.omg.org.

[29] T. Pokorny. *The Model Driven Architecture: No Easy Answers*, 2005.

[30] T. Quatrany. *Visual Modeling with Rational Rose 2000 and UML*. Addison-Wesley, second edition, 2000.

[31] C. Raistrick, P. Francis, J. Wright, C. Carter, and I. Wilkie. *Model Driven Architecture with Executable UML*. Cambridge University Press, 2004.

[32] V. Repa. Modelling business processes in public administration. *Advances in Information Systems Development. Bridging the Gap between Academia and Industry*, 1:107–118, 2006.

[33] J. Rumbaugh. Omt: The developing process. *Object Oriented Programming*, (8):14–18, 1995.

[34] J. Siegel. *Developing in OMG's Model-Driven Architecture*, 2001. OMG document omg/01-12-01. Available at http://www.omg.org/mda/papers.htm.

[35] T. Skersys and S. Gudas. Class model development using business rules. *Advances in Information Systems Development. Bridging the Gap between Academia and Industry*, 1:203–216, 2006.

[36] D. Tkach, W. Fang, and A. So. *Visual modeling technique: object technology using visual programming*. Addison Wesley, 1996.