ISSN 1407-7493 COMPUTER SCIENCE DATORZINĀTNE 2009-7493

APPLIED COMPUTER SYSTEMS
LIETIŠĶĀS DATORSISTĒMAS

OBTAINING OF ELEMENTS OF UML CLASS DIAGRAM FROM INITIAL INFORMATION ABOUT PROBLEM DOMAIN

UML KLAŠU DIAGRAMMAS ELEMENTU IEGŪŠANA NO SĀKOTNĒJAS INFORMĀCIJAS PAR PROBLĒMAS VIDI

Natalya Pavlova, Dr.sc.ing., leading researcher, Riga Technical University, Meza 1/3, Riga, LV 1048, Latvia, phone: +371 26394784, natalja.pavlova@rtu.lv

Oksana Nikiforova, Dr.sc.ing., assoc. professor, Riga Technical University, Meza 1/3, Riga, LV 1048, Latvia, phone: +37167089598, oksana.nikiforova@rtu.lv

Class diagram, business process model, concept model, MDA, system static structure

1. Introduction

The area of software development encompasses many approaches, methods and techniques. At present, the most frequently used method is Model Driven Architecture (MDA) [1]. MDA is an approach to system specification, which employs three different levels of abstraction. The high level of specification is outlining a general purpose of the system (Computation Independent Model or CIM); then the specification of the system functionality in Platform Independent Model (PIM) follows; and the specification of the implementation of this functionality on a specific technology platform in Platform Specific Model (PSM) is developed [2]. The above models are detailed in the Object Management Group (OMG), the places of transformations are defined and the goal of the application of MDA philosophy is indicated. An MDA-based development process can be divided in three domains – a problem domain (CIM), a solution domain (PIM) and a software domain (PSM) [3].
Class diagram serves as a primary artefact in software development tools based on the idea of MDA. The class diagram is a basic component for solution domain or PIM representation; however, formal generation of the class diagram from the problem domain is still being researched.
Usually developers start class diagram modelling directly before writing of application code, but still they construct the initial architecture defined in terms of the class diagram in manual way reading the requirement specification. The problem arranged with two different kinds of models – a user oriented model and a developer oriented model, can be solved by generation of the class diagram from the initial knowledge about business [3].
A lot of organizations are using different tools for business process analysis and therefore they have complete and consistent models of their organizational structure, responsibilities of employer, business processes and the structure of documentation flows, in other word well structured initial business knowledge [4]. Therefore, the class diagram received in the formal way from the initial business knowledge serve as a bridge between user oriented models and developer oriented models of system. This class diagram helps to implement software closely to initial description of system processes [3].
Nothing new is in the definition of classes and attributes without relationships among them. Software development techniques and methodologies propose a lot of methods to indicate classes in problem area or in initial models. For example, entity-relationship diagrams define how entities should be indicated.
The authors assume that the concatenation of concept model with business process diagram allows to select relationships of classes from initial knowledge. Every concept of concept model defines data structure for one or several data flows on business process diagram. The idea of common consideration of both models is well-known, nevertheless the usage of combination of business

process model and concept model for definition of class responsibilities and relationships among classes in object oriented approach is not widely published.
The authors of the paper in [3], [5], [6] have been proposed how classes and its object's operations can be defined based on so called two-hemisphere model [4], which essence is two interrelated models:

- business process model describes processes, which occurs in the developed system;
- concept model describes the objects, which interacts during the system work.

The purpose of the paper is to find the way of obtaining of such elements of class diagram as classes, its attributes, operations, related classes and kinds of relationships among classes. A base for this definition is information, available in business process and concept model.
As the concept model is part of given initial information, definition of classes and its attributes is not complex task, nevertheless relationship of classes are not obviously defined in initial information. Therefore it can be investigated how to receive possible more precise static structure of developed system.
The assumption, that business process model could be considered as graph with arcs (data flows) and nodes (business processes) is a base of research, described in the paper. The graph presentation gives a possibility to apply transformations of graph theory to business process model.
The second section of the paper discusses essence of two-hemisphere model, which is applied for presentation of initial information about system. The third section describes algorithm, proposed for receiving of classes, its attributes and operations. The fourth section discusses algorithm of definition of kinds of relationships among classes. The fifth section is conclusion, where results of the paper are discussed.

2. Essence of two-hemisphere model

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 model driven (2HMD) 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 coexistence 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 [4]. Fig. 1 shows the schema of 2HMD approach for system development [7].

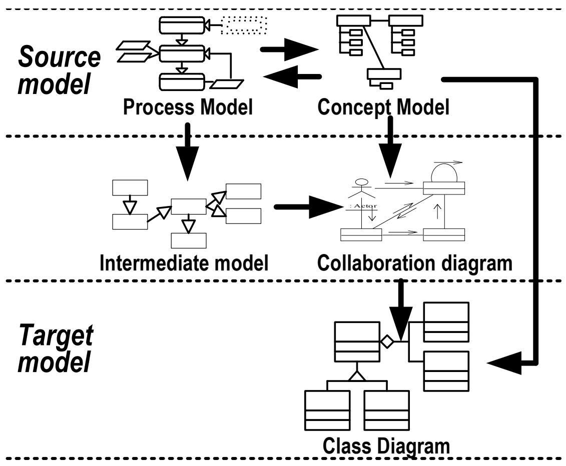

Fig.1. Transformations from two-hemisphere model into class diagram under 2HMD approach

A notation of the business process model is optional, however, it must reflect the following components of business process model: processes, performers, information flows, and information (data) stores [4]. Real-world classes relevant to the problem domain and their relationships are

presented in concept model. It is a variation of well known entity relationship (ER) diagram notation [8] and consists of concepts (i.e. entities or objects) and their attributes. The notational conventions of the business process diagram give a possibility to address concepts in concept model to information flows (e.g. events) in process model.
Table 1 presents elements of the business process model [9]. The main elements are listed in the first column of Table 1. The second column describes the meaning of these elements.

Table 1

Elements of business process and concept model

Name of element	**Description of element**
Business process diagram/ Process process name; description; triggering condition; performer; expression; duration; start option; end option; no start option; assignment	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) [10].
Business process diagram/Event name; transfer name; set option; repeat option	Events are defined (as a rule) in the moment when they are mentioned for the first time in BP or TD diagrams. 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 [10].
BP diagram/Data store store name; comment; ER model<ER name>	The data store is a persistent (independent of the current task) storage of data or materials. In the case of an information system, the data store most likely will be converted to a database with a certain data structure (Entity Relationship Model). On the highest levels of business models, the data store can be used to denote an archive of data or it can also be used to represent a warehouse or stock of goods [10].
Concept model/Concept name	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 [11].
Concept model/Attribute name; type	An attribute is a logical data value of an object [11].

The elements of business process and concept model are important for further system analysis and design. Information from these elements is significant, and nothing from it should remain without any usage on the lowest levels of abstraction. Business processes or tasks present system functionality, its activities and operations. If this information is lost it is necessary to find system functions for implementing operations of classes in description of the problem domain. Events, data stores and data objects are parts of the data structure model. With these elements initial static structure of system is presented in the two-hemisphere model, and should be transited into the class diagram.
The main elements of the class diagram are listed in Table 2 [9]. The list of elements of class diagram is presented in the first column. The second column has the description of the listed elements.
To receive one model from another is necessary to apply some transformation. Essence of transformation is the following: transformation tool or approach takes a model on input and creates another model on output [12]. The two-hemisphere model has been marked as input with mapping rules, the class diagram and transformation trace has been received on output [9]. 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.

Table 2

Elements of class diagram

Name of element	Description of element
Class diagram/Class	A class is the descriptor for a set of objects with similar structure, behavior, and relationships [11].
Class diagram/Class/Attribute name; type	An attribute is a logical data value of an object [11].
Class diagram/Class/Operation name; return type; argument; precondition; postcondition	The UML formally defines operations. To quote: "An operation is a specification of a transformation or query that an object may be called to execute" [13].
Class diagram/Relationship type; multiplicity; role	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 [11].
Class diagram/Class/Stereotype	Stereotypes, which provide a way of extending UML, are new kinds of model elements created from existing kinds [11].
Class diagram/Constraint	A constraint is a condition that every implementation of the design must satisfy [11].

The purpose of this research is to find the way how elements of business process model can be transformed into elements of class diagram according to the definition of transformations in the framework of MDA.

3. Obtaining of elements of class diagram

As it is said above for obtaining of elements of class diagram the diagrams of two-hemisphere model and communication diagram are considered as graphs, for which transformations of graphs could be applied. Fig. 2 presents a sketch of two-hemisphere model, where both – business process and concept models are presented as graphs [7].

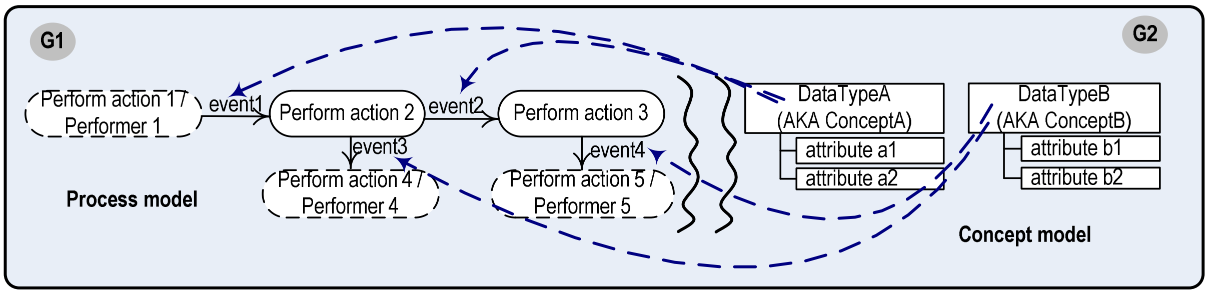

Fig.2. Sketch of initial information in form of two-hemisphere model

For current research the first hemisphere – business process model is constructed with GRAPES [10] notation. The second hemisphere is concept model. C.Larman defines concept model as "The concept model captures real-world concepts (objects), their attributes, and the associations between these concepts" [11]. Graph G1 which is business process model consists of five processes (Perform action 1, Perform action 2, Perform action 3, Perform action 4 and Perform action 5); three performers (Performer 1, Performer 4 and Performer 5) and four events (event 1, event 2, event 3 and event 4).

Graph G2 which is concept model consists of two concepts – DataTypeA and DataTypeB, where DataTypeA defines data type for event 1 and event 2 of business process model, and DataTypeB defines data type for event 3 and event 4 of business process model.
For performing of transformation to class diagram the intermediate model (graph G3 on Fig.3) is introduced in [7]. Intermediate model is used to simplify the further transition between business process and object interaction models, which now is presented in the form of UML communication diagram. Intermediate model is generated from business process model using methods of directed graph transformation [14], when arcs of one graph (G1 on Fig. 3) are transformed into nodes of another graph (G3 on Fig. 3) and nodes of one graph (G1) are transformed into arcs of another graph (G3) [3]. Fig. 3 presents the transformation from two-hemisphere model to intermediate model (graph G3 on Fig.3) [7].

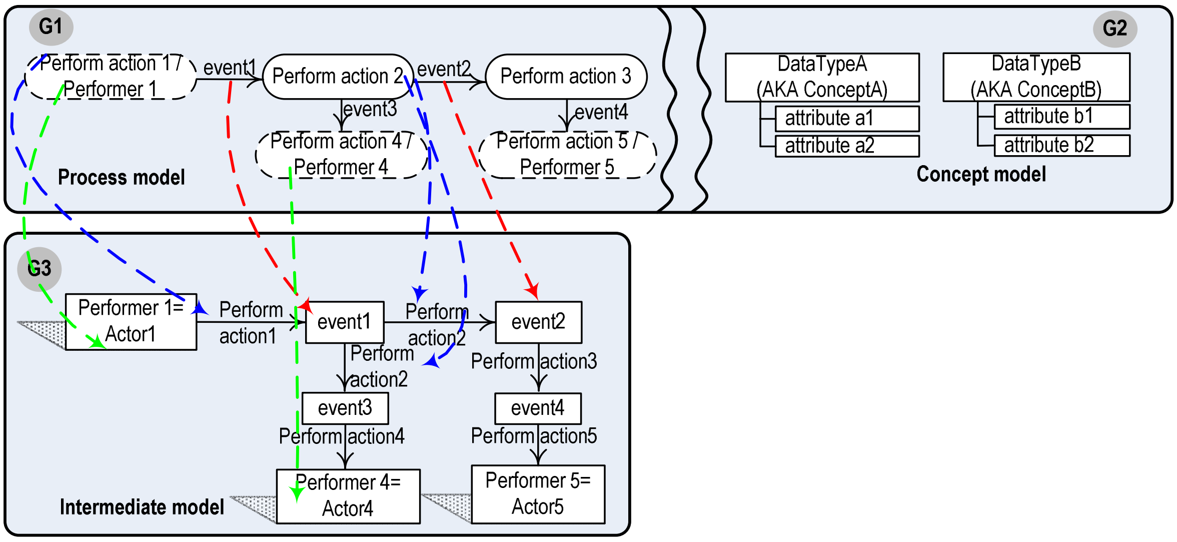

Fig.3. Forming of intermediate model

The transitions of elements are shown with coloured arrows. Nodes of business process model (business processes Perform action 1, Perform action 2, Perform action 3, Perform action 4 and Perform action 5) are transformed into arcs of intermediate model with the same names (see blue arrows on Fig.3 for example). The elements like “Performer 1”, “Performer 4” and “Performer 5” are transformed as a node of intermediate model, example of this transition is shown with green arrows. And arcs of business process model (events event 1, event 2, event 3 and event 4) are transformed into nodes of intermediate model with the same names as on business process model (see red arrows on Fig.3 for example).

3.1. Construction of communication diagram

The business process model and concept model gives enough information for constructing of communication diagram of UML. The intermediate model is constructed to receive a base of communication diagram [7]. As since nodes of intermediate model are received from arcs (or events) of business process model, we know data types for all nodes of intermediate model which are defined by concept of concept model. Thus all nodes of intermediate model serves as a base for objects of communication diagram (red arrows from G3 to G4 in Fig.4), in the same time data type or concepts of conceptual model serves as a base for definition of class for each object (red arrows from G2 to G4 in Fig.4). Finally communication diagram has four objects, which belong to two classes – objects Event1 and Event2 belong to ClassA, as DataTypeA define data type for event 1 and event 2 of business process model and objects Event3 and event 4 belong to ClassB, as DataTypeB define data type for event 3 and event 4 of business process model. Actors of communication diagram are defined based on performers of process model, which are transformed into actors on intermediate model. The example of receiving of actors is shown with green arrows in Fig.4.

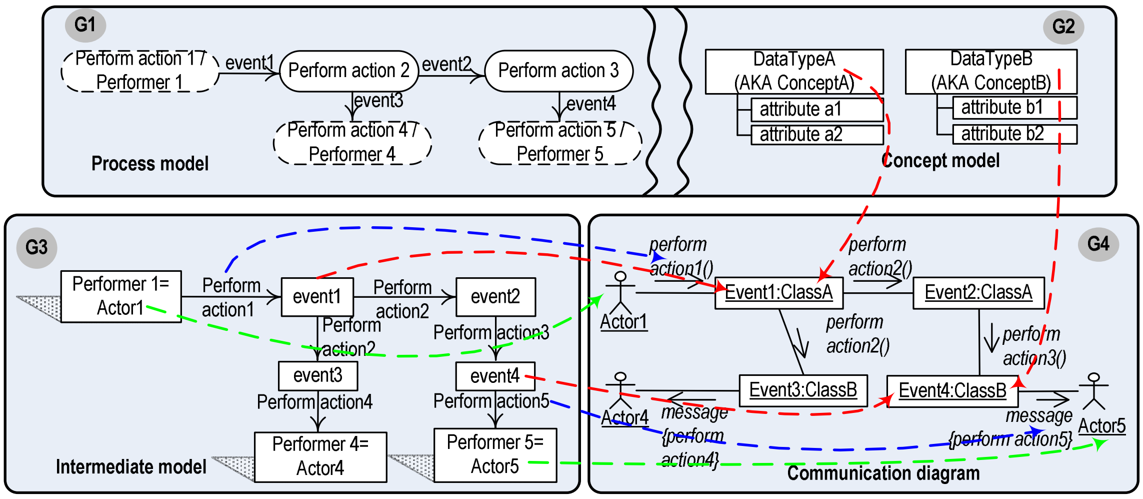

Fig.4. Obtaining of communication diagram

Operations and messages for objects of communication diagram are received from business processes, or arcs of intermediate model (see blue arrows on Fig.4). Not all business processes are transformed into operations. Case, when actor receives some information is different. Not operation, but only message is received by actor in this case. Thus, three operations are defined in described case – perform action 1(), perform action 2() and perform action 3(). Operation perform action 2 () is executed from two objects simultaneously. It means, that in real case initial business process Perform action 2 requires detailed elaboration in initial stage.
The result of transformation, described in this subsection is graph G4, which is communication diagram with actors, objects, messages and operations.

3.2. Definition of operations and related classes

The communication diagram gives enough information for formal receiving of class diagram. Methods, which propose algorithm for receiving of classes and operations from communication diagram, exist, one of these methods is proposed by C.Larman in [11], another is described in [13]. Possibility of obtaining of elements of class diagram is described in this subsection.
With communication diagram and concept model is possible to define classes, its attributes, operations and relationships. Scheme of class with attributes obtaining is resented in Fig.5. Coloured arrows show the transformation way from initial information to class diagram.
It is well known, that objects of communication diagram is classes of class diagram [11], therefore we can define two classes – ClassA and ClassB based on communication and concept model (red arrows from communication to class diagram in Fig.5). Attributes for these classes are defined from concept diagram (see red arrows from concept to class diagram in Fig.5), each class has two attributes. Class diagram contains not only usual classes, but also actors. Obtaining of actors is shown on Fig.5 with green arrows. Actors are defined in intermediate model based on information about performers of business process model. These actors are transited to communication diagram as executor of some process, or sender of message; the same actors appear on class diagram.
Operations are important part of class diagram. Business process model, which is transformed into intermediate model, contains enough information for definition of operations.
The operations are defined during communication diagram. The same part of communication diagram – messages, and its executors serves as a base for determination of related classes. Blue arrows show obtaining of operations. Class diagram, which is graph G5 in Fig.5 contains too classes and three actors. For ClassA are defined operations perform action 1() and perform action 2().

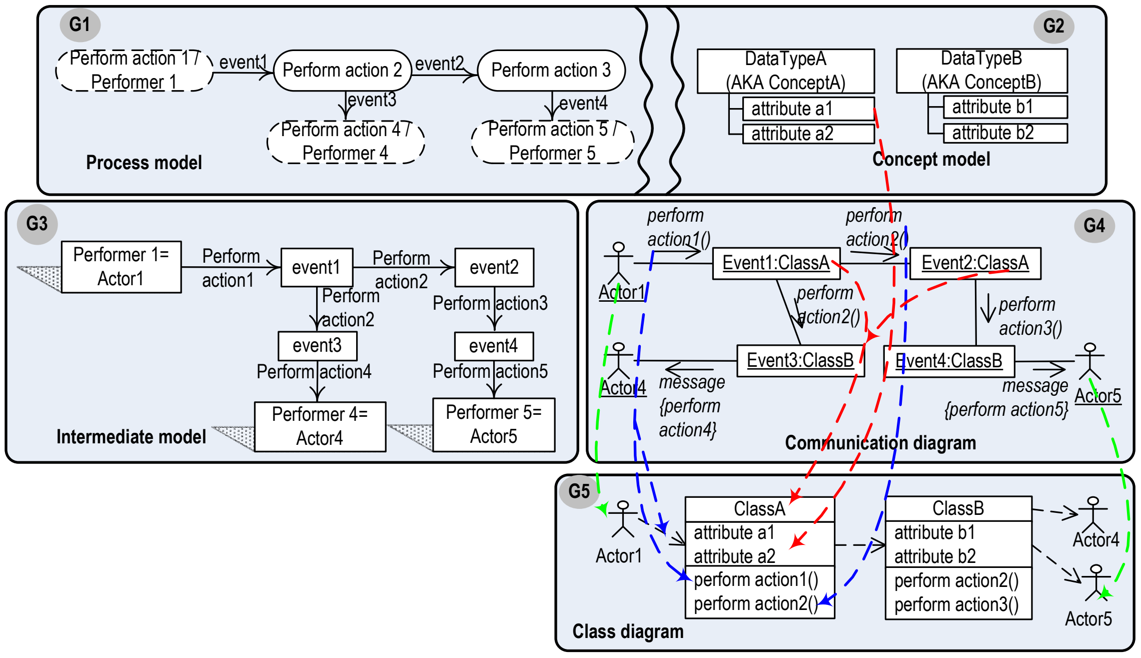

Fig.5. Obtaining of elements of class diagram

For class ClassB are defined operations perform action 2() and perform action 3(). The same as on communication diagram operation perform action 2() belongs to both ClassA and ClassB simultaneously. With messages of communication diagram is defined not only operations, but related classes too. If one class executes operation from another, or sends message to another class, then these classes are related on class diagram. This dependency is shown with blue arrows in Fig.5.
Now is received class diagram, with defined classes, attributes, operations and relationships among classes.

4. Definition of kinds of relationships

The relationships of classes are defined based on information presented in business process and concept models. The combinations of input and output data flows allow to define how classes corresponding to these data flows are related to each other. During the research of the possibility of indications of class relations a set of combinations of data flows and business processes is investigated, and only combinations, which give a possibility to define aggregation, generalization or dependency [6].
For investigation of obtaining of class diagram elements the business process diagram is split to fragments (or transformation cases), each of it includes one business process and all data flows which are input and output of the business process. These cases are selected so as a lot of combinations would be examined [15]. The paper shows one example for definition of each kind of relationship.

4.1. Aggregation

First discussed relationship is an aggregation. Aggregation is the part-of relationship [16]. Example how the aggregation could be defined is presented with transformation case 1, it has three different data flows – two on input and one on output as it is shown in Fig.6.

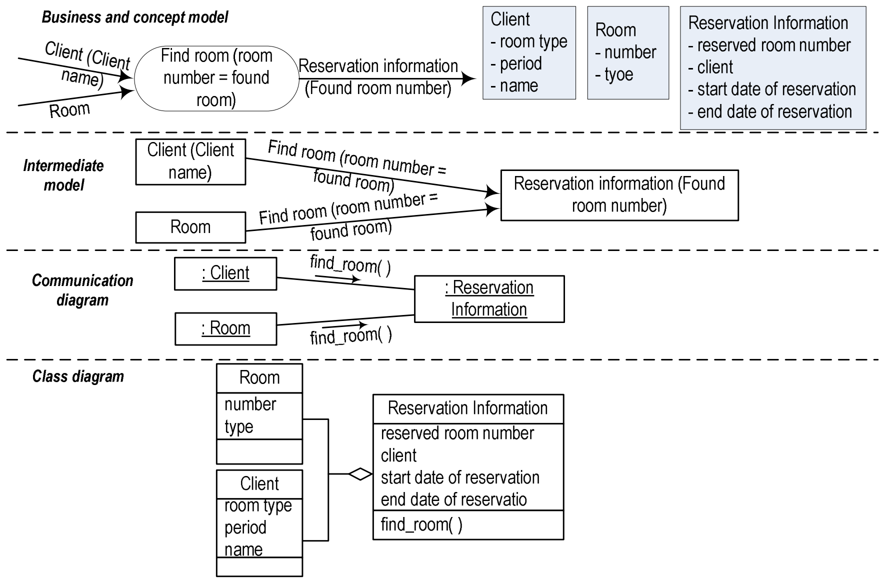
Business and concept model
Client (Client name)
Room
Find room (room number = found room)
Reservation information (Found room number)
Client - room type - period - name
Room - number - tyoe
Reservation Information - reserved room number - client - start date of reservation - end date of reservation
Intermediate model
Client (Client name)
Room
Find room (room number = found room)
Reservation information (Found room number)
Communication diagram
: Client
: Room
find_room( )
: Reservation Information
Class diagram
Room
number
type
Client
room type
period
name
Reservation Information
reserved room number
client
start date of reservation
end date of reservatio
find_room( )

Fig. 6. The business process, data structure, intermediate, communication and class diagrams for transformation case 1

The data structure consists of three objects ”Client”, “Room” and “Reservation Information”. There are the following data flows in the process diagram: “Client (Client name)”, “Room” and “Reservation information (found room number)”.
The fragment of business process and concept diagrams gives enough information to the construction of the class diagram fragment [15]. It is important to note that there is missed an external information, except output data flow “Reservation information (found room number)”, therefore class “Reservation Information” consists of “Client” and “Room”.
Construction of intermediate model and the communication diagram is necessary for receiving the elements of class diagram. These models will help to define classes, their operations and relations among them. Intermediate model is received as it is described in section 3 – using issues of graph theory and transformations. The operation “find_room” of the class “Reservation Information” is received from the communication diagram in Fig. 6. Thus this transformation case, where two input and one output data flows are presented, allows defining classes and operations, as well as the aggregation among these classes. Class “Reservation Information” aggregates classes “Room” and “Client”, as since it is received from joining of these to classes [6].

4.2. Dependency

A dependency exists between two elements if changes to the definition of one element (the supplier) may cause changes to the other (the client) [16]. Transformation case 2 on Fig. 7 presents case with two inputs and one output. There is a variation of one type of data flow on input. And variation of another type of data flow on output. The example chosen for illustration of this case is process of room finding by request from a client. The found room is received as a result of the process “find the required room”. Information necessary for this operation is “Room type” and “Period”. This information is taken from the object “Reservation Blank” (see the fragment of business process diagram and the corresponding concepts in Fig. 7). Arcs of business process model “Reservation Blank (Room type)”, “Reservation Blank (Period)” and “Room (Found room number)” should be transformed to nodes according to the proposed graph transformations [14], [15]. There is only one process “Find required room” in Fig.7. It means that nodes “Reservation Blank (Room type)” and “Reservation Blank (Period)” should be joined with “Room (Found room number)”using the arc “Find required room” as it is shown in Fig.9. Therefore, intermediate model consists of three nodes and two similar arcs.

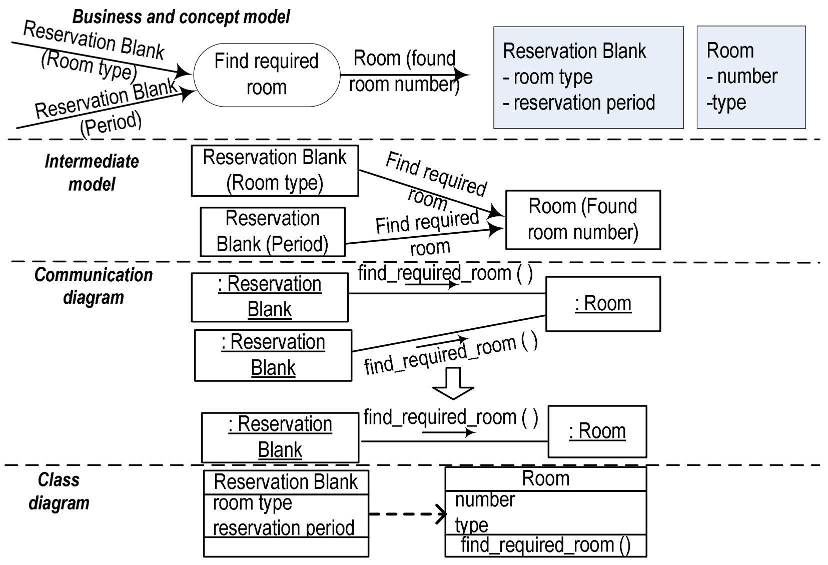

Fig. 7. The business process, data structure, intermediate, communication and class diagrams for transformation case 2

Fig.7 shows duplicated nodes in intermediate model and objects in the first variant of the communication diagram. It is permissible for the intermediate model to have two similar nodes with the same processes on output. This duplication should be rid in the communication diagram. Making use of the concept view, where data structure is defined the duplicate objects could be "joined" in one object "Reservation blank", which sends the message "require_client_data ()" to the object "Room", where object room is received joining the node of intermediate model "Room (found room number") and the data structure "Room".
Now the communication diagram is received, and it is possible to get classes from it. Based on the communication diagram (see Fig.7) it can be concluded that two classes correspond to the discussed case, one of which executes an operation from another, and change of one class may cause changes to another. Bottom part of Fig.7 presents classes, which are received from initial data of this case [6].

4.3. Generalization

Generalization allows to indicate superclasses and sublcasses in class diagram [16]. Transformation case 3 belongs to a series of cases, wherein are multiple outputs. An example chosen for the simplest case is the process "Check room status", where the same object defines data type on input and on output. The input flow for this process is "Room (room number)". Two output flows are here: "Free room" and "Reserved room". One data structure "Room" is defined for the discussed case (see Fig.8). The receiving of classes, attributes and operations is performed as it is described in section 3.
Applying graph theory [14] the nodes are transformed from arcs of the business process model. The arcs are transformed from the node of the business process diagram. There are three nodes and two arcs in intermediate model. Intermediate model and the received communication diagram are presented in Fig.8. Communication diagram contains three objects – "Room", "Free room" and "Reserved Room" and one operation "check_room_status", which is transformed from the process "Check room status".Two variants of the communication diagram are in Fig.8. The first variant is received directly from intermediate model "as is". However, as Fig.8 shows, interactions occur among instances of one object "Room". The communication diagram allows defining a class diagram and a kind of relations, i.e., "generalization", as bottom part of Fig.8 presents [6].

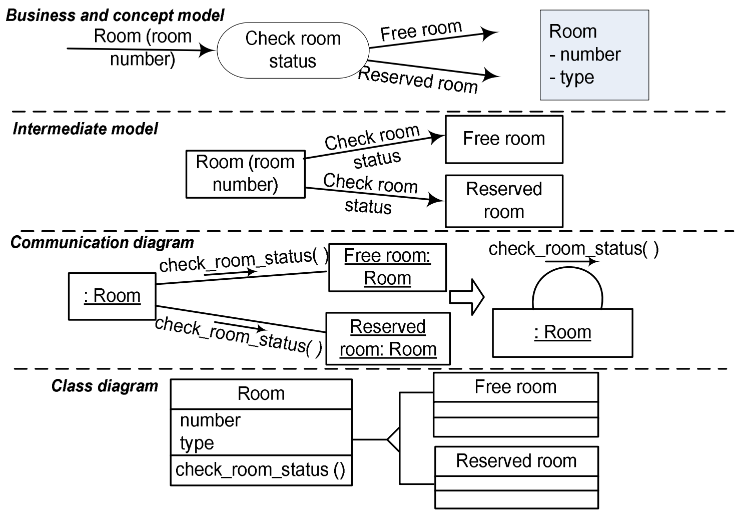

Fig. 8. The business process, data structure, intermediate, communication and class diagrams for transformation case 3

Initial model shows that one data flow “Room” is split into two data flows “Free Room” and “Reserved Room”. Therefore class “Room” is related with classes “Free Room” and “Reserved Room” with generalization, where “Room” is superclass, “Free Room” and “Reserved Room” are subclasses.

5. Conclusions

Business process and concept model is discussed as graphs with arcs and nodes for receiving of class diagram from initial knowledge. Analysis of graphs makes possible to apply graph transformations, where arcs may be transformed into nodes, and nodes may be transformed into arcs. Such analysis gives a possibility to discuss about sharing of responsibilities among classes, defining the relationships between them, defining of classes, its attributes and operations. Thus making use of the above discussed transformations it is possible to define such elements of class diagram as: classes, attributes, operations, dependency, aggregation and generalization. After performing all transformations from the initial model of business knowledge, the elements of the UML class diagram are received. The formally received class diagram is the base for further construction of the system architecture. The MDA Platform Specific Model can be obtained by adding platform specific properties to class diagram. The dynamic structure f the system, represented with state and activity diagrams could be received from two-hemisphere model, but it requires additional researches.
It is hopefully that the class diagram elements are received in the formal way, because currently developers construct class diagram manually without considering of user oriented models created in the initial project stages. A lot of organizations use 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. Therefore, the class diagram received in the formal way from the two-hemisphere model serve as a bridge between user oriented models and developer oriented models. This class diagram helps to implement software more close to initial description of system processes. The transformations discussed in the paper served as a basis for definition of formal algorithm [5], which gives a possibility to generate structure of class diagram from initial business information. The algorithm allows to develop a tool [5] for the generation of specification of class diagram and is applied for several problem domains, defined in the form of two-hemisphere model. These applications make the approbation of defined transformations and have proved the possibility to generate classes, their responsibilities and three types of relationships in the way described in the paper.

6. 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.” The research reflected in the paper partly 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. Developing in OMG’s Model Driven Architecture / OMG, 2001. – http://www.omg.org/mda/papers.htm. - 2001.
2. MDA Distilled, Principles of Model driven Architecture / Mellor S. J., Scott K., Uhl A., Weise D. – London: Addison-Wesley, 2004. – P.150.
3. Nikiforova, O., Kirikova M., Pavlova N. Principles of Model Driven Architecture in Knowledge Modeling for the Task of Study Program Evaluation // Databases and Information Systems IV, 2007. – Vilnius: IOS Press in the series "Frontiers in Artificial Intelligence and Applications", Vasilecas O., Eder J., Caplinskas A. (eds), P.291-304.
4. Nikiforova O., Kirikova M. Two-Hemisphere Model Driven Approach: Engineering Based Software Development // Proceeding Of The 16th International Conference Advanced Information Systems Engineering CAiSE’2004. – Berlin: Persson A., Stirna J. (Eds.), LNCS 3084, Springer – Verlag Berlin Heidelberg, 2004. – P. 219-233.
5. Nikiforova O., Pavlova N. 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 2008), International Workshop on Enterprise Information Systems (ENTISY 2008), October 26-31, Sliema, Malta, 2008, Mannaert H., Dini C., Ohta T., Pellerin R. (Eds.) – Published by IEEE Computer Society, Conference Proceedings Services (CPS), 2008. – pp. 105-112.
6. Nikiforova O., Pavlova N. Foundations on Generation of Relationships Between Classes Based on Initial Business Knowledge // Proceeding of the 17th International Conference on Information Systems Development (ISD2008), Information Systems Development: Towards a Service Provision Society, August 25-27. – Paphos: 2008. – (in press).
7. Nikiforova O., Pavlova N., Grigorjevs J. Several Facilities of Class Diagram Generation from Two-Hemisphere Model in the Framework of MDA // IEEE 23rd International Symposium on Computer and Information Sciences (ISCIS 2008), October 27-29. – Istanbul: 2008. pp. 1-6 – at IEEE Xplore: http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=4717848&isYear=2008&count=124&page=4&ResultStart=100
8. Chen P. The entity relationship model – towards a unified view of data // ACM Trans. Database Systems, 1976, P. 9-36.
9. Nikiforova O., Pavlova N. 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), October 13-15. – Brno: 2008. – pp. 133-146
10. GRADE Business Modeling, Language Guide. INFOLOGISTIK GmbH.- 1998 P.170.
11. Larman C. Applying UML And Patterns: An Introduction to Object-Oriented Analysis and Design.– New Jersey: Prentice Hall, 2000.
12. Kleppe A., Warmer J., Bast W. MDA Explained: The Model Driven Architecture. Practise and Promise, London: Addison Wesley, 2003. – P.192
13. Rumbaugh J., Jacobson, I., and Booch, G. The unified modeling language reference manual.– London: Addison-Wesley, 1999.
14. Grundspenkis J. Causal Domain Model Driven Knowledge Acquisition for Expert Diagnosis System Development // Kaunas. – Kaunas University of Technology Press, 1997.
15. Pavlova N. Approach for Development of Platform Independent Model in the Framework of Model Driven Architecture, Ph.D. thesis, Riga Technical University.– 2008.

16. Fowler M., Scott K. UML Distilled Applying The Standard Object Modeling Language.– London: Addison-Wesley, 1999.

Pavlova N., Ņikiforova O. UML klašu diagrammas elementu iegūšana no sākotnējas informācijas par problēmas vidi

Modeļu vadāma arhitektūra ir plaši pielietojama programmatūras izstrādē. Modeļu vadāmas arhitektūras pamata koncepcija ir programmatūras izstrādes sadalīšana atbilstoši abstrakcijas līmeņiem un attiecīgajiem modeļiem – no skaitļošanas neatkarīgs modelis (angl. – Computation Independent Model), platfromneatkarīgs modelis (angl. – Platform Independent Model), platformai specifisks modelis (angl. – Platform Specific Model) un programmatūras kods. Šis raksts apskata centrālas platformneatkarīga modeļa komponentes – sistēmas statiskas struktūras UML (angl. Unified Modeling Language) klašu diagrammas veidā –konstruēšanu. Eksistē daudz pieeju klašu diagrammas konstruēšanai, daži pētnieki mēģina atrast iespēju formāli iegūt klašu diagrammu. Tagad tikai daļa no klašu diagrammas elementiem var būt iegūta ar formāla algoritma palīdzību. Rakstā ir piedāvāta metode klašu diagrammas elementu iegūšanai no sākotnējas informācijas par problēmas vidi, kura ir attēlota savstarpēji saistīto biznesa procesu un konceptuālo modeļu veidā. Piedāvāta metode ļauj iegūt tādus klašu diagrammas elementus kā klases, to nosaukumi, atribūti, metodes, un attiecības starp klasēm. Šo elementu formāla iegūšana ļauj izvairīties no kļūdām sistēmas analīzes un projektēšanas gaitā. Šajos posmos ir ļoti svarīgi dabūt korektu sistēmas modeli, jo no projektējuma modeļa ir atkarīga programmatūras kvalitāte.

Pavlova N., Nikiforova O. Discussion of obtaining of elements of UML class diagram from initial information about problem domain

The Model Driven Architecture (MDA) is frequently used approach to software development. Basic conception of MDA is splitting of software development by levels of abstraction, where every level is presented with separate model – Computation Independent, Platform Independent, Platform Specific Models and code. The construction of central component of Platform Independent Model – system static structure in form of Unified Modeling Language (UML) class diagram is discussed in the paper. A set of approaches proposes the way for construction of UML class diagram. Different researchers try to find the possibility to obtain elements of class diagram in the formal way. This paper describes the method of obtaining of elements of class diagram from initial knowledge about the problem domain, which is described with interrelated business process and concept model. The proposed method defines main elements of class diagram, such as class name, class attribute, class operation, relationships among classes. Formal definition of these elements allows to avoid mistakes in the software analysis and design stage. It is very important to obtain correct model on this stage, as the design model is a base of future software.

Павлова Н., Никифорова О. Получение элементов классовой диаграммы языка УМЛ из начальной информации о проблемной области

Управляемая моделями архитектура (англ. Model Driven Architecture) широко применяется в разработке программного обеспечения. Основная концепция управляемой моделями архитектуры это разделение процесса разработки программного обеспечения на несколько уровней, которым соответствуют предлагаемые модели – независимая от вычислений, независимая от платформы, специфичная для платформы модели и прграммный код. Данная статья рассматривает конструирование центральной компоненты независимой от платформы модели – статической структуры системы в виде классовой диаграммы в нотации УМЛ(англ. – Unified Modeling Language). Разные исследователи пытаются найти способ формального получения элементов классовой диаграммы, но пока это удается только для части элементов. Эта статья описывает предлагаемый метод формального получения таких элементов классовой диаграммы как класс, его атрибуты, операции и отношения между классами из начальной информации о проблемной области, в видевзаимосвязанных модели бизнес процессов и концептуальной модели. Формальное получение этих элементов позволяет избежать ошибок на стадиях анализа и проектирования, что дает возможность разработать качественое программное обеспечение.