UDC 004.432
Original Scientific Paper

Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA)

Martin Kardoš *martin.kardos@fri.uniza.sk*
University of Zilina, Slovak Republic
Faculty of Management Science and Informatics

Matilda Drozdová *matilda.drozdova@fri.uniza.sk*
University of Zilina, Slovak Republic
Faculty of Management Science and Informatics

Abstract

Information system's models on higher level of abstraction have become a daily routine in many software companies. The concept of Model Driven Architecture (MDA) published by standardization body OMG[1] since 2001 has become a concept for creation of software applications and information systems. MDA specifies four levels of abstraction: top three levels are created as graphical models and the last one as implementation code model. Many research works of MDA are focusing on the lower levels and transformations between each other. The top level of abstraction, called Computation Independent Model (CIM) and its transformation to the lower level called Platform Independent Model (PIM) is not so extensive research topic. Considering to a great importance and usability of this level in practice of IS[2] development now our research activity is focused to this highest level of abstraction – CIM and its possible transformation to the lower PIM level. In this article we are presenting a possible solution of CIM modeling and its analytic method of transformation to PIM.
Keywords: transformation, MDA, CIM, PIM, UML, DFD.

1. Introduction to Model Driven Architecture - MDA

Software application development of IS is since the origin UML[3] [1] in nineties, connected with higher level of abstraction. Many software architects have realized that number of changes; and editing in development of a software application of IS has dropped when we first examine the application on higher level of abstraction. Later this fact has become a background for MDA creation [2].

MDA is a concept for software development of IS based on creation of models and transformations between them, defined by a standardization body in software engineering, OMG. OMG claims that MDA is an evolutionary step in the software development.

MDA has three main advantages against other methodologies of software development: transferability that is connected with platform independency, interoperability that is closely related to standard development and reusability that is the result of the previous two advantages.

[1] Object Management Group

[2] Information System

[3] Unified Modeling Language

2. Levels of MDA

As it has been mentioned in the previous chapter, MDA recommends certain models – levels of abstraction that can be used in the process of IS development. We assume that the expression "model" should define certain level of abstraction and since it can be defined by number of models in various forms (UML, BPMN[4], DFD[5], ...) we decided to use MDA term "level" that better expresses different level of abstraction. OMG describes different levels and their relations but it does not specify how to create these abstract levels and which exact models and notations to use for their representation and how to transform them with one another. There are some recommendations from various researchers in the field that can be similar in certain points and different in others.

Basic levels of abstraction are called: Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM) and Implementation model (IM) - Code. In the process of IS development MDA models are created according to the order depicted in figure 1.

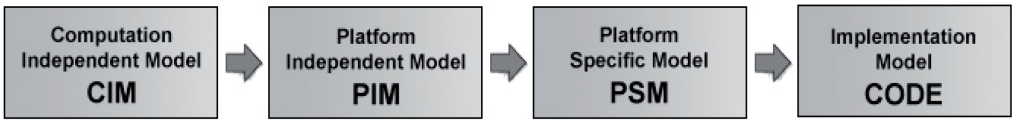

Figure 1. Basic levels of abstraction in MDA [3].

2.1. Computation Independent Model (CIM)

CIM is the level that does not display details of IS construction but it specifies activities that are being processed in the IS. In other words this level represents business processes of the organization for which the IS will be developed. CIM is meant to the analysts on the top level that can be a business analysts, domain experts or domain users of the system. CIM is sometimes called domain model. CIM does not have an information about models or artefacts that will be used for IS implementation. It describes the environment in which the IS operates and it helps to recognize what do we expect from the IS. It is useful not only as an aid to understand the problems of the IS design but as well as a "vocabulary" for usage of this system in other systems. CIM plays an important role in passing the gap among specialists for domain (business and domain experts) and specialists for design and development of IS (software analysts). In MDA specification the requirements on CIM should have relations to PIM and PSM construction and vice versa.

2.2. Platform Independent Model (PIM)

PIM level shows certain level of independence in such a way that models that are represented there (mostly UML models), are suitably chosen for use in various platforms. PIM describes IS, but hides details in usage of concrete technology. PIM creates specification for required services of the information system without technical platform dependent details. Software analysts are involved in creation of this level of MDA. When considering PIM models, often the question is raised if the model is platform independent or not. This question has a simple answer: "depends on point of view". As it is mentioned in [4], the same model can be dependent on concrete platform like middleware CORBA but platform independent in a way that can be used with various operation systems.

[4] Business Process Modeling Notation

[5] Data Flow Diagram

2.3. Platform Specific Model (PSM)

PSM is a model in a relation to IS from the view of a certain platform. PSM connects specification from PIM with details that specify what type of platform will be used by IS. PSM can offer more or less detailed technical specifications depending on its purpose. PSM will be realized by transformation into an implementation model - code that defines all information for IS creation and its launch. One PSM can appear as PIM for another PSM levels that can be further transformed into the code [3, 4]. This level is created by software developers. The basic idea of the PSM is to realize transformation from PIM to PSM models [3].

3. General transformations of models in MDA

The foundation of MDA architecture is creation of models. However, there is an important issue – transformation among these models. The concept of OMG says that in the MDA process the main idea is to transform higher levels (CIM, PIM) into lower levels (PSM) that are used to create implementation code. Transformation of a model is a process when one model is a source, converted into another model – destination with the use of certain transformation rules. There are various methods used as transformation rules. This process is being displayed on fig. 2.

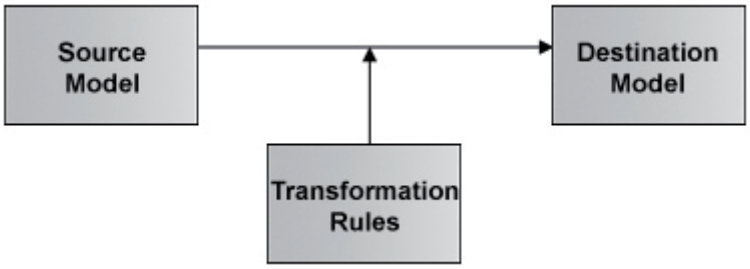

Figure 2. General MDA transformation process.

Transformation rules describe how source model elements should be transformed to destination model elements. Transformation rule consists of left and right side. Left side accesses source model and adds additional information for its creation and right side extends it to the destination model. Both sides can be described by variables, templates, logical rules etc.
Since till now, most of IS designs are based on the PIM level of MDA, the biggest emphasize is put on PIM to PSM transformation in both ways. There are various CASE tools presented as MDA tools that automate this transformation in a great manner. CIM to PIM transformation is not mentioned often by OMG and as this can be of a great help for business analysts and domain experts we have decided to focus on this higher level of MDA.

4. The CIM to PIM transformation in MDA

The problem of transformation from CIM to PIM expressed in MDA is becoming actual in relation to effective use of IS when implementing innovations in enterprises and institutions. Transformation of CIM to PIM cannot be considered as a technological translation from one model to another. The description of an organisation is the foundation for the success of the final solution. Therefore it is necessary to remind the reader that before the CIM to PIM transformation we have to actually transform the organization processes based on the analyses of the institution into the CIM model according to the principles of the innovation management. The solution of the transformation of models is possible to be done after creation of a process map of new and innovated processes.

Many software architects understand that CIM level and its following transformation to PIM is the first step to quality design of complex IS in relation to the environment where they will be used. On the contrary, the state of solution of this problem is currently much underrated and most of the research concentrates on PIM to PSM transformation and PSM to IM transformation. When solving the CIM to PIM transformation it is necessary to understand that activities and processes on CIM level represent business reality and further levels are necessary for the IS development. CIM analytic model describes all the activities, manual and half automatic therefore it is necessary to do reengineering or redesign of existing processes. This change needs creation of „soft" processes, for implementation of new policies and new enterprise culture.

Creation of CIM level is not unified now and does not use unified standard but it is assumed that this level is represented by the model of business processes [9], [10]. It is a fact on which we will build our approach.
According to [9] the transformation CIM to PIM is presented like disciplined approach. It uses UML2 activity diagrams which model the business processes. It is modelled like the user's tasks. From detailed activity diagrams, system requirements are specified. From the model of requirement elements the system components are created. Finally, a set of business archetypes helps to transform the system components to the PIM layer in details. In [10] there is presented an approach in which CIM level is represented by business processes in BPMN notation. Various UML Use Cases which present some part of IS are obtained from business processes using by QVT[6] [13] rules. In [11], approach is represented where the features and components are adopted as the key elements of CIM and PIM building and responsibilities as the connectors between features and components to facilitate CIM to PIM transformation.
In the next chapters we are introducing some models (diagrams) representing CIM and PIM levels and further introduce analytic process of transformation of CIM level to PIM level in MDA.

5. Models using our CIM to PIM transformation in MDA

Our approach is not fully automatic way of transformation CIM to PIM but it provides one way to transform business requirements to UML models. At first we had to determine which models represent CIM level and which models represent PIM level of MDA. In next chapters we are introducing those models.

5.1. Models representing our CIM level

5.1.1. Data Flow Diagram (DFD)

As we mentioned earlier CIM level is mostly represented by the model of business processes. For modelling business processes many notations are used. For our analytical solution of transformation we have picked out the first – DFD [5], [8] diagram for the following reasons. The first one is the easy understandability for all the present parties (both IT and business), the second one is their easy creation and the third one is their wide usage. In general, DFD can be defined as one of the older techniques for business process modeling.
DFD is a modelling technique used for software engineering that graphically displays data flows from external entities into the IS and vice versa. DFD displays data that pass from one process to another. Data flows diagrams are used as graphical notation for process expression of data processing. In the practice, software designer often first draws the context level of DFD that represents interaction among the system and the surrounding elements. This level of DFD is decomposed into lower levels that model parts of the system in a greater detail. The

[6] Query View Transformation

notation BPMN is explained in the [12] .Transformation of CIM to PIM using this notation will be our further research priority.

5.2. Models representing our PIM level

Models representing PIM level are mostly represented as UML language models. We work with 4 types of UML in current analytic transformation process on this level: **use case diagrams, activity diagrams, sequence diagrams and domain diagrams (models).**

5.2.1. UML use case diagram

Use case diagrams capture functionality that will be covered by the future IS that means they describe the exact functions that will be performed by IS. The implemented IS will not contain anything else but what is described in the use cases. Each use case describes one of the ways of usage of the system from the user's point of view thus it describes its needed functionality.

5.2.2. UML activity diagram

Activity diagrams are often represented models of business processes but mostly they are used for representation of operations of the system. In UML 1.x the activity diagram is a variation of the state diagram where states represent operations and changes represent activities that happen after the operation. In UML 2.0 the activity diagram resembles UML 1.x diagram and its semantics is based on Petris nets. Even though activity diagram can model inner logics of a complex operation, it is much more preferable to decompose it to more simple operations. In many regards, activity diagrams of UML 2.0 version are object oriented equivalents of DFD diagrams. More information about UML activity diagram is in [6].

5.2.3. UML sequence diagram

This type of diagram displays various activities that are processed in sequence. We are talking about data flows in the processes of information system that are exchanged among processes that implement the behaviours of this system, ordered in time. They display flows of direction among objects.

5.2.4. UML domain diagram

According to [7] domain diagram is an alternative to the conceptual model DFD. The difference is that the domain diagram does not model the problem from the whole complexity point of view but it focuses on a certain area – domain that is the subject of our interest.
It introduces relations among objects that are performed in the IS where it defines single concepts and their meaning.

6. Our analytical CIM to PIM transformation

Creation of business process maps of the organization is currently done in various ways. Apart from traditional notations there are new ones now and many analytic companies have own "know how" for business process map drawing. As we mentioned in previous chapters one of the widely known ones is DFD notation – Data Flow Diagrams. It is suitable for basic description of a system of an organization because it can effectively describe 4 types of diagrams: Current physical, current logical, new physical and new logical. From the well described DFD we can start when creating other types of diagrams necessary for design of an IS and its description. According to traditional use of DFD and its simplicity, the first

reflection about CIM to PIM transformation uses DFD diagrams which represent business processes as models for CIM level description. For the PIM level we consider UML diagrams. When changing CIM to PIM, it means going from the level of business analysis to the partial IS design, so we start from DFD and then we will define basic rules for some UML models creation.
As an example of the DFD representation we use the business process on figure 3 which represents one subprocess of main education process. It is the process of creation and saving an e-Content. This subprocess is one of those which supports electronic form of education.
According to DFD representation of this business process we develop a part of IS called E-Repository which will be used as repository for E-Content.

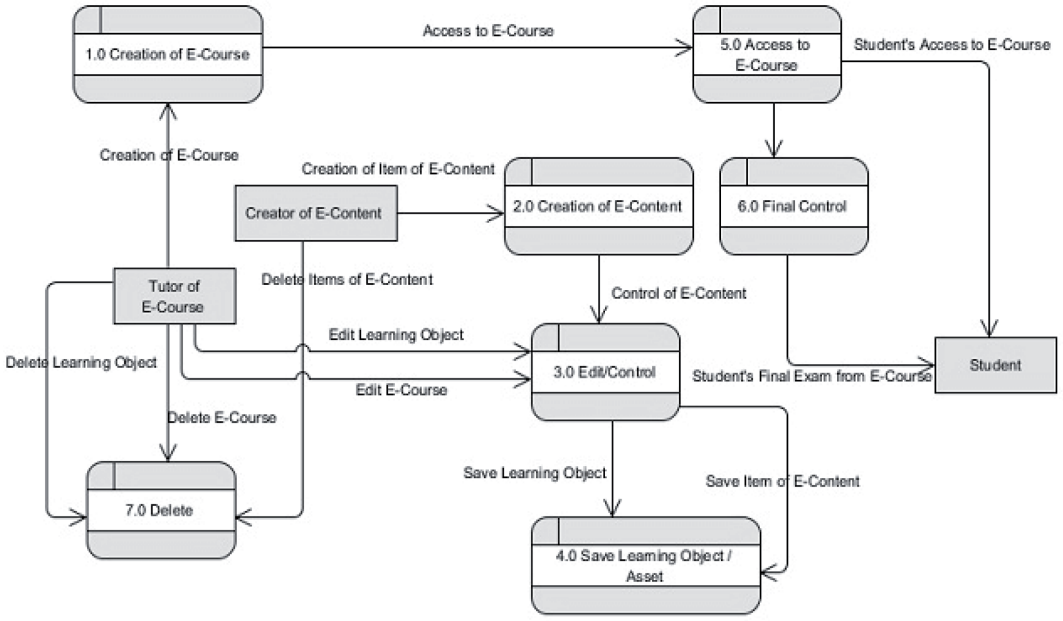

Figure 3. Example of business process of creation and saving an e-Content illustrated in DFD.

6.1. Use case model creation

From the DFD we can specify external entities that will represent actors in the use case diagrams. These processes on the first level of DFD, operated by the external entity, we will transform into functionalities in the use case diagram. By further analysis of DFD diagrams from the first level diagram to the lowest levels we are choosing the processes that are external entities responsible for. Then they create associations among external entities and processes. Associations are created based on responsibilities of external entities for the activities in the DFD diagrams. Among the use cases there were included and excluded relations that define dependency of the processes. Associations include/exclude that are represented in the Use Case have to be already clear from the process description. figure 4 illustrates process of transformation DFD (from figure 3) to UML use case diagram.

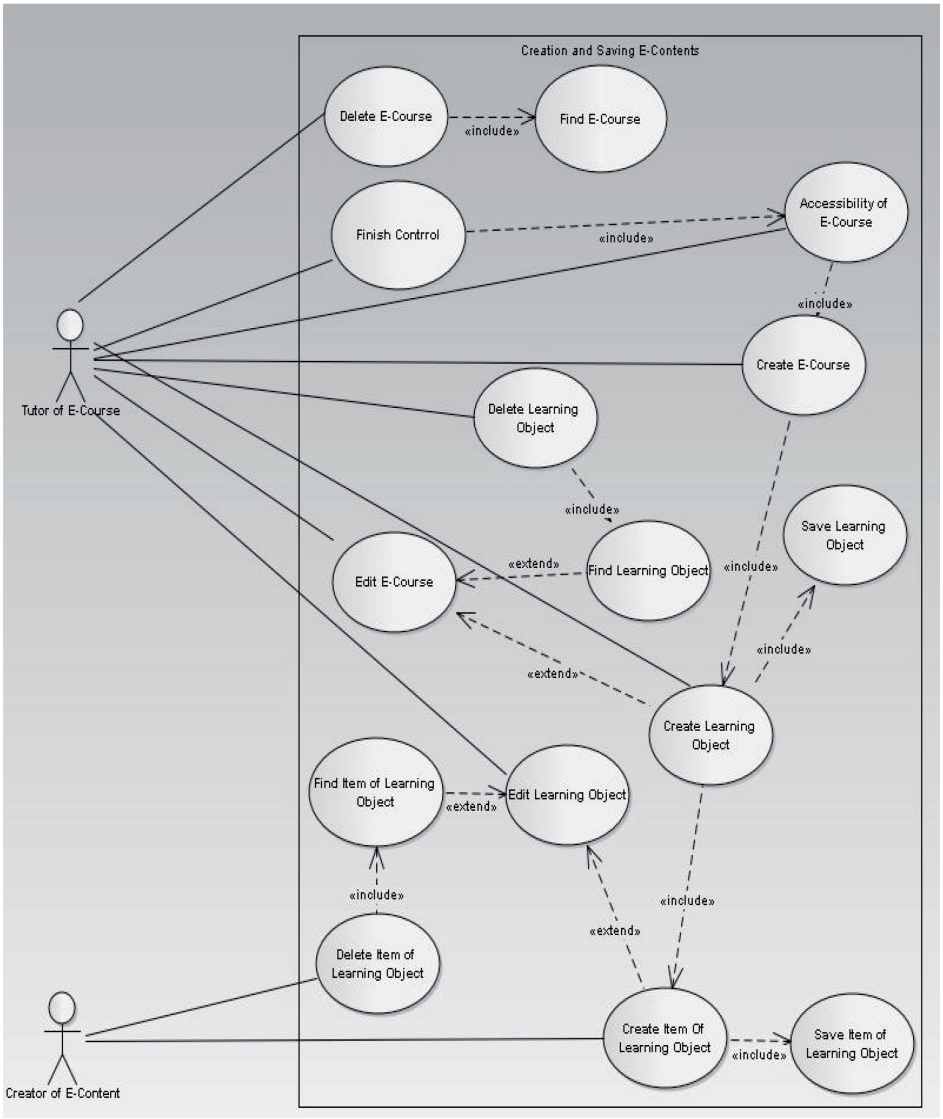

Figure 4. Example of transformation DFD diagram to UML use case diagram.

In Use Case diagram the external entity "Student" from DFD diagram (figure 3) is not illustrated because this entity will use E-Content by means of LMS system. DFD diagram describes reality, every data flow and every entity which exists in organization. But in use case diagram we have the functionalities which are necessary for the concrete use case of information system.

6.2. Creation of the other type of diagrams

Except Use Case diagram we use also other types of diagrams. We mentioned those diagrams in section 4.2. For simplicity, from DFD diagram on figure 3 we demonstrate the transformation only for subprocess *6.0 Final Control*. This subprocess presents a creation of exam test like the final verification of student's knowledge from particular e-course. The decomposed subprocess is illustrated on figure 5.

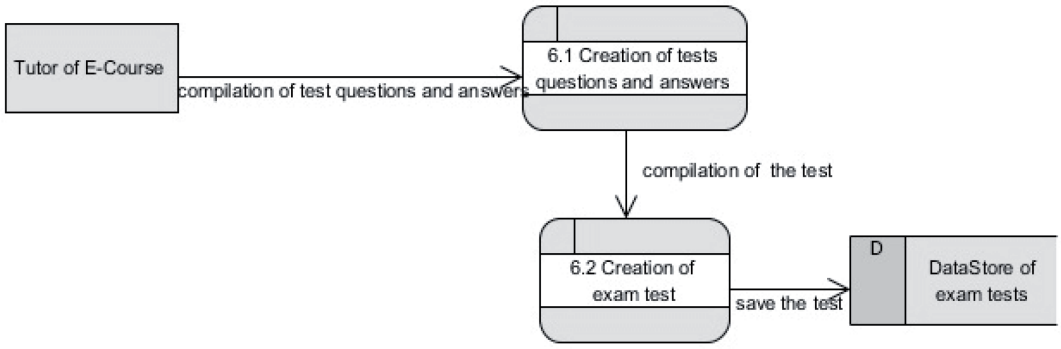

Figure 5. Decomposition of DFD subprocess *6.0 Final Control*.

6.2.1. Creation of the activity diagram

Another UML diagram is the activity diagram. When creating these diagrams we start often from DFD of the lower levels of processes. For each subprocess on lower level of DFD we have to find out its inputs and outputs and what external entities cooperate on it. Based on the external entities from DFD, the swimlanes are created and they define responsibility for the activities in the process. The specification of the inputs and subprocesses from the DFD diagram help us to define logics to each subprocess in the manner of decisions and by adding further necessary activities that define the process in more detail.

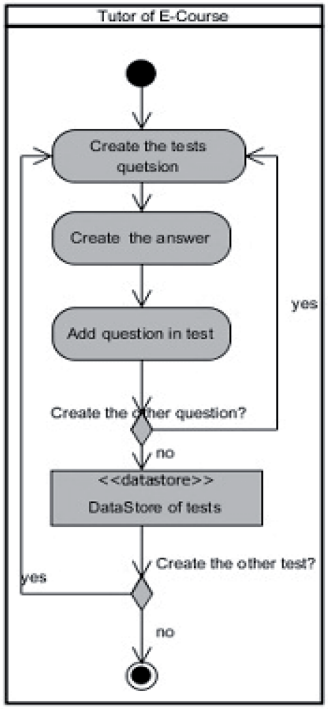

Figure 6. Example of transformation DFD to the UML activity diagram.

6.2.2. Creation of the sequence diagrams

Sequence diagram is used to display relations among objects in the order in which these relations are appearing. Apart from documenting of organizational items this diagram can be used as the necessary document of communicational requirements for further implementation of the IS. They act as the description of use case diagrams supplement.

When designing sequence diagrams it is necessary to find out which external entities enter the subprocesses. Further it is necessary to think about entities of systems that cooperate with external entities. Mostly it is the IS itself or another software application and its database.

Further the sequence of execution of the activities is being drawn from the subprocesses into the sequence diagrams.

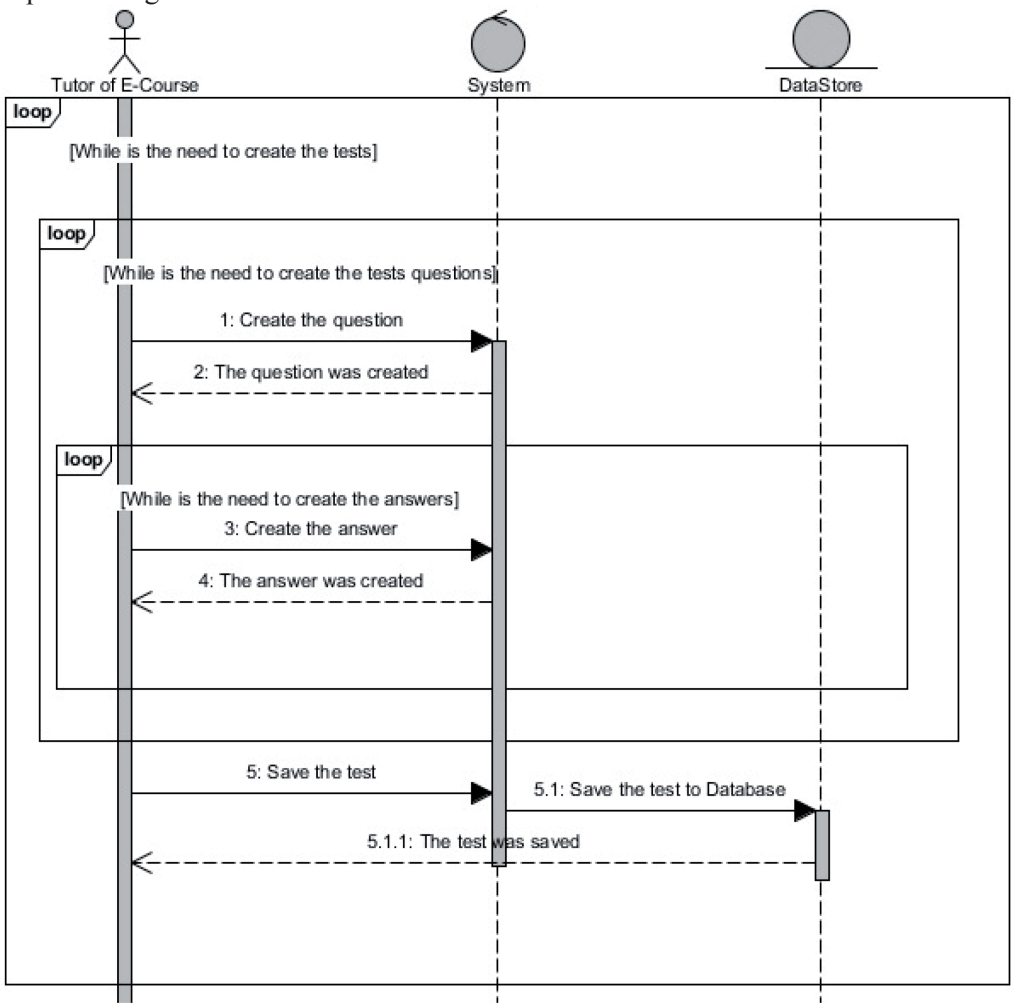

Figure 7. Example of transformation DFD to the UML sequence diagram.

6.2.3. Creation of the domain model

The domain model reduces the difference between the requirement analysis and creation of the design specifications. It represents general understanding of the key concept in the organization. It is created for better understanding of the class diagram and is not aimed only to IT specialists. It displays objects that will be in the program structure and the relations among them. From the domain diagram we can create the class and database diagram. When creating the domain model we need to display all the external entities and elements (system, database…) that they cooperated with from the DFD diagram. From the DFD diagrams they are projected into the domain model. Finally it is necessary to reflect on the relations among various activities of the elements that are not derived from the DFD or the processes. These are the 1 : 1, 1 : n and m : n relations.

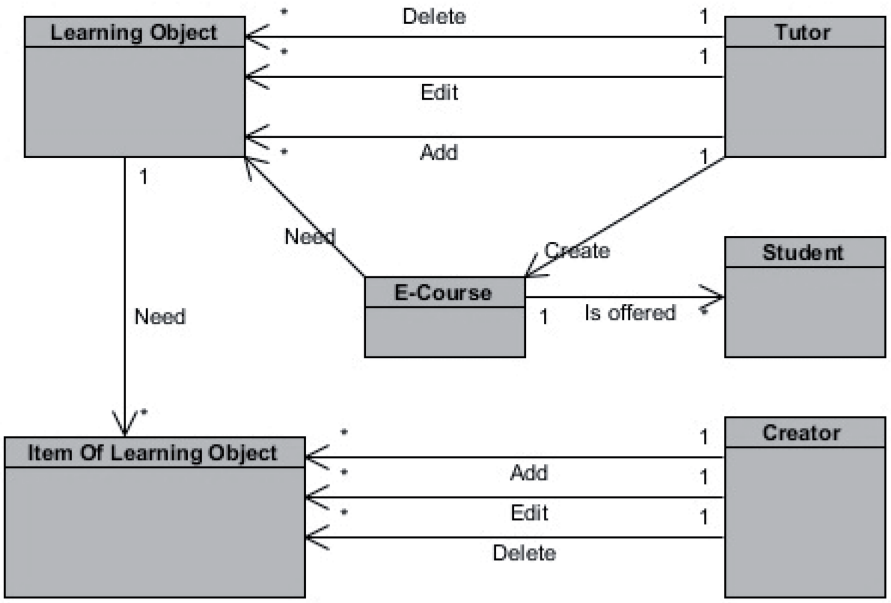

Figure 8. Example of transformation DFD to the domain model.

7. Conclusion

Analytic method of transformation among CIM and PIM level of MDA is described in his contribution. Same models that represent these two levels are introduced. In case of CIM level business processes models are represented by DFD and the textual description of processes while in case of PIM level UML models, especially use cases, activity diagrams, sequence diagrams and domain diagrams are utilized.

Using these models, a transformation business processes represented in DFD into UML diagrams is discussed. This way of transformation should be a good foundation for better and more detailed design of IS and software applications as well as it opens a scientific discussion of this problem.

Electronic support for the business process of creation and saving an e-Content as one example of usage of this transformation is depicted. This electronic support is one of the e-learning tools described in [Grondzak].

Our future research will be oriented on automatic way of transformation CIM into PIM using BPMN notation to business processes modelling at CIM level.

Acknowledgements

This work has been supported by the grant KEGA No. 3/5205/07 entitled "Digital Library for Metadata Objects and its Management".

References

[1] Arlow, J; Neustadt, I. UML2 and the Unified Process: *Practical Object-oriented Analysis and Design*, Addison Wesley, 2005

[2] OMG Inc. Model Driven Architecture (MDA) FAQ.... http://www.omg.org/mda/faq_mda.htm, downloaded: October, 29th 2009.

[3] Miller, J; Mukerji, J. MDA Guide Version 1.0.1, OMG Inc., 2003. http://www.omg.org/docs/omg/03-06-01.pdf, downloaded: October, 29th 2009.

[4] Bizoňová, Z; Ranc, D. MDA used at analysis of LMS systems and creation of general model. In *Objekty 2008: 13th year of conference*, Žilina, November, 20.-21st, 2008: Publisher: Edis, University of Žilina, pages 162-170, ISBN 978-80-8070-927-3, 2008.

[5] Bieliková, M. *Softvérové inžinierstvo – Princípy a manažment*, skriptá, Publisher: STU, Bratislava, ISBN 80-227-1322-8, 2000.

[6] Visual Paradigm. UML 2 Diagrams, Activity Diagram. http://www.visual-paradigm.com/VPGallery/diagrams/Activity.html, downloaded: October 29th, 2009.

[7] Molhanec, M. Vývoj řízený modelem a tradiční modely softwarového inženýrství. In: *Objekty 2008: 13th year of conference,* Žilina, ISBN: 978-80-8070-927-3, 2008.

[8] Hoffer, J.A; George, J.F; Valacich, J.S. *Modern system analysis and design*. Prentice Hall ISBN 0-13-145461-7, 2004.

[9] Kherraf, S; Lefebvre, E; Suryn, W. Transformation from CIM to PIM Using Patterns and Archetypes, 19th Australian Conference on Software Engineering, 2008, http://www2.computer.org/portal/web/csdl/doi/10.1109/ASWEC.2008.63, downloaded: February, 27th 2010.

[10] Rodríguez, A; Fernández-Medina, E; Piattini, M. Towards CIM to PIM Transformation: From Secure Business Processes Defined in BPMN to Use-Cases. In: *Lecture Notes in Computer Science*, Publisher: Springer Berlin / Heidelberg, ISBN 978-3-540-75182-3, pages 408-415, 2007.

[11] Zhang, W; Mei, H; Zhao, H; Yang, J. *Transformation from CIM to PIM: A Feature-Oriented Component-Based Approach*. Work presented at MoDELS 2005, Montego Bay, Jamaica, 2005.

[12] White A.S. *BPMN Fundamentals*, 2005, http://www.omg.org/docs/pm/05-12-06.ppt, downloaded: February, 27th 2010.

[13] QVT: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, 2008, http://www.omg.org/spec/QVT/1.0/PDF/, downloaded: February, 27th 2010.

[14] Grondzak, K. E-learning tools. In: *RTT 2007 - Research in telecommunication technology: 8th international conference*, Žilina - Liptovský Ján, September 10-12, 2007, Slovak republic. - Žilina: 2007, University of Žilina, pages 114-117, ISBN 978-80-8070-735-4, 2007.