http://dx.doi.org/10.5755/j01.eee.19.2.1717 ELEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392-1215, VOL. 19, NO. 2, 2013

Enterprise Knowledge-based Generation of Class Model

A. Lopata[1,2], M. Ambraziunas[1], S. Gudas[1], R. Butleris[2], R. Butkiene[2]
[1]Kaunas Faculty of Humanities, Vilnius University,
Muitines St. 8, LT-44280 Kaunas, Lithuania
[2] Centre of Information Systems Design Technologies, Faculty of Informatics,
Kaunas University of Technology,
Studentu St. 50–313a, LT-51368 Kaunas, Lithuania
audrius.lopata@khf.vu.lt

***Abstract*—The article presents Knowledge- Based MDA Class Model (UML notation) generation technique. Knowledge-Based MDA approach is a progressive Information Systems development method which combines the main components of traditional MDA and the best practices of Knowledge-Based Information Systems Engineering. The main advantage of such method is that problem domain knowledge acquainted at user requirements acquisition and business modelling stages are validated against formal criteria, in such way reducing empirical factors that could negatively impact the whole IS development process. The starting point of Class model generation process is a selection of particular entity from the Enterprise Model as the main class. System Analyst is able to select one of four elements types for class model generation process: Actor, Process, Information Activity and Flow. The main steps of Class model generation from Enterprise Model are presented in this article as well as Class model generation example. The final result of the generation process is the particular Class model that specifies static structure of the particular problem domain and can be used as business entities framework for developers of Information Systems.**

***Index Terms*—Class model, enterprise model, knowledge based, model driven architecture, information system engineering**

I. INTRODUCTION

Currently Model Driven Architecture (MDA) is one of the most widely used Information Systems (IS) development methodologies, which focuses on functional requirements acquisition and system. MDA is a theoretical framework which specifies guidelines of CIM, PIM, PSM models creation and their reciprocity only. The presented Knowledge-Based MDA approach combines the main components of traditional MDA and the best practices of Knowledge-Based Information Systems Engineering (ISE) [1]–[9].The main advantage of proposed method is that problem domain knowledge acquainted at user requirements acquisition and business modeling stages are validated against formal criteria, in such way reducing empirical factors that could negatively impact the whole IS development process. The main component of Knowledge-Based MDA is Knowledge-Based Subsystem which consists of Enterprise-Model (EM) and Enterprise Meta-Model (EMM) and serves as the main repository of problem domain knowledge necessary for development of particular IS [1].

Manuscript received May 18, 2012; accepted October 12, 2012.

The main components of user requirements and design stage models can be generated in semi-automatic way from this repository, thus implementation of such functionalities in CASE tool will increase the productivity of the following participants of IS development process: System Analyst, System Designer and Developer.

There are numerous tools that partly or fully implement MDA framework (e.g. “AndroMDA”, “Enterprise Architect”, “Acceleo”, “MagicDraw”). Most of these tools implement generation and composition functionalities and focus on PIM/PSM/Code models of MDA process where is the lack of tools that analyze CIM models and validate them against user requirements and business modeling standards. MDA tools can be classified by the functions they can perform. These functions can be as follows: testing, creation, analysis, transformation, composition, simulation, reverse engineering and Metadata management. Most of the tools have graphical model creation environment.

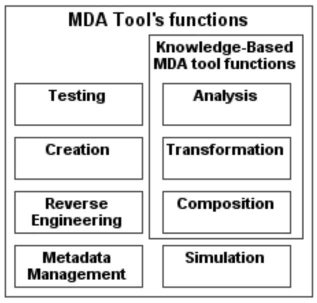


Fig. 1. The Knowledge-Based MDA tool functions.

Knowledge-Based MDA tool discussed in this article implements analysis, composition and transformation functions, which are necessary to generate user requirements specification and design stage models. Knowledge-Based MDA tool’s analysis process is performed using Enterprise Model based on Enterprise Meta-Model structure in order to validate user requirements according to particular problem domain knowledge. Composition functionality describes how particular UML models from CIM or PIM layer are transformed to particular EM model. Transformation is performed from EM to PIM internal models. One of these models is UML Class model. This particular transformation will be presented bellow in the article.

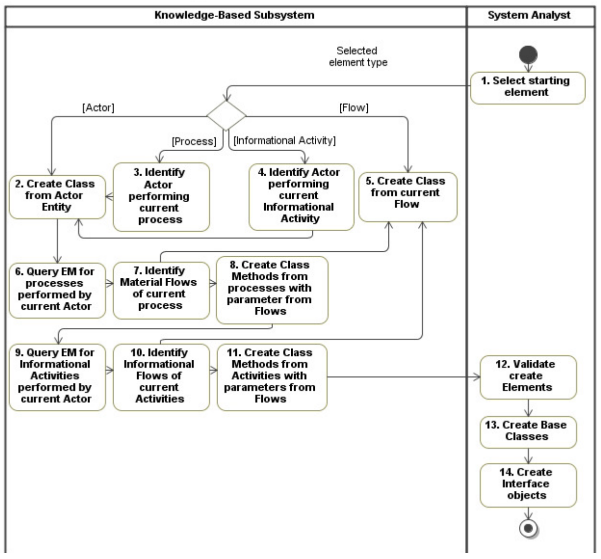
II. Related Works

Paper authors have performed wide publications review and came up with conclusions that there are two main trends for internal UML models transformations. There are authors who are working on various UML models transformations [4], [7], but these transformations are based on two models e.g. Use Case (UC) model to Class model [7], UC model to State model [4]. These transformations lack of integrity because the information captured in single model (e.g. UC) may not contain information of different view (e.g. UC model lacks information about system's structure). This issue can be solved using addition models or knowledge repository (Enterprise Model). There are several authors who are working on this approach. Most of these authors are working on particular UML diagrams (Class diagram [2], Use Case [3], modified workflows diagram), not on set of diagrams as it is necessary for full MDA process. The above mentioned authors use the Enterprise Meta-Model [3], [4], [9] created by joined scientific group of Vilnius University and Kaunas University of Technology. Particular Enterprise Meta-Model is based on the fundamentals of Control Theory [5]. There exist many different enterprise modeling standards as well, including: CEN EN 12204 [6], UEML [8], 40003 (CIMOSA) [10] etc. These standards have the following shortcomings when are used for user requirements acquisition, analysis and specification steps of Information Systems Development Life Cycle (ISDLC): the content of enterprise models is not verified according to formalized criteria; user requirements specification processes are performed in empirical way.

III. Class Model Generation from Enterprise Model

UML Class model is widely used by system analysts and architects for IS structure representation. This model is created after initial user requirements acquisition (represented as UC diagrams or as text document) process is completed. There are methods that can perform automatic transition from UC model to Class model [7], but in this particular case the Class model is created only from single initial model (UC model => Class model). KB-MDA is suggesting different approach: {UC model, BD model, Requirements model, Activity model} => Enterprise model (internal validation of knowledge) => Class model. As can be seen from provided example the main difference is that Class model is generated from EM, which contains knowledge from set of different UML models not from one particular as well as knowledge that is stored in EM is validated against formal criteria [5].
The starting point of Class model generation process is a selection of particular entity from Enterprise Model as the main class. System Analyst can select one of four element types for class model generation: Actor, Process, Informational Activity, and Flow (material or information). Depending on the selected element Process is performed in the following manner:

1) *Actor* – as the initial element. Class and class attributes are created from Actor entity. Methods are created from Actors Processes and Informational Activities. Enterprise Model is queried for Actor related processes and Informational Activities. If the processes or activities are found, they are converted to Class's methods. Input and output parameters are defined by particular process or activity' flow. In process case - Material Flows and in Information Activity case - Informational Flows are used. Flows are created as separate classes, in case they represent complex structures.
2) *Process* – as the initial element. Based on the selected process Actor's search is performed. After Actor is found, next step No.2. "Create Class from Actor entity" is implemented. From this step Class model generation process proceeds in similar case if Actor would be selected as the initial element.
3) *Informational Activity* – as the initial element. Based on the selected Activity search is performed in order to find Actor who performs this particular Activity. In case Actor is identified, the class objects generation process proceeds to second step No.2 "Create Class from Actor entity".


Knowledge-Based Subsystem
System Analyst
Selected element type
1. Select starting element
[Actor]
[Flow]
[Process]
[Informational Activity]
2. Create Class from Actor Entity
3. Identify Actor performing current process
4. Identify Actor performing current Informational Activity
5. Create Class from current Flow
6. Query EM for processes performed by current Actor
7. Identify Material Flows of current process
8. Create Class Methods from processes with parameter from Flows
9. Query EM for Informational Activities performed by current Actor
10. Identify Informational Flows of current Activities
11. Create Class Methods from Activities with parameters from Flows
12. Validate create Elements
13. Create Base Classes
14. Create Interface objects
Fig. 2. The main steps of Class model generation from EM.

4) *Flow* as the initial element. In case the flow represents complex structure, Class object is created to represent this structure. The attributes are taken from Enterprise Model while they are used as input or output parameters for created class methods. The next step is to identify the process or the activity that uses the particular flow No.4 "Identify Process or Activity". When the process or the activity is identified, Actor's identification procedure is performed as No. 3 "Identify Actor performing this Process" or No. 4. "Identify Actor performing this Activity".

After the basic elements are created automatically, System Analyst validates the created objects and appends model with Abstract/Virtual classes or Interfaces in case it is necessary.

IV. Class Object Generation from Actor Entity

The detailed activity diagram describing how from Actor entity Class object is created, will be presented in the picture below In the chapter above it has been stated that Actor is one of the four entities that can be used for Class model generation.

The Class creation activity diagram can be divided into three sub-activities: create empty class, create class attributes, and create class methods. Class attributes are created from attributes records in Actor's (in EM database). Each attribute can be created as separate class (complex type) or as basic type (simple type). Sub-Activity Create class method analyses Actor's Processes and Informational Activities, and creates for these objects separate methods. The created methods can have input and output parameters. The parameters are defined by input and output Flows of particular Process or Informational Activity. In most cases Flows are represented as Class objects as well because of a complex internal structure. In the Fig. 3 is depicted class entity generation algorithm from Actor entity.

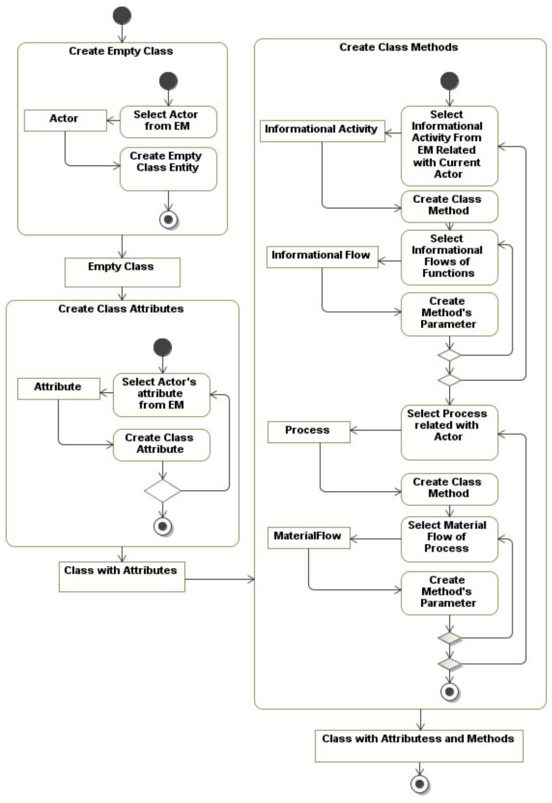

Create Empty Class
Actor
Select Actor from EM
Create Empty Class Entity
Empty Class
Create Class Attributes
Attribute
Select Actor's attribute from EM
Create Class Attribute
Class with Attributes
Create Class Methods
Informational Activity
Select Informational Activity From EM Related with Current Actor
Create Class Method
Informational Flow
Select Informational Flows of Functions
Create Method's Parameter
Process
Select Process related with Actor
Create Class Method
MaterialFlow
Select Material Flow of Process
Create Method's Parameter
Class with Attributess and Methods

Fig. 3. The Class generation starting from Actor Entity.

All steps and step results are as follows: 1. Select Actor from EM (Actor); 2. Create empty class entity (Class). 3. Select Actors Attribute (Attribute); 4. Created Attribute in Class Entity (Class with Attributes); 5. Select Informational Activity (Informational Activity); 6. Create Class Method (Class with Methods); 7. Select Informational Activity flows (Information Flow); 8. Create Method's parameters (Class with Methods and Methods parameters); 9. Select process (Process); 10. Create Class Method (Class with Methods); 11. Select process material flows (Material Flow); 12. Create Method's parameters (Class with Methods and Methods parameters). The final result of all process is Class with attributes and methods with input and output parameters.

V. CLASS MODEL GENERATION FROM ENTERPRISE MODEL: EXAMPLE

An example of Class model generation from particular Enterprise Model elements is presented in the pictures and the table below. For representation reasons EM entities and their interactions are presented in Work Flow diagram. This example is a part of larger scale Work Flow diagram taken from the particular practical work. In particular case there are actors of three types: Warehouse, Logistic Department, and Customer. Input Flow for Warehouse (not displayed) is GoodsList purchased by Customer. Using this list Warehouse performs an activity named "Package Goods and Prepare Invoice". Goods packaging is a material process and its output is packaged Goods (M). Invoice preparation is Informational Activity and its output is prepared Invoice (I).

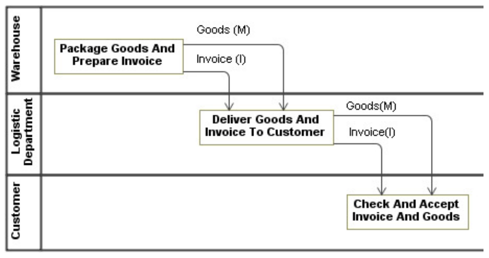

Warehouse
Package Goods And Prepare Invoice
Goods (M)
Invoice (I)
Logistic Department
Deliver Goods And Invoice To Customer
Goods(M)
Invoice(I)
Customer
Check And Accept Invoice And Goods

Fig. 4. The fragment of Work Flow diagram.

TABLE I. THE MAIN COMPONENTS OF WORK FLOW DIAGRAM FRAGMENT PRESENTED IN FIG. 4.

Actor	Activity	Material Flow		Informational Flow	
		Input	Outpu t	Input	Output
Ware.	Package Goods And Prepare Invoice		Goods	Goods List	Invoice
Log. Dep.	Deliver Goods And Invoice To Customer	Goods	Goods	Invoice	Invoice
Cust.	Check And Accept Invoice And Goods	Goods		Invoice	

The next activity is "Deliver Goods and Invoice to Customer". This is performed by Logistics Department. Input Flows are Goods (M) and Invoice (I). Logistics Department delivers these objects to Customer (Goods(M) delivery in particular case is material Process and Invoice(I) delivery is Information Activity), who is responsible for accepting or rejecting Goods(M) and Invoice(I). Using Class model generation process described in chapters above, class model fragment could be generated from these data in the following way: Actors are used as a starting point. Initial classes are as follows: Warehouse, Logistics Department, and Customer. The next step is class's attributes creation, but in particular case no Actors attributes are available, so no class attributes will be created. Class methods are created based on Informational Activities and material Processes. In current Work Flow diagram each single activity implements two internal activities: one involving material Process and one Informational Activity. For each internal Process and Informational Activity private methods are created. Public method representing the whole activity is created as well. As for the methods parameters, they are three (based on Flows): Goods List, Invoice, and Goods. These parameters are represented as classes and are used as input and output parameters for methods.

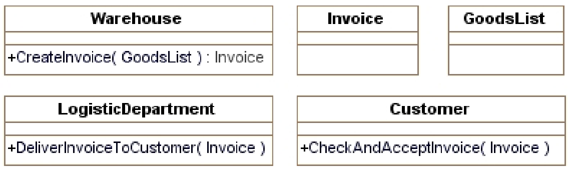

Warehouse
+CreateInvoice(GoodsList) : Invoice
Invoice
GoodsList
LogisticDepartment
+DeliverInvoiceToCustomer(Invoice)
Customer
+CheckAndAcceptInvoice(Invoice)

Fig. 5. Created class objects.

The provided example briefly describes the basic idea of creating UML Class model from Enterprise Model. The created Class model can be used as stand-alone static structure framework for developers or as part of PIM model

VI. CONCLUSIONS

This article presents Knowledge-Based UML Class Model generation algorithm. UML Class model generated using this algorithm is validated against formal criteria provided by EMM, thus reducing empirical nature of particular Class Model. As starting point for Class Model generation process can be used any entity of EM (e.g. Actor) Process, Informational Activity, Material Flow). The final result of generation process is the particular Class Model that represents static structure of particular problem domain and can be used as business entities framework for IS developers. The UML Class Model is able to be used as one of the parts of KB-MDA method for programming code generation feature. The next task of KB-MDA method development is transition definition for Sequence model as well as Case Study analysis using KB-MDA method. This method is under development using the following technologies: MS SQL Server, MS .Net C#, XML, XMI, "MagicDraw" UML and is intended to be created as plug-in for MDA tools.

REFERENCES

[1] A. Lopata, M. Ambraziūnas, "Knowledge–Based MDA Approach", *BIS 2011 International Workshops, Series: Lecture Notes in Business Information Processing.* Poznan, Poland. vol. 97, pp. 160–165, 2011.
[2] T. Skersys, "Business Knowledge–Based Generation of the System Class Model", *Information technology and control,* vol. 37, no. 2, pp. 145–153, 2008.
[3] A. Lopata, S. Gudas, "Meta-Model Based Development of Use Case Model for Business Function", *Information Technology and Control,* vol. 36, no. 3, pp. 302–309, 2007.
[4] T. Yue, S. Ali, L. Briand, "Automated Transition from Use Cases to UML State Machines to Support State-Based Testing", *Series: Lecture Notes in Computer Science,* Birmingham, UK, vol. 6698, pp. 115–131, 2011.
[5] M. M. Gupta, N. K. Sinha, *Intelligent Control Systems: Theory and Applications.* IEEE Press, 1996, p. 820.
[6] *Standard ENV 12204: Advanced Manufacturing Technology Systems Architecture – Constructs for Enterprise Modelling,* CEN TC 310/WG1, 1996.
[7] D. Liu, K. Subramaniam, A. Eberlein, B. H. Far, "Automating Transition from Use Cases to Class Model", in *Proc. of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 03),* Calgary, 2003, pp. 831–834.
[8] F. Vernadat, 'UEML: Towards a Unified Enterprise modeling language", in *Proc. of the MOSIM 01*, Troyes, France, 2001.
[9] S. Gudas, T. Skersys, A. Lopata, "Framework for Knowledge-based IS Engineering", *Series: Lecture Notes in Computer Science,* pp. 512–522, 2004.
[10] Standard ENV 40003 – Computer Integrated Manufacturing Systems Architecture - Framework for Enterprise Modelling.– CEN/CENELEC.– 1990.