ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2008, Vol. 37, No. 2

BUSINESS KNOWLEDGE-BASED GENERATION OF THE SYSTEM CLASS MODEL[♣]

Tomas Skersys

Information Systems Department, Kaunas University of Technology
Studentu St. 50-309a, LT–51368 Kaunas, Lithuania

Abstract. The article presents principles of the Enterprise model-based generation of the Class model of the system (on the Platform independent level). Enterprise model being an integral part of the Repository of a CASE system becomes a core structure (Knowledge base) for the accumulation of business domain knowledge. The purpose of this article is to show that the knowledge stored in the Knowledge base is enough to generate one of the main models of the object-oriented Information systems development approaches, namely, Class model on the Platform independent level. In order to show the basic principles of the Class model generation algorithm, Enterprise and Class metamodels as well as mappings between the corresponding elements of these metamodels are presented and briefly discussed. The algorithm of the Class model generation and a brief overview of its realization are also presented in this paper.

Keywords: knowledge-based, enterprise model, class model, generation, UML, MDA.

1. Introduction

Recently, one can observe much effort coming from the developers of CASE (Computer-Aided System Engineering) tools to improve processes of the Information systems (IS) development life cycle using various techniques. One of the techniques is the automation of Information system development (ISD). Indeed, in theory all stages of ISD life cycle are closely linked and it should be possible to find stage-to-stage mappings to perform an automated transition from one stage of ISD to another. But the practice shows that most of the CASE tools still face serious problems while trying to reach some satisfactory results in this area. Usually, a model that was developed on the early stages of the ISD (e.g. Business modeling, User requirements specification) is not used for the generation of models on the later stages. System designer develops models of the system by analyzing earlier created models and relying on his own experience and knowledge about the problem domain. In other words, the transition from stage to stage is done empirically. P. Coad identified such logical gaps and called them "twilight zones" [4] nearly 20 years ago. This is one of the reasons why some still use CASE tools simply as a mean to nicely document the specification of the system under development [3]. Practitioners of Agile methods suggest using as simple tools as possible in order to accomplish one or another stage of ISD [2].

No doubt, with the advent of OMG's Model Driven Architecture (MDA) [15, 17] in 2001 the ISD automation processes gained a new boost. The OMG vision was that the models would be specified using UML and UML CASE tools would automate model-to-model transformations, especially forward engineering transformations: Business model (or Computation independent model – CIM) -> Platform independent model (PIM) -> Platform specific model (PSM) -> program Code. Some CASE tools already claim that they fully support MDA; however, in most cases this remains just a claim as some aspects of MDA itself still lack clear definition and are open to various interpretations, and the Repositories of such CASE tools do not store sufficient amount of domain knowledge in order to fulfil such claims [7].

UML alone is not enough to fulfil the main objectives of MDA at the moment. From our point of view, main problems with MDA arise when trying to define and specify Business model (CIM) using UML; yet another issue with MDA that has to be solved is the definition of mappings to realize the transformation step "CIM -> PIM". Objects and classes are core concepts for the object-oriented (OO) system analysis and design, and building the Class model of a system

[♣] The work is supported by Lithuanian State Science and Studies Foundation according to High Technology Development Program Project "VeTIS" (Reg.No. B-07042).

at the Platform independent level is among the main objectives of the OO software development. However, there is still no clear, well-developed process proposed to help the software engineers solve this problem successfully. Following the principle of MDA and assuming that UML Class model is a part of Platform independent modeling (that can later be transformed to one or more PSMs) it is obvious that the source of knowledge for the Class model development (generation) should be a Computation independent model and System requirements model (e.g. Use case model, that may be assumed as a part of CIM, or PIM). However, despite the progress in analysis techniques ISD still suffers from poor requirements acquisition, and full-scale business modeling often is not even recognized as an activity of the ISD. According to [11], 80% of software development projects fail or fall well short of their goals, or significantly overrun their budgets or schedules because of inadequate consideration of the business requirements.

The necessity to establish an explicit, logically motivated link between the business environment and the ISD processes (and, therefore, the IS itself) is relevant and recognized a long time ago, yet, as it has been mentioned already, there are not so many solutions found up to date. We found it the most promising to use the Enterprise model (EM) as the integrating link between the business environment and the ISD [7, 21]. Enterprise modeling stage is set as a starting point of the IS development life cycle here, and the Enterprise metamodel represents a structure for business domain knowledge accumulation – this conforms to the MDA vision of model-driven system development. Moreover, we believe that the use of the Enterprise model in the ISD eliminates, or at least narrows, the existing gap between CIM and PIM, and also gives a great benefit for automation process of the “CIM -> PIM” transformation.

The main purpose of this article is: to present the core of the Enterprise metamodel (EMM) and proposed Class metamodel (CMM); show how the elements of EMM are mapped to the elements of CMM; represent basic principles of the Enterprise model-based generation of the Class model. CMM can be presented as an extension to UML metamodel, but this is not the topic of this article.

2. Current Situation in “CIM -> PIM” Area

There is a great number of Enterprise modeling methods and approaches (such as CIMOSA, GERAM, IDEF suite, GRAI etc) [20], standards (ISO 14258, ISO 15704, PSL, ISO TR 10314, CEN EN 12204, CEN 40003 etc.) and supporting Enterprise modeling tools. Moreover, CASE tools which appear in contemporary market and are intended for the development of Information systems, include graphical editors for Enterprise modeling and analysis techniques. Business process modeling, as an integral part of Enterprise modeling, gradually becomes acknowledged as a part of any ISD process. However, the integration of Enterprise modeling techniques into the ISD process is still not sufficient.

MDA is one of the most significant attempts to standardize the object-oriented (OO) ISD process that is complemented with the Enterprise modeling (Business modeling) stage. Even though MDA declares Business modeling (CIM) as one of the stages of ISD, where CIM specifications should be transformed to PIM specifications [15, 17], it still remains more like abstract declaration with no clear definitions or rules, these transformations hold an empirical character. According to OMG and some other scientists, CIM is not obligatory and, if such model exists, could be used just as a guiding specification in the process of platform independent model (PIM) development [6, 18].

At present CASE tool developers concentrate on “PIM -> PSM” and “PSM -> Code” transformations; some tools generate DB schemas as well. However, modern MDA CASE tools (such as *AndroMDA*, *ArcStyler*, *OptimalJ*, *Together* or *MagicDraw UML*) do not support automated “CIM -> PIM” transformations, as there are no such well known methods that could be implemented at the moment. Class model of a system is a core model of the whole IS design. Nevertheless, there are no well defined methods of automated building (generation) of such Class models (on PIM level) from the business models either. Some supporters of Agile Model Driven Development (AMDD) say that in order to get correct PIM models transformations from CIM should be performed manually [1] (empiric knowledge). There are some approaches that propagate the development of the Class models from the user requirements gathered on system analysis stage – scenarios and Use case models are the most common examples in this case [12, 13, 19, 26]; RUP and ICONIX are among the most well known methods propagating Use case model-driven development of Class models. Some methods propagate the development of conceptual schemas based on linguistic analysis of the requirements [16]. However user requirements specified in a form of Use case models tend to be insufficient for the development of a Class model of a system, and Use case models with high level of detail become very complicated and hardly acceptable by the problem domain experts. This article supports the idea of automated Class model development from the main source of domain knowledge, i.e. Enterprise model [7, 9].

3. Enterprise Metamodel

The implementation of MDA approach in UML-based methods, that are capable to process Enterprise modeling activities, is highly desirable. However, the UML itself does not satisfy the needs and requirements for the domain knowledge modeling in the area of information systems engineering. IS engineering requires business-specific constructs and the Enterprise metamodel (accepted by users as business domain experts and IS developers) from which Enterprise models of specific business domain could be developed.

In [7, 9], basic concepts of the Enterprise metamodel (Figure 1) were presented. At the core of the EM is the interaction of *Function* and *Process*. A *Process* here is a partially ordered set of steps, which can be executed to achieve some desired material end-result. A process consumes material resources (it is an input of the process) and produces some material output (production). From the management point of view a *Process* is defined by two sets of attributes: a set of *Process state attributes*, and a set of *Process control attributes*. A set of Process state attributes includes process *Input* (material flow) attributes, process *Output* (material flow) attributes, and the attributes of the *Process* itself.

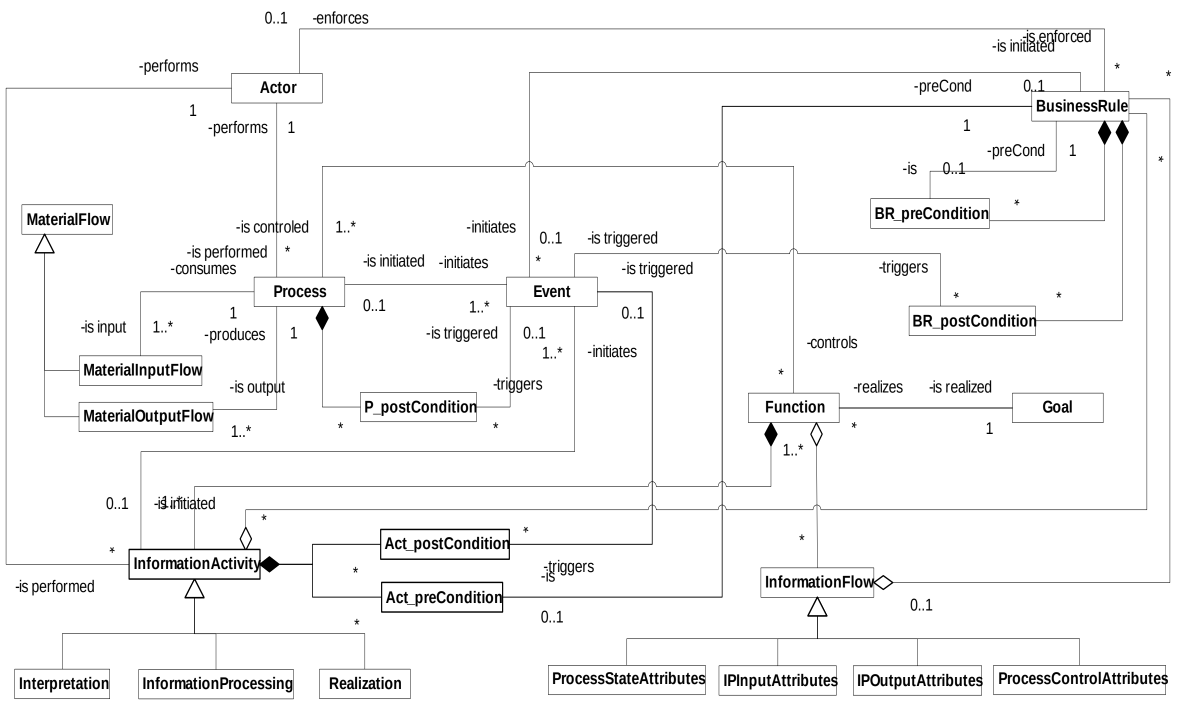

Figure 1. Enterprise metamodel (UML notation)

A *Function* is set to control the flow of one or more processes and the resources assigned to these processes. A function is comprised of the predefined sequence of mandatory steps of information transformation; these steps are called *Information activities* and can be of type *Interpretation, Data processing/ Decision-making (Information Processing)* or *Realization*. Inputs and outputs of information activities are *Information flows*. A sequence of information activeties composes a management cycle – a feedback loop. Making reference to the System and Control Theory, one can state that a process can be effectively controlled only if some feedback loops are implemented [5, 10].

Elements *Process*, *Function* and *Business rule* are triggered by occurrences of one or more events (*Event*). Processes, functions and business rules are performed by certain actors (*Actor*); construct *Actor* is an active resource (human, org. unit, application or machine with control device). Construct *Goal* represents a hierarchical structure of business goals of the organization. Goals of the organization are realized through (influence) management functions and directly influence the content of these functions (i.e. *Business rules*). Business rules are interpreted as the integral part of the decision-making mechanism of the organization. A decision-making mechanism in the proposed EMM is implemented through the composite construct *Function*. Construct *Business rule* in EMM defines conditions, constraints, and calculations to be associated with particular *Function* (its *Information activities*). Talking in object-oriented manner, function encapsulates a well-defined fragment of business logic that is expressed in a form of business rules. More about Business rules integration in EMM, and EMM itself, can be found in [7, 9, 21, 23].

4. Class Metamodel

In OO methods, Class models are typically used: as domain models to explore domain concepts; as conceptual/analysis models to analyze requirements; as system design models to depict detailed design of OO software. Class model is also a part of OMG standard, namely Unified Modeling Language (UML). Class model in UML-based CASE systems serves as a main source of knowledge for the development of Information system prototype: DB specification, graphical user interface (GUI), application code. However, from the ISD perspective UML metamodel is too complicated and heaped with unnecessary elements [25] and it seems that every new version of UML gets more and more complicated. From our point of view it is important to note that UML metamodel does not have sufficient set of constructs, essential for Business modeling (e.g. business rules) as it does not impose none of the fundamental business logic (e.g. feed-back loops, business rules, types of business objects etc.). In this article, Class metamodel is proposed. The Class metamodel is based on UML metamodel, but also incorporates constructs from the Enterprise metamodel (Figure 2).

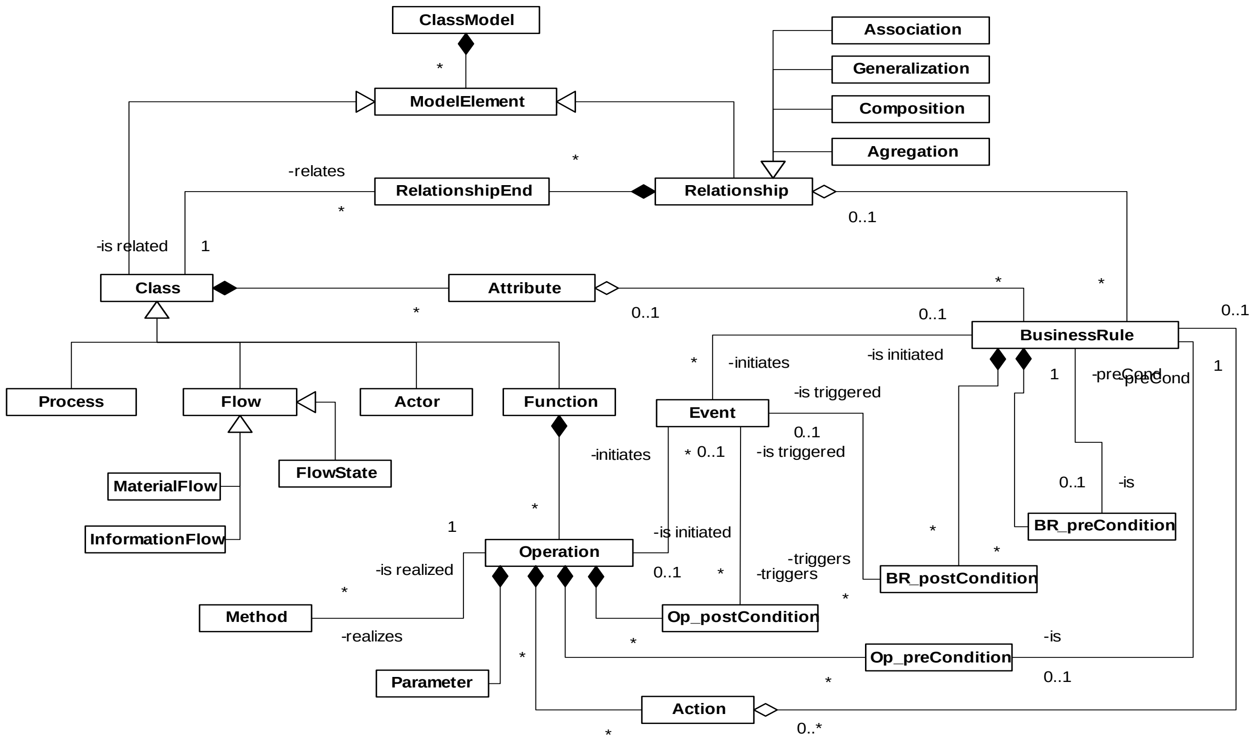

Figure 2. Proposed Class metamodel (UML notation)

The core constructs of the proposed Class metamodel (Figure 2) are as follows:

- Class model (*ClassModel*) is composed of the model elements (*ModelElement*). Class model elements can be either classes (*Class*) or relationships (*Relationship*) that hook these classes to each other. Each relationship has at least two connection ends (*RelationshipEnd*) and also may have some constraints or structural rules (*BusinessRule*) that specify that relationship.
- We enriched construct *Class* with certain subtypes: *Process, Flow, Actor* and *Function*. Such a modification is based on the specification of the Enterprise metamodel (Figure 1). The classification of classes is not a new idea – P. Coad's UML modeling in colour, Robustness diagrams are just a few examples of various class stereotyping techniques. Techniques that classify classes pursue certain practical goals. In our case this classification is made in order to make a close link between the business environment (Enterprise model) and the IS design models (in this case, Class model).
- Classes of type *Flow* may have states (*FlowState*).
- Traditionally, classes have attributes and operations. In the proposed Class metamodel every class may have attributes (*Attribute*), but the operation level (*Operation*) is specific only to the *Function* type classes. Construct *Operation* represents algorithmically-complex operations, and algorithmically-simple operations (such as Create, Connect, Access, Release) are not modeled in order to reduce the complexity of the class models. Classes of type *Function* are at some degree similar to the controller type classes in Robustness diagrams. Class attribute (*Attribute*) may have a number of constraining rules (*Business Rule*).
- Class operation (*Operation*) is composed of actions (*Action*), and may have parameters (*Parameter*) and methods (*Method*) that realize the operation in the certain programming platform. Class operations may also have pre- and post-conditions (*Op_preCondition*, *Op_postCondition*).
- Action (*Action*) represents single business rule (*BusinessRule*) of type *Computation*, *Action* or *Inference* (more on that can be read in [21, 23]). These rules may have pre- and post-conditions (*BR_preCondition*, *BR_postCondition*).

Business rules and operations may be initiated by events (*Event*), but also may trigger the activation of events themselves.

5. Principles of the Enterprise Model-based Generation of the Class Model

5.1. Correspondence between Elements of EMM and CMM

Mappings among the source and target models have to be identified before the algorithm of EM-based Class model generation is presented. In Table 1, it is shown how the elements of Enterprise metamodel are mapped to the elements of Class metamodel (φ: *EnterpriseModel* $\rightarrow$ *ClassModel*).

Mapping "φ2: <EMM.Function> → <KMM. Class>, <KMM.Function>" means that the element *Function* of EMM is mapped to the CMM elements *Class* and *Function* (*Class* being the core element of CMM and *Function* – type of that class).

Table 1. Mappings among the elements of EMM and CMM.

EMM element	Mapping	CMM element
<EMM.ModelElement>	φ1	<CMM.ModelElement>
<EMM.Function>	φ2	<CMM.Class>,<KMM.Function>
<EMM.Process>	φ3	<CMM.Class>,<KMM.Process>
<EMM.MaterialFlow>	φ4	<CMM.Class>,<KMM.Flow>,<KMM.FlowState>
<EMM.InformationFlow>	φ5	<CMM.Class>,<KMM.Flow>,<KMM.FlowState>
<EMM.Actor>	φ6	<CMM.Class>,<KMM.Actor>
<EMM.Event>	φ7	<CMM.Event>
<EMM.InformationActivity>	φ8	<CMM.Operation>
<EMM.Attribute>	φ9	<CMM.Attribute>
Rel-ships among EMM elements	φ10	<CMM.Relationship>,<KMM.RelationshipEnd>
<EMM.BusinessRule>	φ11	<CMM.Action>, <KMM.BusinessRule>
<EMM.BusinessRule>	φ12	<CMM.Relationship>,<KMM.RelationshipEnd>, <KMM.BusinessRule>
<EMM.BusinessRule>	φ13	<CMM.Attribute>, <KMM.BusinessRule>

The same principles are applied to all of the mappings in Table 1. It should be mentioned that a set of mappings {φ1, …, φ13} is sufficient to develop all the elements of Class model on the Platform independent level.

5.2. The Algorithm of Class Model Generation

The algorithm of Class model generation (Figure 3) on the basis of business knowledge stored in CASE system's Knowledge base (EM) will be presented in this section.

The core of the algorithm is composed of the set of mappings {φ1, …, φ13} and the set of rules of Enterprise model analysis and data querying. The sequence of the processing of these rules is managed by the CM algorithm itself.

Theoretically speaking, Class model can be developed on the basis of any element of EM (instance of an element), however, actual practical use can be gained from the Class models generated from the selected management function, technological process or structural element of the EM.

The core business object for generation is selected according to the purpose of the future IS (or subsystem of IS). If certain function of EM is selected (e.g. *Workload management*), the scope of the IS design will be narrowed to that segment of the business domain (i.e. to the particular management function *Workload management*); one should select a technological process (e.g. *Furniture construction*) of the EM if the goal is to prepare specifications for the computerization of all the management activities of that particular technological process; if one is willing to computerize some particular work place (e.g. *Accountant*) in the organization.

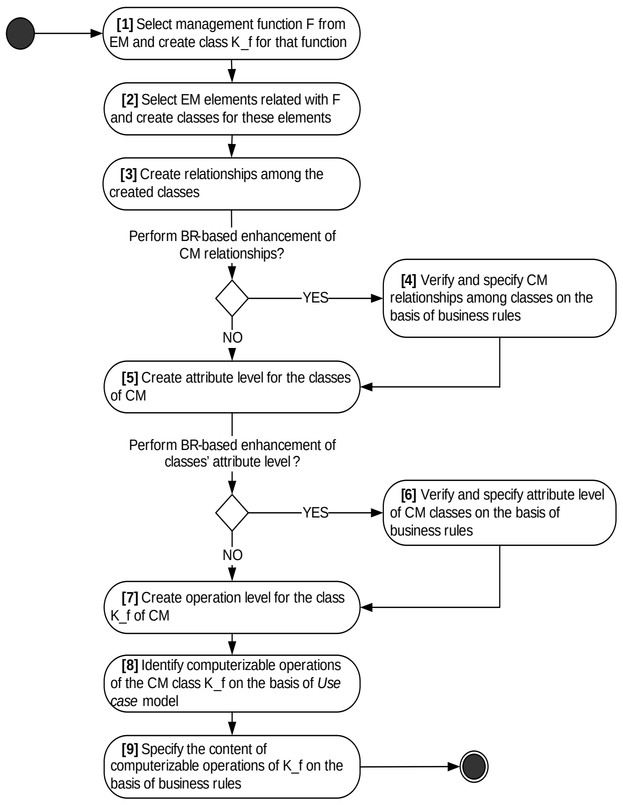
[1] Select management function F from EM and create class K_f for that function
[2] Select EM elements related with F and create classes for these elements
[3] Create relationships among the created classes
Perform BR-based enhancement of CM relationships?
YES
[4] Verify and specify CM relationships among classes on the basis of business rules
NO
[5] Create attribute level for the classes of CM
Perform BR-based enhancement of classes' attribute level ?
YES
[6] Verify and specify attribute level of CM classes on the basis of business rules
NO
[7] Create operation level for the class K_f of CM
[8] Identify computerizable operations of the CM class K_f on the basis of Use case model
[9] Specify the content of computerizable operations of K_f on the basis of business rules

Figure 3. CM generation algorithm (at the PIM level)

Class model generation should be performed on the basis of the selected structural unit of the EM, and the scope of the Class model development will be narrowed to the structural and functional aspects of that particular work place. This article concentrates on the Class model generation on the basis on the selected management function.

Class model generation algorithm (on PIM level) may be divided into four main stages (Table 2): (1) generation of the classes of CM, (2) generation of the relationships among the classes, (3) generation of the attribute level of the classes, (4) generation of the operation level of the classes. These stages may be further decomposed into steps.

Table 2. Stages, steps and mappings of the Class model generation algorithm

Stage	Step	Mapping
Stage 1. Identification of business objects of the problem domain and generation of classes for the Class model.	*Step 1.* Select management function F from EM and create class K_f for that function.	φ1, φ2
	Step 2. Select EM elements related with F and create classes for these elements.	φ1, φ3 - φ6
Stage 2. Identification and generation of relationships among the classes of the Class model.	*Step 3.* Create relationships among the classes.	φ10
	Step 4. Specify and augment relationships among CM classes on the basis of business rules.	φ12
Stage 3. Generation of the attribute level of the classes.	*Step 5.* Create attribute level for the CM classes.	φ9
	Step 6. Verify, specify and augment attribute level of CM classes on the basis of business rules.	φ13
Stage 4. Generation of the operation level of the classes.	*Step 7.* Create operation level for the CM class K_f .	φ7, φ8
	Step 8. Identify computerizable operations of the CM class K_f on the basis of *Use Case* model.	-
	Step 9. Specify the content of computerizable operations of K_f on the basis of business rules.	φ7, φ11

The sequence of the execution of the steps reminds of the traditional Waterfall model where every step is processed one after another one time. However, this sequence may be interrupted by the user (system analyst/designer) at any point and become iterative.

Let us shortly describe steps 1-9 of the algorithm in some more details.

Step 1. Generation process begins with the selection of the certain management function *F* (i.e. the instance of the EM element *Function*) from the EM. After the function *F* is selected, new Class model *M1* is created and the class *K_f* is created in *M1*. The name of *F* and the stereotype <<Function>> is assigned to the class *K_f* .

Step 2. Analysis of EM and data querying is processed. During this process instances of the EM elements *Process*, *Actor* and *InformationFlow* that are related to *F* are collected. Structural elements (*Actor* instances) and material flows (*MaterialFlow* instantces) that have relationships with the selected processes (*Process* instances) are also collected. For each collected instance of the EM elements *Process*, *Actor*, *InformationFlow and MaterialFlow* a corresponding class in *M1* is created. The names of the classes correspond to the names of the instances of the EM elements, and stereotypes of these classes are assigned with regard to the type of the particular EM element (i.e. *Process* -> <<Process>>, *InformationFlow, MaterialFlow* -> <<Flow>>, *Actor* -> <<Actor>>).

Step 3. The relationships among the classes of *M1* are specified with respect to the corresponding relationships among the elements of EMM. For example, if there is an association between the *Process* and *Function* in EMM, then there will be an association created between *M1* classes that have stereotypes <<Process>> and <<Function>>. In other words, Enterprise metamodel is the main guide in the process of the generation of relationships among the classes of CM.

Step 4. The relationships that were generated in Step 3 can be automatically validated and augmented with respect to the particularity of the problem domain. This is achieved using Business rules. Business rules can specify additional relationships between certain classes or augment the existing relationships with stricter cardinalities and other constraints (the latter may not be visible in graphical view of the Class model). Business rules-based specification and augmentation of the Class model is quite a complicated activity that was presented and extensively discussed in [22, 24], therefore this topic will not be further elaborated in this article.

Step 5. Attributes of the instances of the EM elements are mapped to the attributes of the corresponding classes of *M1*. Some system attributes of the instances of EM elements (e.g. system name, id) are also stored in the attribute level of the classes as system attributes – this is done in order to maintain close link between EM and CM. System attributes of EM allow us to track changes in business environment and react accordingly.

Step 6. The attributes that were generated in Step 5 can be automatically validated and augmented with respect to the particularity of the problem domain. This is achieved using Business rules. Business rules can specify additional attributes specific to a certain business object or specify constraints on certain attributes. Again, for more details on this step, refer to [22, 24].

Step 7. Algorithmically complex operations may be owned just by the classes of *M1* that have stereotype <<Function>>. Operation level for the class *K_f* is generated on the basis of the information activities (*Information activity* – Figure 1) that compose the management function *F* in EM. Each information activity is mapped to one operation of *K_f*. If the operation is quite complex, it can be further decomposed into a set of operations of lower complexity, however, it is advisable to perform such actions on the Enterprise modeling level (one can model a hierarchy of information activity workflows on the Enterprise modeling level [14, 8]).

Step 8. Operation set generated in Step 7 is a complete set of operations of the particular management function, however not all of them are necessarily computerizable. The best way to identify computerizable operations is to merge them with the use cases of the Use Case model (developed for the same problem domain) and find the overlaps. Not overlapping operations of the class *K_f* are automatically identified as non-computerizable and gain invisibility property (these operations can be changed to visible at any time later). The process of the Enterprise model-based development of Use Case models is presented in [14].

Step 9. The Repository of the Enterprise model stores not only structural rules (Terms and Facts) but operational rules (Computational, Inference, Action rules, Constraints) as well. Operational rules are used to formally specify the content of the computerizable operations of *K_f* declaratively. This is a third additional step of business rules-based specification and augmentation of CM.

It should be noted that Steps 4, 6, 9 can be performed independently, without a reference to other steps of the algorithm. Such an approach was demonstrated in [22, 24]. This means that the developed class model can be additionally validated and augmented using business rules at any time later.

The generated PIM level Class model can be additionally customized and be transformed into one or more PSM level models. In order to utilize the existing "PIM -> PSM" transformations, the proposed Class metamodel should be presented as an extension of the UML metamodel (this can be achieved using UML extension mechanisms).

5.3. Experimental Realization of the Algorithm

The prototype of the algorithm has been implemented as an add-on to the CASE tool Visio 2000. A short illustration of how the algorithm works is presented in Figures 4 and 5 – it is just enough to illustrate the main principles of the algorithm (Figure 3). Figure 4 presents the interaction of the management function "Order estimates management" and process "Order fulfillment". The problem domain is presented in a form of modified workflow models in the environment of *Provision Workbench* CASE tool. The workflow model is captured into EM using certain algorithms [14].

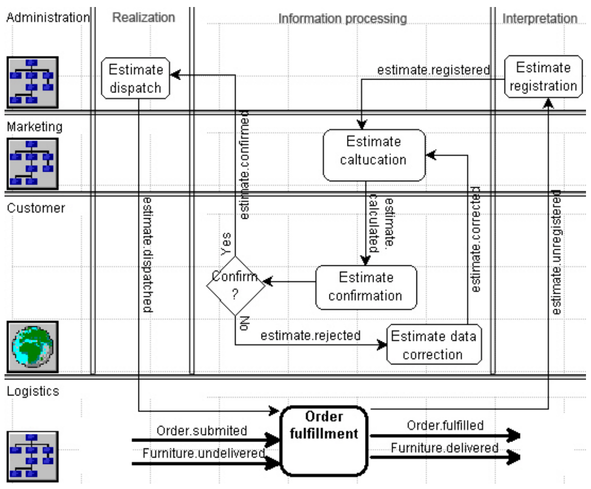

Figure 4. Workflow model of the problem domain "Order estimates management"

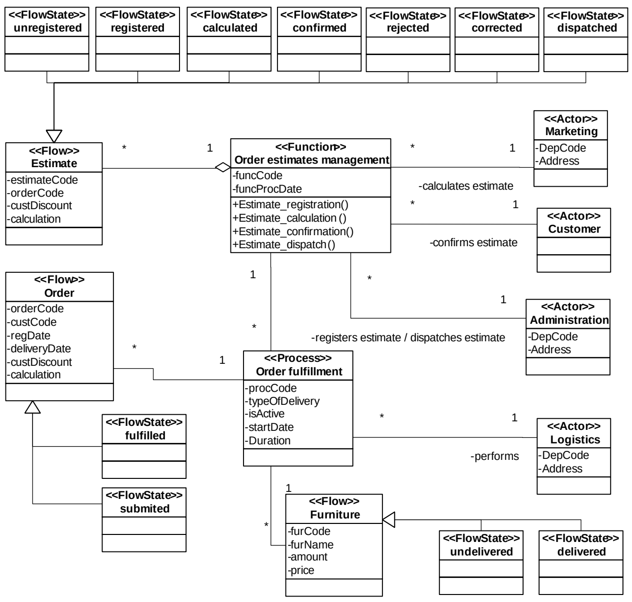

Figure 5. Generated Class model for the management function "Order estimates management"

Actors of the problem domain are deployed on the left side of the model (Figure 4). Actor *Logistics* performs technological process *Order fulfillment*. The process has material inputs – material flows *Order. submited* and *Furniture.undelivered*, and outputs – material flows *Order.fulfilled* and *Furniture.delivered* (*.submited*, *.undelivered*, *.fulfilled*, *.delivered* are the states of the corresponding flows). Actors *Administration*, *Marketing* and *Customer* perform certain information activities that compose the management function "Order estimates management". These information activities are of the particular type (*Interpretation*, *Information processing or Realization*).

Information activities have their inputs and outputs – information flows; these flows may also have states. Remark: we assumed that information activity "Estimate data correction" will not be computerized; therefore it was hidden in the operation level of the class "Order estimates management" (Figure 5).

It should be pointed out that not all of the knowledge of the business domain can be represented in graphical notation of the workflow models. Business rules are not visible in the graphical notation of workflow models – they are gathered using another specialized tool [21] and kept in formalized textual form apart from the workflow models. Features (attributes) of business objects cannot be visualized in Figure 4 as well, but they are stored in the Repository of EM. Each stereotype of the Class model may have its own colour setting, however, for the sake of a standard black and white paper presentation format all the background colours were reset to white (Figure 5).

The developed Class model is independent of any platform. In order to develop PSM models, new platform specific classes may be added and even the existing classes restructured. The boundaries of the developed Class model are restricted by the selected management function "Order estimates management".

Algorithmically complex operations are assigned to the stereotyped <<Function>> class. Classes with stereotypes <<Process>>, <<Flow>>, <<Actor>> may be assumed as entity (or persistence) classes which on the stage of DB schemas (or other data models) development are transformed into tables (entities) – these classes supply data for the operations of <<Function>> class. At the stage of user interface development, <<Actor>> classes may indicate the need for boundary classes as well.

6. Conclusions

Nowadays efficient IS development and Enterprise modeling are directly related issues. Enterprise modeling can be a source of enterprise knowledge that adds value to the business process and also influences methods of ISD. Some of the ISD approaches use Enterprise models as a source of structured knowledge about the real world (business domain) in ISD life cycle stages, such as user requirement analysis and specification, development of detailed IS project solutions and other. According to [11], MDA will not reach its goals unless the Business model (CIM) is formally connected to other layers of MDA (first of all to PIM) as well. Not the less important is to maintain such developed models and they will not be maintained unless they are connected to the code – this includes the Business model as well.

One of the main goals of this article was to show how Enterprise model could be used for ISD purposes. Enterprise model becomes the main source of knowledge in various processes of ISD, such as model development, augmentation and validation. Moreover, the usage of such EM facilitates the automation of model development and therefore the automation of the whole ISD. This article concentrated on the issues of Class model generation. The proposed solution narrows the existing logical gap between Business modeling and ISD stages and also automates the process of ISD at some degree.

References

[1] **S.W. Ambler.** A Roadmap for Agile MDA. *The Official Agile Modeling Site,* 2005. *Interactive, last visited* 2007 06 16, *www.agilemodeling.com.*

[2] **S.W. Ambler.** It's "Use the Simplest Tool" not "Use Simple Tools". *The Official Agile Modeling Site,* 2004. *Interactive, last visited* 2007 06 16, *www. agilemodeling.com.*

[3] **J. Bettin.** Model-Driven Software Development: An Emerging Paradigm for Industrialized Software Asset Development. *SoftMetaWare Ltd,* 2004.

[4] **P. Coad, E. Yourdon**. Object-oriented analysis. 2*nd ed. Yourdon Press, New Jersey,* 1991.

[5] ENV 12204 – Systems Architecture: Language Constructs for Enterprise Modelling. *Revision of ENV* 12204, CEN/TC 310/WG1, 2002.

[6] **W. Froidevaux.** An approach to MDA using MOF modeling and lessons learned. *OMG Meeting*'2002. *Helsinki. Document number: ad*/2002-09-20, 2002.

[7] **S. Gudas, A. Lopata, T. Skersys.** Approach to Enterprise Modelling for Information Systems Engineering. *INFORMATICA, Vol.*16, *No.*2. *Institute of Mathematics and Informatics, Vilnius,* 2005,175-192.

[8] **S. Gudas, A. Lopata.** Workflow models based acquisition of enterprise knowledge. *Information technology and control, Vol.*36, *No.*1A, *Kaunas University of Technology, Kaunas,* 2007, 103-109.

[9] **S. Gudas, T. Skersys, A. Lopata.** Framework for Knowledge-based IS Engineering. *Advances in Information Systems* (ADVIS'2004). *Lecture Notes in Computer Science Vol.* 3261, *Springer-Verlag, Izmir,* 2004, 512-522.

[10] **S. Gudas.** A framework for research of information processing hierarchy in enterprise. *Mathematics and Computers in Simulation, Vol.*33, *No.*4. *North Holland,* 1991, 281–285.

[11] **S. Hendryx.** Response to the OMG BRWG Business Rules in Models RFI. *Object Management Group Home Page,* 2002. *Interactive, last visited* 2007 06 16, *www.omg.org/docs/ad/02-11-02.pdf.*

[12] **D. Liu, K. Subramaniam, A. Eberlein, B.H. Far.** Automating Transition from Use-Cases to Class Model. *IEEE Canadian Conference on Electrical and Computer Engineering* (*CCECE*'03), *Calgary,* 2003, 831-834.

[13] **D. Liu.** Automating Transition from Use Cases to Class Model. *Thesis, University of Calgary,* 2003.

[14] **A. Lopata.** Enterprise Model Based Computerized Specification Method of User Functional Requirements. *Thesis, Kaunas University of Technology, Kaunas, Lithuania,* 2004.

[15] **S.J. Mellor, S. Kendall, A. Uhl, D. Weise.** MDA Distilled: Principles of Model-driven Architecture, *Addison-Wesley Pub Co.,* 2004.

[16] **L.C. Niba.** The NIBA Workflow: From textual requirements specification to UML-schemata. *International Conference on Software & Systems Engineering and their Applications* (*ICSSEA*'02), *Paris*, 2002.

[17] Object Management Group (OMG): MDA Guide v.1.0.1. *Object Management Group. Document number: omg*/2003-06-01, 2003.

[18] Object Management Group (OMG): Production Rule Representation. *Object Management Group, Request for Proposals. Document number: br*/2003-09-03, 2003.

[19] **D. Rosenberg, K. Scott.** Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example. *Addison-Wesley*, 2001.

[20] **J. Schekkerman.** How to Survive in the Jungle of Enterprise Architecture Frameworks. *Trafford, Canada*, 2003.

[21] **T. Skersys, S. Gudas.** Business rules integration in information systems engineering. *Proceedings of the* 13*th International Conference on Information Systems Development: Advances in Theory, Practice and Education* (*ISD*'2004), *Technika, Vilnius*, 2004, 253-263.

[22] **T. Skersys, S. Gudas.** The Enhancement of Class Model Development Using Business Rules. *Lecture Notes in Computer Science: The Tenth Pan-Hellenic Conference on Informatics* (*PCI*'2005), *Springer, Berlin*, 2005, 480-490.

[23] **T. Skersys.** Computer-aided information systems development method based on the business rules-extended enterprise model. *Thesis, Kaunas University of Technology, Kaunas*, 2006.

[24] **T. Skersys, S. Gudas.** Class model development using business rules. *Advances in Information Systems Development: Bridging the Gap between Academia and Industry* (*ISD*'2005), *New York*, 2006, 203-215.

[25] **D. Thomas.** UML – Unified or Universal Modeling Language? *Journal of Object Technology, Vol.*2, *No.*1, 2003.

[26] **K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer.** Scenario Usage in System Development: A Report on Current Practice. IEEE Conference on Requirements Engineering (ICRE'98), Colorado Springs, 1998, 33-45.

Received February 2008.