Model Oriented Enterprise Integration: Metamodel for Realizing the Integration

Maria Raffai

Chair for Information Science, Szechenyi University
Gyor, 9026 Egyetem ter 1, Hungary raffai@sze.hu http://rs1.sze.hu/~raffai

Abstract. The main task of IT-specialists in the last decades was to develop applications supporting different business functions and processes. For today the enterprises are well equipped with IT-devices and applications, but in most cases they run separately in various business units and/or on different platforms, they work with data stored not only redundant but also in other forms. Consequently there is an urgent need for integrating systems and fulfilling the entrepreneurs', customers', citizens' needs for on-line inter-organizational cooperation.

Keywords: *Enterprise integration, Inter-organizational cooperation, Modeling/mapping, metamodel, Metalanguage, Model transformation*

1. INTRODUCTION

Today the enterprises have to face the challenge of the increasing system complexity, the software-intensive business environment and the strong competition, and after all the entrepreneurs are under compression to exploit new technologies parallel with the increasing demand for reducing costs, improving quality and responding faster and faster the continuously renewing facilities. Electronic commerce, global markets, business-to-business communication, enterprise portals, Internet-based business transactions and services require fundamental changes. The IT (r)evolution has more and more progressive influence on the organizations, the intensity of their impact is increasing unbroken. From another aspect the demand for using enterprise integration adds a new dimension to the complexity. The developers must now focus not only to the disparate systems and data formats, but also to the differences between the enterprise's systems and the B2B message formats. The inter-organizational cooperation is fluid from the standpoint of the computer systems, the partnership undergoes continuous change therefore the companies are forced to merge their island systems by extranets.

Well, the Enterprise Integration (EI) is not a new problem. From the very beginning of computing using only mainframes at that time, there had already been taken efforts for synchronizing information across the organization. As the networks and the personal computers by the end of the seventies had become general new expectations arose all over the enterprises and beyond.

2. ENTERPRISE INTEGRATION

In order to make the term of integration clear, we have to interpret it's meaning in general. Integration means combining parts so that they work together and/or form a whole. In business and in information technology the term of integration can be interpreted in different ways [1]. From computer systems' approach the targets, the functionality and every other features of an enterprise are described in different forms and from different points of view depending on the specification scopes. This means that the integration process can relate on different components of the organization, so it is possible and in most cases desirable to integrate

- business units of inter-organizational enterprises,
- functions and processes enterprise-wide or partial,
- the requirements of the entrepreneurs, managers and/or employees,
- investments and costs of different business units,
- business components: resources, infrastructure, technology, products,
- IT systems: IT infrastructure, applications, data, corporate knowledge etc.

Beneath the term *enterprise integration* we understand a complex flow *from* discovering and analyzing a problem *through* designing the system *to* implementing it on the appropriate technology. In order to specify the main target of integration correctly it is necessary to see, that the organizations require interacting solutions for cooperation of

- the new and old (traditional or legacy),
- the custom and off-the-shelf and even
- the internal and external systems.

This also means that the enterprises need to define the exact goals of integration otherwise it can not meet the business requirements. Without specifying the scopes clear and utilizing IT technologies the integration process can not be effective.

2.1 Enterprise Application Integration

As the different business tasks and functions in most enterprises are already supported by information technology, as the IT solutions play definitive role so becomes the development of integrated working systems a key problem. The Enterprise Application Integration (EAI) is a conceptual IT category, which unites methods, techniques and tools, integrates applications within the enterprise environment in real-time. In order to make profit out an enterprise-wide integration of legacy systems and new application, the CEO and the CIO are forced to utilize the advantages of the IT technology.

Keeping the enterprise integration in sight and focusing on applications it is important to restrict the definitions. The Gartner' definition relating to EI/EAI seems to be relevant: *The enterprise integration is an emerging category of products that provide messaging, data transformation, process flow and other capabilities to simplify the integration of enterprise resource planning, legacy and other applications* [11]. In other worlds the EAI is the creation of the new strategic business solutions by

combining the functionality of the existing applications, the commercial packaged applications and the new codes using common middleware [2]. Today, the integration is not a technical issue anymore; it is not only an efficient tool to solve the communication problems between different subsystems. In most publications the EAI is interpreted as usages of architectural principles regarding to software and computer systems in order to integrate a set of computer applications.

2.2 EAI Implementation Pitfalls

The surveys and analysis relating to the results of integration projects show rather sad picture proving that it is a fairly difficult task. Some years ago it was reported that 70% of all EAI projects fail. Most of these failures are not due to the software itself or any technical difficulties, but basically to management issues. As the situation to 2007 has not changed considerably, the European Chairman of EAIIC has outlined main pitfalls undertaken by companies using EAI technologies as follows [3]: constant change, competing standards, loss of details, accountability, emerging requirements, lack of EAI experts, EAI as a paradigm, building interfaces as an art, protectionism. With this statement he called attention to the fact that the integration needs can be satisfied only by effective methods and tools and that the IT-professionals are responsible for reorganizing the computing systems in order to save the existing information assets. Making efforts to improve efficient solutions for EI, it is necessary to apply proven and well-accepted engineering methods and tools.

3. MODELING APPROACH

The modeling paradigm is one of the most important concepts for realizing the enterprise-wide integration. The *model* is a simplification of the reality, a blueprint of a system. It is the result of an abstraction process, which reflects the general, essential and permanent features from the modeling target's view, a formal specification to describe the functionality, the structure, and/or the behavior of the system. A good model includes those elements that have broad effect and omits on those minor elements that are irrelevant to the given level of abstraction. As the reality is very complex and complicated, it may be described from different aspects, that we call *model views* being semantically closed abstractions of a system. Developing models for an enterprise computing system prior to its construction or renovation is a well-considered abstraction process. The efforts to implement the complexity of the systems underline the importance of using good modeling techniques [4].

3.1 Model Driven Engineering

The Model Driven Engineering (MDE) refers to the systematic use of models applied to software, system and data engineering as primary development artifacts throughout the whole engineering lifecycle. There are already several attempts that give framework proposals for MDE process. In order to solve the integration

problems effective the leading companies are pressed to work out and define a common acceptable solution.

The OMG TF (Object Management Group Task Force; being responsible for the standardization) in 2005 accepted the MDA (Model Driven Architecture) as a standard of model driven engineering concept. This framework is an innovative approach to construct enterprise architecture by abstracting and visualizing business requirements in the form of technology independent models. The MDA separates implementation details from business functions 4, and it gives chance for Rapid Enterprise Integration [7]. It is a set of guidelines for structuring specifications expressed as models, and it defines system functionality using a platform-independent model (PIM) based on the appropriate Domain Model. The MDA principle can also be applied to business process modeling where the PIM is translated to either automated or manual processes. With full knowledge of Platform Definition Model (PDM) the PIM is translated to one or more platform-specific models (PSMs) what computers can run.

3.2 Enterprise Modeling

An enterprise model is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals and constraints of a business, government or other enterprise system. An enterprise model is "... an attempt to describe the interrelationships among a corporation's financial, marketing and production activities in terms of a set of mathematical and logical relationships which are programmed into the computer" [10]. These interrelationships should represent in detail all aspects of the firm including "... the physical operations of the company, the accounting and financial practices followed, and the response to investment in key areas" [8]. Enterprise modeling is the process of improving the enterprise performance through the creation of enterprise models. This includes modeling both of business processes and IT. There are several techniques for modeling the enterprise such as Active Knowledge Modeling, Process Modeling (CIMOSA, PERA, LOVEM and DYA etc.), Object-Oriented Modeling and/or modeling the enterprise with multi-agent systems. The enterprise mapping process is much more important than ever before; as it is an accepted architecture

- to integrate what you have built, with what you are building, with what you are going to build,
- to declare the implementation of one or more platform-specific models (PSM) based on the platform-independent model (PIM),
- to remain flexible in the face of constantly changing infrastructure and
- to lengthen the usable lifetime of the software, lowering maintenance costs and raising ROI (Return of Investment).

3.3 The Metamodel Concept

The highest level of the abstraction is the *metamodel level*. The word Meta is a prefix used to indicate the result of an abstraction process, a map expressing

something *about its own category*. For example, metadata are data about data (who produced it, when was it produced, what format the data are stored and so on). Most general, *metamodeling is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for the modeling in a predefined class of problems*. This concept is composed with the notions of the terms meta- and modeling. Thus metamodeling is the construction of a collection of concepts within a certain domain, a precise definition of the constructs and rules needed for creating semantic models. As a model is an abstraction of real world phenomena, a metamodel is yet another abstraction, highlighting properties of the model itself in the form of an abstract language for defining different kinds of metadata. From computational perspective, the metamodel concept is used practically in computer science and also in computer system/software engineering process. Analyzing the metamodels they are closely related to ontology, as both are often used to describe and analyze the relations between concepts [9], and they can also be considered as an explicit description (constructs and rules) of how a domain-specific model is built. In particular, it comprises a formalized specification of the domain-specific notations.

Putting the question, *what are the metamodels good for*, we can give several answers depending on the purpose for which any given metamodel was developed for? In most cases the common purpose is to give: (1) a *schema* for semantic data that needs to be exchanged and/or stored in a repository, (2) a *language* that supports a particular methodology or process. In this sense the *metadata* is a general term for data that *describes information about data and models*. But the model gets here broader meaning as usual, it means any collection of metadata that is related in the following ways: (1) it describes information that is related with itself, (2) it conforms to rules governing its structure and consistency; that is, it has a common abstract syntax, and (3) it has real meaning in a common semantic framework.

4. THE ENTERPRISE INTEGRATION METAMODEL

The Enterprise Integration MetaModel (EIMM) is a framework for defining metadata, representing the integration process currently underway in the areas of object repositories, object modeling tools and a metadata management in distributed business environments. Since there are many possible kinds of metadata in a system, the EIMM has to contain information about many different business models, which are integrated by defining a common abstract syntax specified for metamodels 5. This *metamodel involves information about components, architecture and characteristics of all models reflecting the corporate features* from different views, and it supports the enterprise integration process on different mapping levels. Designing a generally applicable metamodel it is important to use standards, modeling and transformation tools that help to realize the enterprise-wide integration).

The first step to define a metamodel is to distinguish the different *enterprise models* (see Table .) and then to define the *metamodel architecture* itself (see figure 1).

4.1 The Metamodel Architecture

Although a model view represents the functionality, the structure or behavior of a system, the models can perform their purpose only if we separate the domain specific information from the business details, the relevant business-specific information from the technical details and the complexity of its implementation. In order to meet every demand of the integration needs the EIMM has to be developed for satisfying these requirements. The *four layered architecture* of the enterprise metamodel logically follows the mapping process satisfying the needs for generating clear understandable and consecutive models that have already been cleaned from the elements and features not relevant to the developing purposes. Let me explain the layers of EIMM more detailed!

Table 1. Model Classification Viewpoints and the Categories

Model Classification	Model Categories
modeling aims	the target of the mapping/modeling process
model creation	defined by creation date, mapping methods
model types	depending on modeling subject: industrial, trade, healthcare etc. models
mapping approach	the general concept and philosophy used during the mapping process
model views	*from actors' point of view:* users, owners, developers view
	from system approach: components, architectural, process, actor, control, methods, behavior, business units, devices etc. views
model details	specification levels differentiating in details of model definitions
mapping levels/layers	domain, business, platform-independent, platform-specific, implementation, deployment, operation etc. levels
used modeling tools	process mapping tools, standards, modeling languages, executable transformation tools,
model interfaces	*interface issues:* generated, received etc.
	interface form: paper, electronic, etc.
data models	data forms, management systems
transformability	platform identity, common interfaces, transformation tools

The first level of the abstraction is the business domain layer (M1), which reflects the main characteristics of the business processes, the business and process elements and their relations. This model is often called Computational Independent Model (CIM), because it focuses on the real processes, the system features, the functionality and the nature of the modeled system described in different forms (e.g. verbally, graphically, in matrix or mathematical forms) and stored in different documents. Figure 1 shows the components of the domain model that reflect on the activity of different business units and handles and stores data on different places and platforms.

In order to have a correct base for enterprise-wide and inter-organizational integration it is necessary to separate the unimportant components and features, and to map the domain model to business model. This abstraction process results another set of models that play definitive role in creating a correct enterprise metamodel, not only because it constitutes the basis of the enterprise integrated system, but also because it identifies and controls the deployment, maintenance and the software quality assurance. The business model layer is expressed mainly in use cases and object

models; it is implemented as classes with interfaces, activity, sequence and collaboration diagrams.

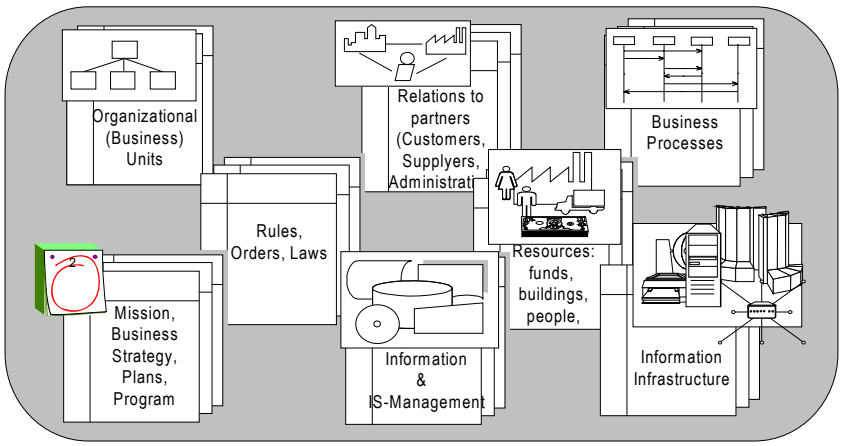

Figure 1. The Business Domain Represented in Different Forms

Going ahead on the mapping process the developers have already information about the analyzed organization. The model set shows the internal features and also the external relations of the given organization and it can be expressed as a number of classes. These classes represent both the static and the dynamic characteristics of the system. As the main purpose of the abstraction process is to have information for the integration, it is necessary to have unambiguous specification about all the subsystems no matter, they are manual operated, fully/partly computer aided, stand alone and/or partly or fully integrated. The object model is the right metamodel layer for mapping the system and for expressing relations and static/dynamic features. It is also destined for being the starting point of the next integration process steps. The classes of EIMM object model represent metadata about the business models expressing the relations and characteristics of these models.

Figure 2 shows the object model package containing the business metadata.

The classes of the EIMM object model are ordered into three packages.

1. The business package (businessPackage) involves all the corporate models that are described in different forms and languages (e.g. textual, diagram, matrix, graph form). This package contains 6 classes: (1) The class businessUnit describes all kinds of models reflecting to the components, relations and features of the business units and has information about the models in the form of attributes. (2) The class processes specifies characteristics of every models that mirror all business processes operating in separate workplaces. (3) The class resources contains data about models relating to all of the resources that are owned and/or used within the organization in different places, in different business units and are needed for different operations. (4-5) The classes stratProg and rulesLaw have

information about document models, that describe models of the relevant rules, and/or models that mirror enterprise strategy and programs. (6) The class orgEnv summarizes the data about the models describing the enterprise environment, e.g. vendors, competitors, suppliers, administration partners.

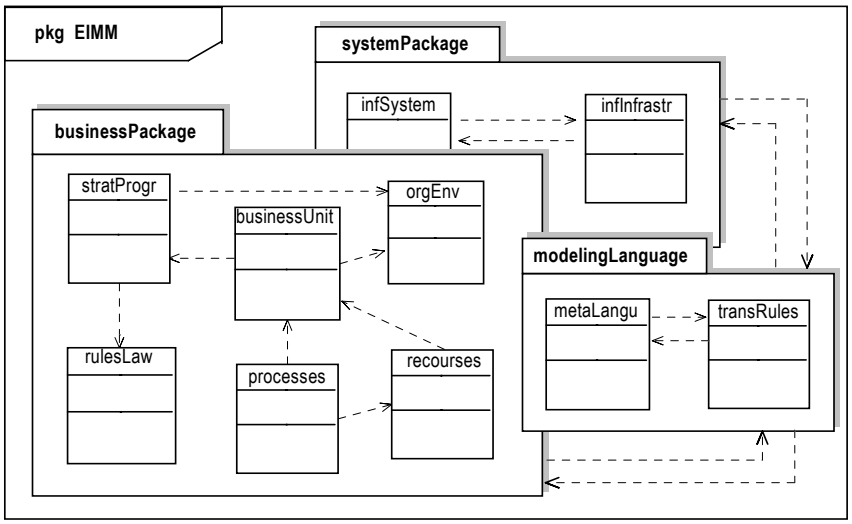

Figure 2. The Object Model Representing Metadata about the Business Model

2. As the IT plays a special role in the organization by providing different services it is necessary to distinguish the components from the business elements. As the business so the computer system can also be analyzed also from different views. From integration point of view it is required to separate the infrastructure from the running system, so the system package (systemPackage) contains two classes: (1) The infInfrastr class contains information about the IT models, it describes the platform on which the different system components are deployed and where they are running, e.g. data about server model, network model. (2) The infSystem specifies the characteristics of the soft models, e.g. operation systems, protocols, programming languages, applications.

The modeling language package (modeling language) is for fulfilling the requirements suitable for automated interpretation. This package contains two classes:

(1) With the help of the modeling languages the different enterprise model views can be expressed in a good understandable form; usually not verbally but with graphical notations and syntax and with a special mechanism to define and specify the abstraction of the system. As every model view has different components, the modeling languages also need to be defined what kinds of components they work with, what are their relations and behavior, and of course the syntax and rules of

usage. Every semantics and syntax of model description is specified in the metaLangu class (see figure 3).

(2) An integration process can only be effective if we have knowledge enough to transform the models from one level of abstraction to the next level. The transformation can be carried out only if we have knowledge enough about the syntax and also the semantics of the modeling language, and we find a good and fast approaching rule system. The class transRules involves all the important data reflecting to the transformation process.

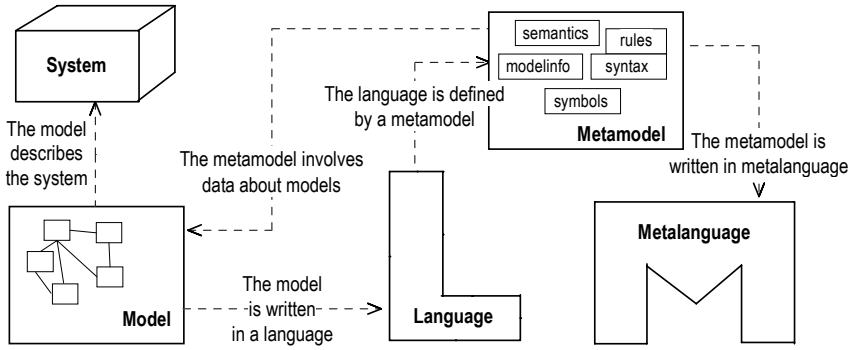

Figure 3. Relation of Models, Languages, Metamodel and Metalanguage

4.2 The Model Transformation Process

The specialists responsible for integration have to understand the models on different abstraction levels, and in order to make these models capable for interaction it is necessary to transform them on a unique way. For example the source of the analysis model is the business model; the design model is generated from the analysis model. In every model transformation step there are exact rules what describe how a model in the source language has to be transformed into a model in the target language. For making the transformation process effective the developers need *transformation tools* that possess definitions describing how a model should be transformed. In order to apply the transformation tools in all environments and on every platform independently of the source model we need different subsets of definition. These subsets are formalized by profiles that define completely new language and make possible to convert models into different other languages.

5. CONCLUSIONS

As the enterprise-wide integration is not only a challenge, but it is the condition of the competitiveness and even of the organizational surviving I intended to develop a generally usable solution: a metamodel framework. The EIMM containing information about the different enterprise models is a powerful base to start the integration process. Filling the repository with model information up, the developers are able to design and create high level integrated systems by knitting the preexisting

islands, the manual operated and the computer aided systems together instead of replacing the existing systems/subsystems with extraordinary expensive fully integrated new ERP systems (e.g. SAP). The EIMM enterprise integration architecture provides an efficient framework in which the effective legacy systems in cooperation with the newly developed software are able to continue their function on a very competitive way with only few losses of human, material and technological resources.

The model driven EAI technologies are still being developed and there isn't consensus on the ideal approach or on the appropriate technology a company should use. With developing enterprise integration metamodel I intended to give a generally applicable framework for the developers contributing to the efforts that many institutes and professionals make in order to increase the effectiveness of the enterprise integration.

REFERENCES

1. *OMG Meta Object Facility V 1.4*, OMG http://www.omg.org/ (Accessed April 10, 2002)
2. T. Naylor, *Corporate simulation models and the economic theory of the firm*, in *A. Schrieber (editor), "Corporate simulation models"* (University of Washington Press: Seattle, 1970), pp.1-35.
3. E.B. Söderström, P. Andersson, E. Johannesson and B. Wangler, Towards a Framework for Comparing Process Modeling Languages, in *Lecture Notes In Computer Science, Volume 2348, Proc. of the 14th ICAISE* (2001).
4. J. Bézivin, From Object Composition to Model Transformation with MDA, in *Proc. of IEEE-Tools-39* (Santa Barbara, USA, 2001).
5. M. Raffai, The UML-based Transformation of the Domain Model - Modeling Strategy for the Enterprise Application Integration, in *Proc. of Business Information Systems*, Gyor (2003).
6. G. Trotta, *Dancing Around EAI 'Bear Traps'* Retrieved on 2006-06-27 (2003).
7. M. Raim, *Implementation Infrastructure: Enablers for Rapid Enterprise Integration*, in *OMG Information Day* (2002)
8. GartnerGroup Inc. *Glossary.* www.gartner.com (Accessed March 27. 2007).
9. G. Gershefski, What's happening in the world of corporate models? *Interfaces*. Volume 1, Number 4, (1971).
10. W.A. Ruh, F.X. Maginnis, and W.J. Brown, *Enterprise Application Integration: a Wiley Tech Brief* (John Wiley & Sons: 2001).
11. *Wikipedia Enciclopedia.* http://en.wikipedia.org/wiki/ (Accessed April 7. 2007).
12. M. Raffai, Increasing IS Requirements and New Engineering Technologies, in *Proc. of 10th Interdisciplinary Information Management Talks* (Zadov, 2002).