Cartography of the Enterprise Modeling Approach

Hatem Ben Sta

***Abstract*—In this paper we will elaborate the state of the art related to Enterprise Modeling and Interoperability achievement. Also we give a detailed description of tools and method dealing with interoperability such as Model Driven Architecture. The paper proposes a move towards a description of Enterprise Modeling Approach and the different concepts which we will present.**

***Index Terms*—Interoperability, enterprise modeling, MDA, metamodeling**

I. Introduction

In recent years, it has become increasingly necessary for enterprises to use software and computer-based systems if they are to grow. In order to improve their competitiveness, these enterprises have certain common needs, such as sharing and processing information. Furthermore, they no longer want to be dependent on continually changing technologies and frameworks, and seek to preserve their knowledge and know-how when they switch from an existing technology to a new one. They also need to make this change in a shorter time. This is where the challenge of productivity and Interoperability arises.

On the one hand, and to give a promising answer to the productivity problem, the Model Driven Architecture (MDA) approach was proposed by the Object Management Group (OMG) in 2001. This approach intends to promote the use of models as a basic way of designing and implementing systems. It also aims to provide enterprises with the ability to evolve with rapidly changing technologies. On the other hand, the use of different systems and technologies by the same enterprise raises the needs to make these applications (which are based on different platforms) communicate with each other. This is what is called the Interoperability problem between Enterprise Applications and software. Interoperability is defined as "the ability for a system or a product to work with other systems or products without special effort on the part of the customer". It can also be defined as "the ability of Enterprise Software and Applications to interact" [1]. In order to consider that interoperability has been achieved, interactions between Enterprise Applications must take place on at least three levels, data, application and enterprise business.

Nowadays, all kinds of enterprises are involved in achieving interoperability, regardless of whether they are large organisations or small and medium-sized enterprises (SME). In this context, a given enterprise is considered to be competitive or not depending on its ability to interoperate with other enterprises. The INTEROP Network of Excellence project is one of the main projects that have been set up to deal with Enterprise Modelling and the Interoperability issue. This project, which is backed by the European Commission, aims to create the conditions for innovative and competitive research in the field of Interoperability for Enterprise Applications and Software. The INTEROP Project has three main research domains: the first deals with the subject of Enterprise Modelling in order to define interoperability requirements. The second research domain, called Architecture and Platforms, aims to provide solutions to the question of how to implement frameworks. The third domain, named Ontology, aims to identify interoperability semantics within the enterprise. The subject of our research concerns the domains and objectives of the Enterprise Modeling Most of our work are involved the search for solutions to the issue of achieving Interoperability and Enterprise Modelling. The aims are to analyse and propose solutions for achieving Model Driven interoperability. It is also involved in another subject, which intends to facilitate interoperable Enterprise Software Applications (ESA) using the model-driven approach. Finally, our project aims for the future is to provide a complete model of interoperability achievement by using the findings of its work and all the results obtained in other existing solutions.

Manuscript received October 9, 2012; revised November 15, 2012.

Hatem Ben Sta is with the LI3 – Laboratoire de l'Ingénierie Intelligente des Informations, Higher Institute of Computer Science, Universit éof Tunis at El Manar 2, Rue Abou Rayhane El Birouni 2080 Tunis, Tunisie (e-mail: Hatem.Bensta@ensi.rnu.tn).

II. Enterprise Modelling

A. General Overview of Enterprise Modelling

Over the last decade, there has been an overwhelming demand for different Enterprise Modelling (EM) techniques due to their benefits both in industry and in acquiring experience and know-how. Enterprise modelling represents and structures the business organisation as well as its internal structure and its relationships with the external environment. This enables enterprises to adapt to change quickly and easily, to satisfy customer requirements, and to improve their performance. Consequently, there has also been an increasing need to implement and integrate enterprise information systems with software engineering approaches. Enterprise modelling is the key fact to achieve this integration because it provides models that are used in planning and optimisation activities. According to Vernadat definition, Enterprise Modelling is "externalising and expressing enterprise knowledge" [1, 2], which allows enterprises to improve and share their knowledge between the different members of staff. This is really the key to why modelling has an important value for the enterprise. This enterprise knowledge represents the firm's operations and

resources, such as business activities, behaviour, and organisation units. These enterprise operations and resources are represented in computational enterprise models, through enterprise modelling techniques and tools [3]. The firm's workers use these models to obtain the enterprise design, analyses and operations that best fit the business requirements in general. In addition, the capability of enterprise models can be expressed in terms of the following aspects [3]:

1) Purpose and scope of the model
2) Expressiveness of the Enterprise Modelling Language, using several graphical representations.
3) Capabilities of supporting tools
4) Values to be created using the Enterprise Modelling Language
5) Modelling process supports and facilities
6) Scalabilities, extendibility and interoperability of the infrastructure that supports the modelling Enterprise Modelling is considered to be a prerequisite to achieve Enterprise integration.

B. Main Goals of Enterprise Modelling

The main purpose of Enterprise Modelling is to help enterprises to capitalise on and improve their experience and know-how by sharing enterprise knowledge. Furthermore, Enterprise Modelling is used for other purposes that can classify as follows [2], [3]:

Analysis: The aim of Enterprise Modelling in this area is to detect problems and define solutions in order to improve enterprise business, as follows:

1) Business analysis for problem detection
2) Business process reengineering
3) Compliance with existing quality standards (ISO 9000, ISO 14000)
4) Decision-making design and simulation
5) Enterprise integration and interoperability

Technical: In this area Enterprise Modelling is useful for enterprise implementation in the following situations:

1) Analysis, design and implementation of information and computer systems
2) Implementation of enterprise system (ERP)
3) Control of enterprise's processes using models

Another classification of the purposes of enterprise modelling is proposed as follows [3]:

4) Human sense-making and communication: one of the main purposes of enterprise modelling is to make sense of aspects and to help along communication between people
5) Computer-assisted analysis: Enterprise Modelling provides a gain in knowledge about the enterprise through simulation and mathematical deduction

Model deployment and activation: Enterprise Modelling aims to integrate enterprise models in the firm's information system. In this context, there are three ways to integrate models:

1) Manually: this solution is used when the Model Process is not supported by the system.
2) Automatically: where the system plays an active role
3) Interactively: where the computer and users work together to interpret models in different situations
4) The enterprise model: This is considered to be a basis that gives a description of the traditional development project EM is becoming more and more sought-after in industry, because it allows the enterprise experience and know-how to be capitalised on.

C. Enterprise Modelling Standards

This section reviews some of the international standards concerning enterprise modelling, such as EN/ISO 19439: Enterprise Integration – Framework for enterprise modelling; ISO 14258: Concepts and Rules for Enterprise Models; ISO 15704: Requirements for Enterprise Architectures and Methodologies; or ISO/IEC 15288: Life-Cycle Management System/Life Cycle Process [4].

EN/ISO 19439: Enterprise Integration – Framework for enterprise modelling: Approved in 2002, this standard aims to provide and define the main generic concepts required to enable the creation of enterprise models and to provide the main basis for enterprise engineering modelling [4]. It also serves as a basis for identifying and coordinating the development of standards for modelling enterprises. It is also used by further standards for the development of models.

EN/ISO 19400: Enterprise Integration – Constructs for enterprise modelling : ISO 19439:2006 specifies a framework conforming to the requirements of ISO 15704, which serves as a common basis for identifying and coordinating the development of standards for modelling enterprises and emphasises, but is not restricted to, computer integrated manufacturing [4]. ISO 19439:2006 also serves as the basis for further standards for the development of models that will be computer-enactable and enable model-based decision support of business processes leading to model-based operation, monitoring and control. In ISO 19439:2006, four views of enterprise models are defined in this framework. Additional views for particular user concerns can be generated but they are not part of this International Standard. Possible additional views are identified in ISO 15704.

D. Enterprise Modelling Languages

Over the last few years, many enterprise modelling languages have been elaborated. These methods focus on the informational view and functional view of a business organisation. We will now go on to describe the main enterprise modelling languages that have been used. Each of them has its own purpose and scope. Entity-relationship method: this method is based on the Entity-Relationship (ER) model created by Chen (1976). It provides models that describe the organisation's structure in terms of entities, relationships among entities, and their attributes. This method describes, through its models, the informational view of an enterprise.

SADT: Structured Analysis and Design Technique is a language used for communicating ideas. It provides a graphical notation with which to represent complex systems in terms of sub-systems. SADT provides a simple definition of an activity as well as its related properties, i.e. input, output, control, and so forth.

IDEF: Integrated DEFinition is a set of modelling methods (IDEF0, IDEF 1, and IDEF 2, and so forth) that can be used to describe operations in an enterprise. IDEF 0 methods

allow for the creation of graphical models containing the different functions of an enterprise. These models show all the necessary information related to enterprise functions, such as who performs the function, what it produces or what relationship exists between functions.

Petri nets [5]: is also a graphical and mathematical modelling method that consists of places, transitions, and arcs that connect them. This tool is used to analyse and represent systems that are characterised as being concurrent, asynchronous and distributed. The Petri nets tool includes a set of various extensions, like object-oriented Petri nets [4], which can be used to achieve enterprise modelling.

SA/RT: Structured Analysis with Real Time extensions is a complex method designed for system analysis and design. It is one of the most frequently used methods in technical and real-time applications. This method addresses the function and information views of an enterprise system with a special emphasis on behavioural aspects [4].

OOA and OMT: Object Oriented Analysis (Booch, 1994) and the Object Modelling Technique (Rumbaugh et al., 1991) are object-oriented methods used for enterprise modelling. These methods, which can be applied at the requirements level and design modelling level, cover only the informational perspective of an enterprise.

CIMOSA: Computer Integrated Manufacturing Open System Architecture is one of the most popular enterprise modelling architectures. It covers functional, informational, resource and organisational aspects of an enterprise at various modelling levels. This architecture, developed by the AMICE consortium, defines concepts and rules to facilitate the building of systems [4].

ARIS: Architecture of Integrated Information Systems is a product that provides architecture for the information system that includes the functional view, the control view, the data view and the organisational view of an enterprise [4]. ARIS is used for enterprise modelling at the requirements, design and implementation levels.

E. Projects and Works Dealing with Enterprise Modelling:

The UEML project: UEML, which stands for Unified Enterprise Modelling Language, is a thematic network project founded by the European Commission in order to find a solution to the problem of multiple modelling languages [6]. This project has a common long-term objective that consists in defining a Unified Enterprise Modelling Language. The main characteristics of this language are the following:

1) To provide the business community with a common template-based language. This template will be used on top of commercial enterprise modelling tools.
2) To provide a basis for sharing and exchanging enterprise models implemented with different tools.
3) To support the implementation of an evolutionary Enterprise Model.

During this project, several Enterprise Modelling approaches, like the Generalised Enterprise Reference Architecture and Methodology (GERAM) framework and the Zachman Framework for Enterprise Architecture were compared. The purpose of this comparison was to identify the strengths and weaknesses of each approach in order to be able to give advice and suggestions for the future Enterprise modelling language that will be created. This comparison was made using a UEML framework, which has the following four dimensions:

The enterprise engineering world components dimension: In this dimension, four levels are defined: enterprise instances, enterprise models, enterprise modelling language and enterprise engineering methodologies, and enterprise metamodelling languages and methodologies for the definition of enterprise engineering methodologies.

The enterprise life-cycle dimension: This dimension includes different states of definition of the enterprise throughout its life cycle. The main phases that are considered during this life cycle are initialisation and definition of objectives, definition of requirements, design, and implementation.

The genericity dimension: This dimension aims to classify different enterprise modelling approaches according to their applicability to a diversity of situations. This dimension defines degrees of genericity, like the generic, the partial, and the particular degree.

The view dimension: This dimension is used to decrease the apparent complexity of enterprise models.In this dimension, the following views were defined: the Process/Functional View, the Informational View, the Resource View, and the Organisational View.

III. Model Driven Architecture

A. MDA Development Life Cycle

The traditional software development process: The software development process represents a typical process that is used nowadays. This development process includes the following phases: Gathering of requirements, Analysis, Design, Coding, Testing and Deployment.

The documents and diagrams are the major artefacts during the first three phases. They rapidly lose their value as soon as the coding starts. The main problem of this life cycle is that the changes are often carried out at code level and because of programmers claim that there is not enough time to update diagrams and other high-level documents. Moreover, while the coding phase progresses, the connection between the diagrams and the code fades away. Because of this, maintenance of the software system becomes very difficult due to the incoherence between the different diagrams that describe the system and its real implementation or source code. This gives rise to a real need to adopt and create a new development life cycle, which forces developers to take care of both the documents and the source code.

The MDA Development Process: The MDA development life cycle, which defined by OMG [7], looks like the traditional one and contains the same phases. The only major difference between the MDA development life cycle and the traditional one concerns the nature of the artefacts that are produced. Thus, the MDA development life cycle [8] comes with a new type of artefacts, which are based above all on formal models that can be interpreted by computers.

The main artefacts of the MDA development life cycle are

CIM, PIM, PSM and code [9].

B. MDA Models

Within the MDA approach, everything is considered as being a model, even if it is a diagram or a source code. These models are refined, detailed, and specialised throughout the whole MDA development process. MDA, in fact, has several types of models and we will now go on to have a detailed look at each of them [8]:

* CIM, PIM, PSM and Code

As a result, the MDA development life cycle is based on the following ideas:

The first artefact defined in this life cycle is the CIM, which contains the enterprise and software requirements. This model is then used as an input for the analysis phases, which produce a PIM as their output. The PIM, which describes the system without taking into account the platform of implementation, is transformed into a PSM through the design phase. The PSM, which contains details about the platform of implementation, is transformed into a source code during the coding phase.

C. The MDA Framework

Models: MDA is based on models and their transformation. Such models are written in a well-defined language, which constitutes the main In MDA, there is no restriction for models as regards the way they should look (diagram or syntax), provided that they are well defined. Because of this, the source code is also considered to be a model of the software.

Main MDA models: The MDA approach is based on four levels of abstraction, during the software development life cycle. For each level, a specified model is used. The first level is based on code, which represents the implementation of the system on the target platform. The second level contains models for a specified implementation or platform. The third level consists of the platform-independent model, which describes the software specification without taking into account any specific platform. The fourth level is based on the computational independent model, which represents a high level of abstraction. This level contains the business requirements from an enterprise point of view.

D. Metamodelling

Metamodel: The MDA approach is based on metamodels as well as models [10]. So, what is a metamodel? A metamodel is a model which defines every kind of element that a modeller can use in his or her model. Because a metamodel is considered to be a model, it must be written in a well-defined language. Such a language is called a metalanguage [7].A metamodel is a model that describes a system is written in a specific language. This language is defined by a metamodel, which is in turn written in a metalanguage. At this point, we come to the following definition for a metamodel [11]: A metamodel is a model that formally defines the syntax and semantics of a particular domain-specific modelling environment.

The OMG's four Metamodelling layers: Metamodelling means defining concepts that will be used and applied to model systems. It is also the activity of defining metamodels through their related entities and the relations between them. In this context, the OMG has defined a metamodelling architecture based on four modelling layers, as illustrated in [12]. This four-level architecture is considered to be a basis for defining modelling, metamodelling, and metametamodelling activities [11]. It also provides a basis for exchanging metamodels among different metamodelling environments. This favours interoperability achievement.

IV. Model Transformations

A. What is a Transformation

Transformation between models is achieved through a transformation definition, which is considered to be a set of transformation rules. In the MDA approach a clear distinction is made between a transformation, a transformation definition and a transformation rule: A transformation is the process of generating a new model (target model) from another model (source model), according to a transformation definition. A transformation definition is a set of transformation rules that describe how a target model will be generated from a source model. A transformation rule is a description of how one or more elements of the source model will be transformed into one or more elements in the target model. A model transformation is like a function in which the input and output parameters are models. A model transformation is achieved through the following two steps:

1) Define the transformation rules, using the mapping rules, between the sources and the target metamodels.
2) Apply these transformation rules to the source model in order to generate the target model.

B. Different types of Transformation

1) CIM to PIM transformation.
2) PIM to PIM transformation.
3) PIM to PSM transformation.
4) PSM to code transformation.

Those transformations are described in [9].

C. Model Transformation Approaches

1) Markings: During this approach [8], a specific platform (target platform) is chosen and the definition of the mapping between the PIM and the target platform is prepared. This includes a set of mappings that will be used to mark model elements in order to guide the transformation. Once they are marked, a PIM is transformed into a PSM through the mapping rules.
2) Metamodel transformation: In this approach, a target platform [8] is chosen and a specification of the transformation is prepared. This specification represents the rules for the mapping between source and target metamodels. This mapping is used to guide the transformation of a PIM into a PSM.
3) Model transformation: According to this approach [8], the PIM is defined using platform-independent types. These types are specified in a model. In addition, the elements of the PIM are subtypes of the platform-independent types. A specification of the transformation is then prepared, which will be used as a mapping between the platform-independent types and the platform-dependent types

4) Pattern application: This approach is an extension of the previous two approaches [8]. It is based on the use of patterns in addition to the platform-independent types. These patterns represent basic transformation primitives, which can be mixed in order to describe transformations that are more complex.

V. The Complete MDA Framework

Once we have defined what metamodelling and model transformation are all about, we can integrate them in order to obtain the complete MDA Framework in order to show the complete model [7]. In [7] it shows how the metamodelling level is integrated with the transformation definition language into the MDA Framework. In most cases, developers will see and use only the basis framework shown in the lower half of this figure, and they will not take into account the additional metamodelling level. Only more experienced developers who want to develop and define transformation between languages will use this level.

Interoperability benefit: Interoperability, [7], as an example in PSM levels, means that PSM's, which are targeted at different platforms, can talk to each other directly. This is achieved by means of bridges, which make communication between PSMs easier. For example, if we have two different PSMs for the same PIM and we want to make them communicate with each other, given the first PSM, we know which element in the PIM it has been transformed from. From the first PIM we also know what the corresponding element is in the second PSM. We have all the information we need to generate a bridge between the two PSMs. This is what interoperability refers to [13]. [13] Shows how a class diagram can be interpreted from two points of view. This class diagram can be interpreted from the persistent point of view, with a database. In addition, it can be interpreted from a business point of view with Java programming language. In this case, the PSM Bridge is useful to make them interoperate with each other.

Productivity benefit: Most enterprises are obliged to raise their productivity through the automation of certain operations, due to the increased competition between them. MDA is in perfect harmony with this necessity [7], since it is based on automatic model transformation and generation of the source code. A real decrease in development time can be obtained by using the MDA approach. Furthermore, during the MDA development life cycle, developers are more focused on the development of the PIMs and their transformation definitions instead of programming tasks. Thus, the payback for the effort required to define these transformations is large, because they can be applied in the development of many systems. This makes it possible to achieve improved functionality in less time.

Portability benefit: Within the MDA approach [7], the application systems developed are completely portable because these applications are based on the development of PIMs, which are by definition platform independent. This means that every system specified at the PIM level is therefore completely portable. Consequently, new technologies that appear are easily deployed using the existing PIMs.

VI. Conclusion and Future Work

In this study, we were mainly interested in the area of Enterprise Modelling and achievement of Interoperability using existing Software Development tools and methodologies. For this reason, we started our work by understanding and improving both the GRAI and the UML metamodels, which were to be used as source and target platforms for performing model transformations. After that, we defined a number of mapping rules in order to transform source elements into target elements.

So the main goal of any enterprise, however small or large it may be, is to improve customer satisfaction and competitiveness and, at the same time, to reduce the throughput times by making improvements to the quality of their processes. To fulfil this goal these companies have an increasing need to model, by using enterprise modelling methods, and implement their functions, operations, and business processes in a computerised information system. In recent years, the modelling of enterprise functions and business processes has become a well-known technique in a number of companies [1]. Moreover, these companies, especially the large ones, want to make their different information systems and applications communicate and interoperate with each other. This consequently allows them to save time and money by reducing the time and costs deriving from switching from an existing platform to a new one.

To come up with a solution to this problem, several modelling methods and tools have been created and used in enterprises. Our future work will be to examine these solutions in detail from both the methodological and technical (experimental) point of views:

We then used two different programming tools to define and implement some transformation rules based on model transformations. This experimentation with existing tools gave us a true idea about real problems that have to be overcome in order to achieve interoperability between systems based on different platforms. During our research project, we also succeeded in producing several added values related to this subject at different levels:

References

[1] R. Grangel and J. P. Bourey, "Deliverable D7.1 State of the art of methods to derive IT specifications from models and to develop IT specific components based on IT modelling."

[2] F. B. Vernadat, *Enterprise Modeling and Integration: Principles and Applications*, Chapman and Hall, 1996.

[3] External Extended Enterprise MEthodology Project (2004). [Online]. Available: http://research.dnv.com/external/default.htm

[4] Athena Project, First Version of State of the Art in Enterprise Modelling Techniques and Technologies to Support Enterprise Interoperability Work package A1.1

[5] Department of Computer and Electrical Engineering and Computer Science. [Online]. Available: http://www.cse.fau.edu/~maria/COURSES/CEN4010-SE/C10/10-7.html.

[6] UEML: Unified Enterprise Modelling Language Project (2004). [Online]. Available: http://www.ueml.org
[7] Object Management Group. [Online]. Available: http://www.omg.org.
[8] A. Kleppe, J. Warmer, and W. Bast, *MDA Explained, the Model Driven Architecture: Practice and Promise*
[9] Object Management Group, Model Driven Architecture, MDA Guide. [Online]. Available: (June 2003). http://www.omg.org/mda.
[10] A. Mellor *et al.*, *Introduction to Model Driven Architecture*, Addison-Wesley, 2004.
[11] Institute for software Integrated systems. [Online]. Available: http//www.isis.vanderbilt.edu/publications/archive/Nordstrom_GG_3_0_1999_Metamodeli.pdf#search=%22metamodeling%22.
[12] K. Hubert, *MDA*, DUNOD, Paris, 2005.
[13] G. Doumeingts and D. Chen, "Interoperability development for enterprise applications and software," *Building the Knowledge Economy: Issues, Applications*, 2003.

Hatem Ben Sta received his PhD, from Ecole Central of Paris, France in 2006.He hold a diploma of engineer in computer science. He is now a researcher at the LI3 Tunisia University Research Laboratory, where he preparing his Habilitation for directing research. He also a researcher at the Laboratory of Computer security in the Computer Science Department of Laval University, Canada. His also now the head of the Department of Information System and Software Engineering at Higher Institute of Computer Science, University of Tunis at El Manar. His research interests lie in enterprise modeling and integration, business process re-engineering (BPR), knowledge management, Ontology's and Semantic Web, Software Engineering and decision support systems. He has published more than 20 papers in journals and at international conferences.