27

DEVELOPING AN UNIFIED ENTERPRISE MODELLING LANGUAGE (UEML) – REQUIREMENTS AND ROADMAP

David Chen[1], Bruno Vallespir[1] and Guy Doumeingts[1, 2]
(1) LAP/GRAI, UMR CNRS 5131, University Bordeaux 1 - ENSEIRB
351 Cours de la Libération, 33405 Talence FRANCE
{Chen, Vallespir}@lap.u-bordeaux.fr
(2) GRAISOFT, 33170 Gradignan, France, gdoumeingts@graisoft.com

In the area of Enterprise modelling and engineering, a research subject has been initiated to develop the UEML (Unified Enterprise Modelling Language). This paper starts by introducing the background information on the UEML development. A problem statement is presented and possible benefits expected from the UEML discussed. Then the paper tentatively presents a view on necessary functionality that an UEML must provide and a possible roadmap to follow to reach that goal.

1. INTRODUCTION

This paper intends to provide a view on the development of an UEML (Unified Enterprise Modelling Language). It is based on some preliminary discussions taken place within the French GRP (Groupement de Recherche en Productique) sub-group 5 on Enterprise Modelling and the IFAC-IFIP Task Force Interest group on UEML (Vernadat, 1999) (Vallespir *et al.*, 2001).

Prior to these developments, some earlier initiatives such as for example the International Conference ICEIMT'97, has identified the need for UEML and proposed a development framework referred as 'The layered approach to UEML design' (Petit *et al.*, 1997). The European Standardisation Committee (CEN) has also related the review of an experimental standard on enterprise modelling constructs 'ENV 12204' (ENV 12204, 1995) to potential development on UEML.

The acronym UEML was inspired by UML (Unified object-oriented Modelling Language) proposed by Rational Software Corp. UML can be seen as an extension and improvement of the Entity-Relationship formalism. It was, at the origin, developed for software engineering (in particularly for information systems) and not well adapted for enterprise modelling (there have been some attempts but they do not come up with a well formed metamodel). The development of an UEML in the area of Enterprise Integration and Engineering is similar to what has been achieved in developing UML in the domain of software engineering.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35585-6_68

2. PROBLEM STATEMENT AND BACKGROUND

Since the first development in the area of enterprise modelling started in the US in the years of 70's (ex. SADT, SSAD, IDEF0, Data Flow Diagram,...), a lot of enterprise modelling languages have been elaborated world-wide. We can mention for example, Entity-Relationship model, MERISE, NIAM, M*, GRAI grid and nets, CIMOSA constructs and building blocks, OMT, IEM, ARIS method, IDEF3,... It is generally recognised that there are too many heterogeneous modelling languages available in the 'Market' and it is difficult for business users to understand and choose a suitable one. "Moreover each of these languages has its own syntax be it textural or graphical. More importantly, although they have a clearly defined syntax, most languages do not have a clearly defined semantics" (Petit *et al.*, 1997).

However, this situation can be explained by :
(1) various theoretical basis upon which these modelling languages were elaborated. For example the CIMOSA constructs were mainly developed by people with a background of computer science while the GRAI decisional approach (Doumeingts *et al.*, 1998) is based on system theory and control theory as well as production management theory.
(2) specific application areas. For example, MERISE and M* were developed specifically for designing information systems while IDEF3 is for business process modelling and reengineering.

Main problems related to this situation are:
- Difficulties (impossibility in some case) to translate one model built using a language to a model expressed in another one;
- Difficulties for an enterprise to use a software tool if it is based on languages which are different from the ones adopted by the enterprise.

However, it seems that concepts behind these various languages are similar or slightly differ in details.

The expected UEML would not be a new language to replace existing ones that are currently used but rather, capable of interpreting them. In other words, the UEML has no vocation to be integrated in a "tool box of analysts" so it is not constrained by the criteria of user friendly and operational usability (Vallespir *et al.*, 2001). The possible output of the UEML development is likely a language which is compatible to operational languages widely used such as IDEF, GRAI, Entity-Relationship etc..
Vernadat considers that an UEML could be an "Esperanto" in the area of enterprise modelling and enterprise engineering. It is not the ultimate EM language to replace all previous ones but a standard meta-model (and underlying ontologies) widely accepted by business users and tool developers. It will be easy to learn and to use with sufficient descriptive capabilities (Vernadat, 1999, 2001).

It is generally agreed that the development of an UEML will contribute to:
- a clearly definition of the common semantics of formalisms, and better delimit the domain of enterprise modelling and engineering;
- a better interoperability and communication between modelling agents in a heterogeneous environment;

- a better definition of scientific corpus of enterprise modelling and engineering and thus increase its visibility within the scientific community;
- a generally accepted vocabulary to be used by the standardisation bodies at various levels (national, European and ISO) in the relevant domain.

3. REQUIREMENTS

An UEML can be viewed as a core interfacing with various enterprise engineering and operation users. Expected functional requirements are tentatively stated as follows (also see figure 1):

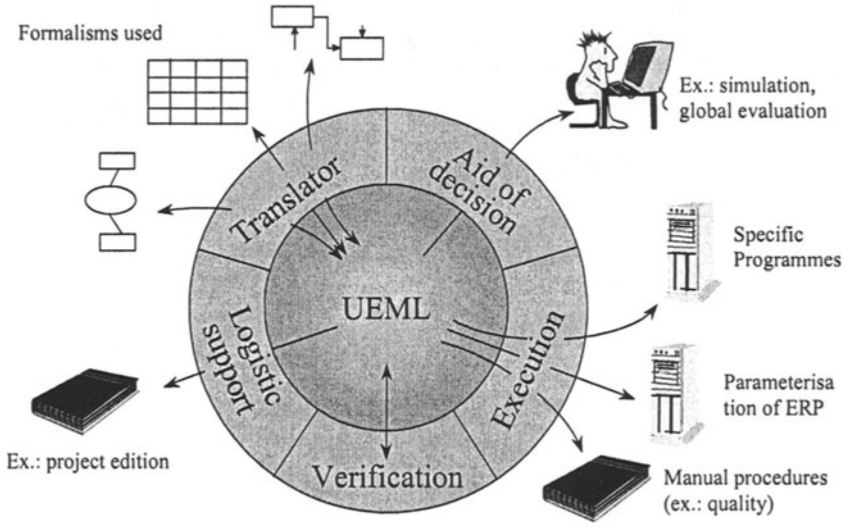
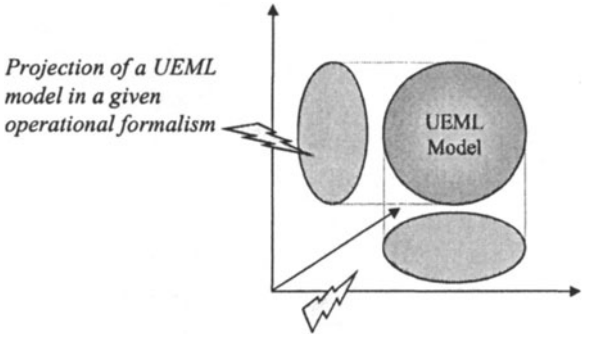


Figure 1 - Potential functionality of the UEML

(1) Translator: UEML plays a role of "pivot" and avoids the one-to-one translation. This functionality can be compared within the role played by STEP in the area of product description data.
(2) Aid of decision: Data and information needed to support decision-making will be stored in the UEML format so that they can be used by any software decision support tool.
(3) Project edition: Like IDEF0/SADT that was used in the 70's by the US Air Force to describe and define projects, UEML can be used to edit project description so that it can inter-operate with other software tools.
(4) Formal verification: Like Petri nets, UEML with its formal definition of syntactic and semantics can support property verification of models.
(5) Execution: Acted as a neutral model, UEML can support not only the execution of enterprise model for process control, but also be used for parameterisation of production management software such as MRPII or ERP as well as for quality procedure edition, etc.

It is also considered in (Petit *et al.*, 1997) that UEML must have a clearly defined graphical and textual syntax (grammar) so that models could be exchanged among tools. Moreover, in order to overcome the complexity facing the important number of concepts used in enterprise integration and engineering, a layered approach could be an adequate answer. A core of UEML will contain the minimal set of constructs necessary for modelling of any enterprise whereas a set of libraries built on top of this core could contain additional, more expressive concepts, possibly specialised for different application domains. To define formally semantics of UEML, some theories can be used such as situation calculus, state-transition diagrams, temporal logic, process algebra, first order logic etc.

The precise definition of functional requirements allows to identify modelling concepts/constructs contained in the UEML core. The core interacts with external users via various interfaces as shown in figure 1. In this sense, the UEML core can be seen as a generic basis. Various existing modelling languages are operational interfaces (i.e. the projections of UEML, see figure 2). This means implicitly that UEML has a larger modelling coverage than any individual existing one. It is the Union of existing languages.



Plan corresponding to a given operational formalism

Figure 2 - Position of the UEML (Projection)

We can remark that the same logic applies to the notion of view developed for example in the CIMOSA approach. The UEML model contains in its neutral format all necessary constructs/concepts for the description of enterprise functions, decisions, processes, activities, resources and behaviours with all information/data needed for its engineering and operations. A view (for example function view) is only a projection of that UEML model according to the selected viewpoint.

4. THE TRANSLATION ISSUE

Among the various functionality mentioned previously, we will show a simplified example to illustrate the problems and difficulties for the functionality 'Translation'. Let us assume that there were only two existing languages available in the world: SADT activity and GRAI activity as shown in figure 3. Each activity formalism uses some concepts: SADT Activity (construct) = {Name, Number, Input, Output, Control, Mechanism}, GRAI Activity (construct) = {Name, Number, Trigger, output, Support}. If one compares the two formalisms, one can conclude that the term 'Activity' exists in both formalisms but it is not formally a common concept.

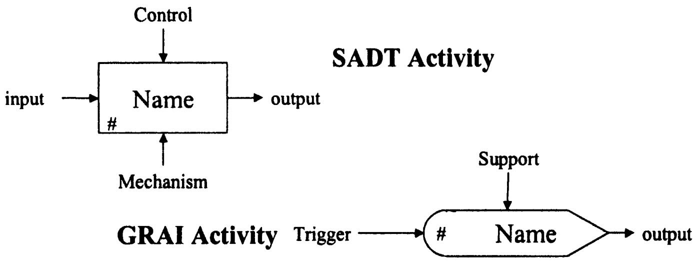


Figure 3 - A simplified translation example

The search for common aspects of the two formalisms leads to the identification of common attributes as shown in figure 4. They are: Activity.min = {Name, Number, Output}. The second conclusion is that 'Activity' is not an elementary concept. An elementary concept is a concept that will not be decomposed when elaborating a model.

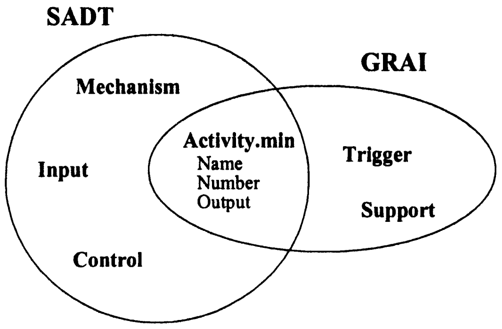


Figure 4 - Search for common concepts

Suppose that an UEML core contains all the concepts identified above. Having applied union, these concepts are organised in a way as represented by figure 5. An Input, a Control, a Mechanism could be a Support. Now let us consider the translation from a SADT model to the GRAI Activity model.

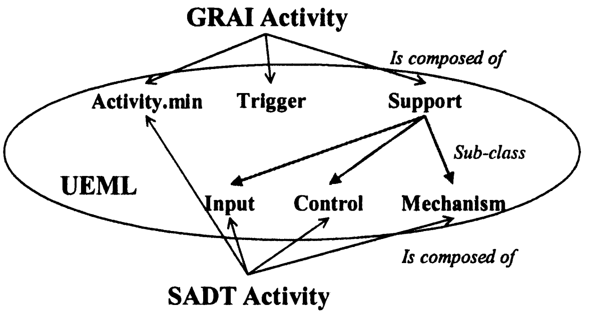


Figure 5 - Simplified illustration of a UEML core for SADT and GRAI Activities

The information is given in a SADT format as shown in figure 6 and the translation is performed via the UEML core as defined in figure 5.

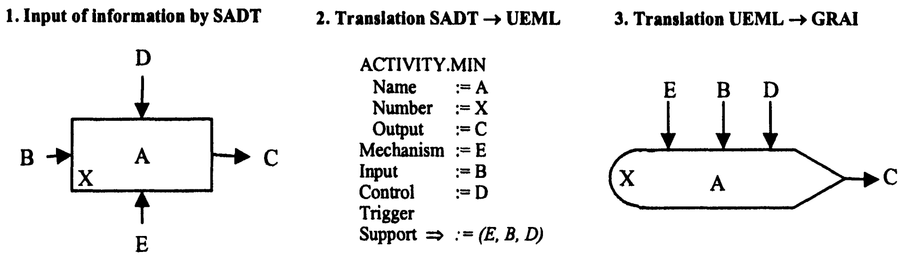


Figure 6 - Translation from a SADT Activity to the GRAI Activity

The result is syntactically correct because 'Trigger' is optional in GRAI net formalism. The GRAI activity has three supports E, B and D.

Now let us see the inverse process to translate a GRAI Activity into the SADT Activity as shown in figure 7. One can find that "B" is eligible to be Input, or Control, or Mechanism. Suppose that the choice is 'Control' than the "B" of GRAI becomes the Control of SADT. This is syntactically correct because the input and mechanisms are optional in the SADT model. However, "D" does not appear in SADT activity but remains in the UEML model (projection).

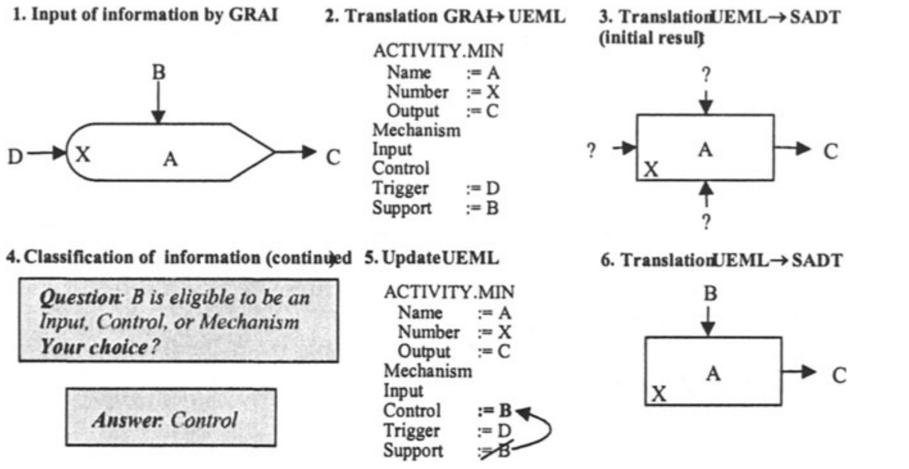


Figure 7 - Translation from a GRAI Activity to the SADT Activity

This simplified example makes appear some specific points concerning the translation process:

- Problem of definition of elementary concepts,
- Concepts can be classes of other concepts,
- There is a possibility that some pieces of information are not translated from an UEML model to another one: projection.

5. ROADMAP

Several approaches are possible to define a roadmap for developing an UEML: top-down, bottom-up and combined approaches (bottom-up for development and top-down for consistency).

The bottom-up approach starts with an analysis and then synthesis of existing enterprise modelling languages. The approach is structured in four steps as follow:

(1) Choice of existing modelling languages. It consists in carrying out a complete state-of-the art study and identifying eligible formalisms among all available ones in the domain. A representative set of formalisms will be selected according to established criteria (such as usability, reconnaissance, etc.).
(2) Decomposition of each chosen formalism to elementary concepts as shown in the illustration example.
(3) Union of all elementary concepts.
(4) Fusion to establish syntactic

The advantages of the bottom-up approach are: (i) more rapid, (ii) avoid to "reinvent the wheel". The inconvenience is that there is nothing to ensure that chosen formalisms are representative.

The top-down approach is an analysis approach. It is also structured in four steps:

(1) Precise definition of requirements and domain.
(2) Choice of a theoretical paradigm. For example: System theory
(3) Definition of elementary concepts
(4) Establishment of the syntactic between concepts

The advantage of the top-down approach is the theoretical consistency (inconsistency is often undetectable a priori). The shortcomings are: (i) it could take a long time to develop, (ii) rework of some existing formalisms.

To keep the advantages of top-down and bottom-up approaches and avoid their shortcomings, a hybrid approach is proposed as shown in figure 8.

This approach starts by:

(1) Define precisely functional requirements (functionality) that UEML must provide.
(2) The choice of existing languages will be done in consistency with required functionality. At the same time the choice of a theoretical paradigm (2bis) should allow to ensure some theoretical consistency.
(3) Then the set of elementary concepts can be obtained by decomposing chosen languages.
(4) The union of elementary concepts allows to remove possible redundancy.
(5) Finally the syntactic will be defined to establish relationships between the concepts.

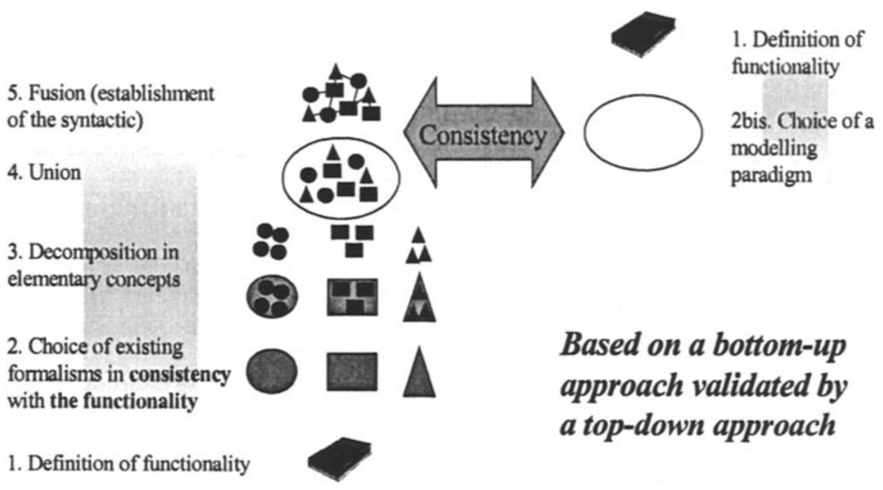

5. Fusion (establishment of the syntactic)
4. Union
3. Decomposition in elementary concepts
2. Choice of existing formalisms in consistency with the functionality
1. Definition of functionality
Consistency
1. Definition of functionality
2bis. Choice of a modelling paradigm
Based on a bottom-up approach validated by a top-down approach

Figure 8 - The proposed roadmap to develop the UEML

6. CONCLUSIONS

This paper presented a view on the development of an UEML taking into account the state-of-the art in the domain. The starting point of a such project is to define functionality that the UEML must provide. The paper has tentatively identified a set of functional requirements. It has been also discussed that neither bottom-up nor top-down approaches can be an ideal methodological process to follow. As the consequence, a roadmap that combines both top-down and bottom-up approaches has been proposed.

7. REFERENCES

1. Doumeingts, G., Vallespir, B. and Chen, D. "Decision modelling GRAI grid". In Handbook on architecture for Information Systems, Peter Bernus, Kai Mertins, Gunter Schmidt, ed. Springer, 1998.
2. ENV 12204. Advanced Manufacturing Technology - Systems Architecture - Constructs for Enterprise Modelling, CEN TC310/WG1, 1995.
3. Petit, M. (Ed.), Goossenaerts, J., Gruninger, M., Nell, J.N. and Vernadat, F. Formal Semantics of Enterprise Models. Proceedings of ICEIMT'97 (edited by K. Kosanke and J. Nell), International Conference on Enterprise Integration and Modelling Technology, Springer, 1997.
4. Vallespir, B., Doumeingts, G. and Chen, D. Problems and Research orientation for UEML: A point of view, Slides presentation at the meeting of the IFAC-IFIP Task force Interest group on UEML, Vienna, September 19, 2001.
5. Vernadat, F. Unified Enterprise Modelling Language (UEML), Slides presentation at the meeting of the IFAC-IFIP Task force Interest group on UEML, Paris, December 16, 1999.
6. Vernadat, F. UEML: Towards a Unified Enterprise Modelling Language. Proceedings of 3^e Conférence Francophone de Modélisation et Simulation (MOSIM'01), Troyes, France, 25-27 Avril 2001.