

Some Methodological Clues for Defining a Unified Enterprise Modelling Language

Michaël Petit

University of Namur, Belgium, mpe@info.fundp.ac.be

Abstract The need for a Unified Enterprise Modelling Language (UEML) that would be used as in inter-lingua among enterprise modelling software tools has been established. The process of defining this UEML goes through the elaboration of a precise meta-model describing the constructs of the language. A possible and reasonable approach for the definition of this meta-model is to integrate (parts of) meta-models of existing enterprise modelling languages. This approach has similarities with the well studied problem of databases integration in which the models of several databases have to be integrated into a single one. In this paper, we make an analogy between the two problems and review a state of the art methodology proposed in database integration to derive methodological clues for the definition of the meta-model of a UEML.

1. TOWARDS A UNIFIED EML

Enterprise modelling (EM) has long been recognised as a valuable activity (Vernadat, 1996). However, the current situation in this domain prevents us from getting the most benefits from EM. Many enterprise modelling languages (EMLs) exist and offer similar but slightly different constructs for modelling. A model written in a particular language can rarely be understood by people not familiar with this language. The different language supporting tools offer different interesting functionalities, but the absence of common understanding of the models by the different tools prevents the exchange of models created by each tool, disabling the user from using these functionalities in a model without rewriting it completely in another language.

As an enterprise is a complex entity and because modelling needs are diverse, several models are often produced with different languages but no

integrated model is available that could provide a coherent and complete view of the enterprise.

This situation has led to the identification of the need for an inter-lingua for enterprise modelling (Goossenaerts, *et al.*, 1997). Such a UEML (Unified Enterprise Modelling Language) could be used to exchange models among tools and would constitute a base to commonly understand models written in different languages, providing the possibility for an integrated model of the enterprise. Some efforts have already been made to define such a language and have resulted in the European pre-standard ENV 12204 (CEN, 1996). Such a need was also identified by other projects such as PSL and projects on ontologies (Schlenhof, *et al.*, 2000, Fox, 1992, Gruber, 1993).

One of the important steps in defining a language is to elaborate a meta-model. The meta-model corresponds to an abstract syntax and describes the constructs of the language, their properties and restrictions on the way their instances they can be combined to form models. It usually serves directly to define a concrete syntax (textual or graphical) and is the basis for implementing repositories of software tools.

This paper proposes methodological guidelines for the definition of the meta-model of a UEML. In particular, some research work from the field of database integration is proposed as an inspiration for defining methodological clues. As some issues in this field are similar to the ones to be solved to define UEML, we propose that lessons learned and tools in the database integration field are reused. This paper is an attempt to make an analogy between these two research fields.

2. THE DATABASE INTEGRATION PROBLEM

The database integration problem arises when several databases exist and contain overlapping or related data, potentially implemented in heterogeneous environments. The problem in these situations is to provide a mechanism for accessing these databases that hides the location of data and provides a seamless access to logically related data present in more than one database. Parent and Spaccapietra, (2000) define database integration as “the process which:

- takes as input a set of databases (schema and population), and
- produces as output a single unified description of the input schemas (the integrated schema) and the associated mapping information supporting integrated access to existing data through the integrated schema.”

As an example, consider a situation where some information about persons (such as e.g. name and birth date) is stored in a database made of Cobol

files and other information (such as name and address) is stored in another SQL database. Integrating these databases would be necessary to answer queries such as “list persons aged more than 50 living in Brussels”.

3. AN ANALOGY BETWEEN UEML META-MODELLING AND THE DATABASE INTEGRATION PROBLEM

UEML should be defined on the basis of the set of existing EMLs. Therefore, it should include constructs that relate as closely as possible to those of the majority of currently used EMLs. UEML should thus be an integration of most existing EMLs. The definition of UEML should therefore be made by examining different EMLs and systematically considering the inclusion of their constructs in UEML. The definition of a meta-model for UEML can be in some respects compared to a database integration problem in which:

- the database schemas to be integrated are the meta-models of a set of candidate EMLs;
- the data on which an integrated vision is desired are the various enterprise models created in these different EMLs.

In Parent and Spaccapietra, (2000) a general methodology for database integration is proposed on the basis of a review of current approaches and solutions to the database integration problem. It consists of three major steps:

1. Preparation for integration
2. Investigation and definition of correspondences
3. Integration

In the sequel of this section, we describe the activities that have to be performed in each of these steps, make an analogy with the UEML definition problem and show the possible implications of the methodology for the definition of UEML.

3.1 Step 1: Preparation for integration

3.1.1 Preparing for the integration of databases

When facing a database integration problem, a natural first activity consists of collecting information about the databases to be integrated. One of the most important pieces of information about a database is the description of its content in terms of classes of data that it may contain. This description is referred to as the database schema (or database model). The schemas of all

databases to be integrated therefore have to be collected. If they are not available, they have to be defined. If only a technological description of the database is available, a reverse engineering process is necessary to obtain a conceptual schema useful for integration at the conceptual level. An example of a software tool that supports this reengineering activity is described in (Hainaut, *et al.*, 2000).

When integrating heterogeneous databases (databases built using different technologies such as files, SQL databases, Object-oriented databases, ...), the schemas are usually of different nature or quality. Additional treatment of the schemas is therefore often necessary to reduce discrepancies among them so that the schemas can be integrated more easily. Three kinds of modifications to the schemas are described by Parent and Spaccapietra, (2000): syntactic rewriting, semantic enrichment and representation normalisation.

First, a *syntactic rewriting* of the schemas might be necessary if the schemas are initially expressed in different languages. If one schema is expressed e.g. using an entity-relationship notation and another is expressed using UML class diagrams, the comparison of the elements of the schemas will be more difficult than if a single language was used. This requires choosing a common language to express the schemas. According to Parent and Spaccapietra, (2000) the chosen language must be rich enough to allow the expression of all information relevant to the different schemas, but must not be too rich to avoid too many modelling alternatives when defining the schema. The later problem, known as semantics relativism, calls for languages with minimal semantics embedded (no complex modelling elements).

Second, the schemas to be integrated might require *semantic enrichment*. This means adding semantic information to the schema such as initially un-stated constraints, implicit assumptions,...

Finally, though the choice of an adequate common representation language reduces the semantics relativism problem, it is probably not possible to avoid it completely. Therefore, the schemas might require a *representation normalisation* that imposes the use of consistent representational strategies within the schema when choices are possible. For example, if an attribute of a class in a schema may have many values for each object of that class, such a strategy might be to represent this attribute as a separate class linked to the initial class with a relationship.

3.1.2 Preparing for the definition of UEMML

Similarly to the database integration problem, the UEMML definition requires the collection and definition of the schemas to be integrated. In this case, the schemas are the meta-models of a set of candidate enterprise mod-

elling languages considered for “integration” in UEML. Meta-models of EMLs are usually not directly available. Most EMLs are described in terms of their syntax, but do not always make explicit all relationships among language constructs and constraints that apply to obtain valid models written in those languages. A first necessary exercise is therefore to define precisely the meta-models of these candidate languages. In some cases, the meta-model can be obtained by reverse engineering the meta-models implemented in supporting software tools that use database technologies to store enterprise models in a repository. In other cases, the meta-models have to be elaborated by hand on the basis of the literature describing the EML.

In previous work, we have done the exercise of defining such meta-models. To express these meta-models we used a common language called Telos (Mylopoulos, *et al.*, 1990). The choice of Telos is justified by its formality (mathematical foundation), expressiveness (especially for expressing constraints) and limited number of concepts (preventing too many choices of representation). The first meta-model we have defined is the one of CIMOSA, a well known EML, which is the result of a European project (AMICE, 1993). The complete meta-model can be found in (Petit, 1999). The second one is the meta-model on the ENV-12204 pre-standard (CEN, 1996). This pre-standard could be considered as a initial attempt of a UEML definition. The definition of the meta-model on the basis of the published standard raised a number of problems and open issue that are reported in (Férier, *et al.*, 2000). In both cases, we have applied implicitly semantic enrichment and representation normalisation. Based on our understanding of the studied documents, we have added constraints and resolved perceived inconsistencies. Representation strategies, while not explicit, were usually applied because the models were elaborated by a small number of people.

Nowadays, class diagrams from the UML (Booch, *et al.*, 1999) are often proposed for defining meta-models of languages. Compared to Telos, UML offers more representational choices and may therefore make meta-models more difficult to compare and integrate. However, it might be more intuitive because it is less formal. Note however that deriving a meta-model expressed in UML from the Telos descriptions is quite easy. We are currently performing some preliminary work on the definition of a meta-model expressed in UML for the Workflow Process Description Language (WPDL) defined by the Workflow Management Coalition, (1999).

For the definition of UEML, a more systematic way of working would be needed by:

- Making explicit the representation strategies used to define the meta-model, and systematically apply these strategies;
- Validating the meta-model through interaction with the language designers or owners and validating the semantic enrichments applied.

3.2 Step 2: Investigation and definition of correspondences

3.2.1 Investigation and definition of correspondences in databases

The next step in the methodology consists in establishing what is common in the databases that are candidates for integration. This amounts for investigating and establishing correspondences among these databases. Parent and Spaccapietra, (2000) explains that the correspondences have to be defined at two levels. *At the data level* (population of the database), the correspondences among instances of classes present in one database and the ones present in another have to be identified. To establish these correspondences, the semantics of the instances have to match. Two instances are considered to have the same semantics if they describe the same real world element. *At the schema level*, correspondences among classes are established if the correspondences among instances apply to a significant set of instances of these classes. The correspondence is thus generalised at the class (schema) level. The correspondence may be further characterised as an equivalence (if both classes have sets of instances that represent exactly the same set of real world elements), as an inclusion (if all instances of one class have a corresponding instance in the set of instances of the other), as an intersection (if there exists correspondences among instances but no equivalence and no inclusion), ... Furthermore, a notation for describing these correspondences is proposed.

3.2.2 Investigation and definition of correspondences between EMLs

For the definition of UEML, we need to establish correspondences among the classes defined in the different meta-models of the EMLs candidate for being integrated in UEML. But as Parent and Spaccapietra, (2000) suggests, we may only define correspondences at this level if the correspondences among instances of these classes can be generalised. This means that we first have to investigate correspondences among models elements created with the different EMLs and then generalise them if these model elements have the same semantics. In our case, the semantics of model elements is more complicated than the one of database instances because enterprise models usually represent sets of elements or happenings of the real world rather than individual elements. For example, a model element such as an object class actually represents a whole population of a database, whose elements themselves have a correspondence in the real world. A correspon-

dence among languages can therefore only be established if model elements created with these languages represent the same set of elements from the real world. The same principle applies to the comparison of the semantics of behavioural models elements such as processes. In this case, the semantics is even more complex to compare since processes have a dynamics semantics, potentially representing infinite sets of behaviours. The semantics comparison must in this case make sure that the sets of process behaviours described by both models are corresponding. This comparison can become possible and be computer-assisted if the semantics of process models is defined formally. Further research would however be needed to allow this kind of automation.

A consequence for the UEML definition process is that to establish correspondences among language constructs (classes of EMLs meta-models), we need models created using these languages. Therefore, case studies have to be carried out in which a single reality is modelled with these different languages. Then correspondences among the obtained models have to be established by comparing the semantics of the obtained model elements in terms of the sets of real world elements or behaviours they represent. On the basis of these correspondences, tentative correspondences may be defined at the language level, among elements of the meta-models. These correspondences can then be validated or infirmed on the basis of further case studies.

A notation for explicitly defining correspondences both at the model and language levels is therefore necessary. In (Petit, 1999), we have proposed a structure corresponding to this notation for a framework made of several languages for Manufacturing Systems modelling. This structure, formalised in Telos, is based on the meta-models of the languages of the framework. Potential correspondences at the language level are defined as “mapping rules” among language constructs, whereas correspondences at the model level are seen as applications (instances) of the mapping rules and establish correspondences among model elements. It should be noted that in our case, the mapping rules were not meant to be general nor automatic, so that the models creator can decide if the rule applies or not on particular elements of the models. This allows for the definition of different kinds of correspondences as described in (Parent and Spaccapietra, 2000) (equivalence, inclusion, intersection,...).

3.3 Step 3: Integration

3.3.1 Integration of the databases schemas

The third and last step is to define the integrated schema of the database by considering the inclusion of elements from the original schemas in the integrated schema. This process treats systematically each correspondence identified in step 2 and decides which elements to include in the integrated schema. Potential conflicts have to be resolved at this level. A simple example of conflict is a “description conflict” which occurs when two corresponding classes have different sets of attributes. In this case the conflict has to be resolved by deciding which of these attributes have to be included in the integrated schema. Parent and Spaccapietra, (2000) provides a list of possible conflicts and references to literature that propose systematic ways of solving these conflicts. When solving conflicts, different strategies can be adopted depending on the objective followed when doing the integration. For example, some strategies may seek completeness of the integrated schema while others may seek simplicity. Parent and Spaccapietra, (2000) therefore insists on the importance of defining the objective of the integration beforehand and adopting an adequate strategy.

3.3.2 Integration of EML meta-models into one UEML meta-model

The meta-model of UEML should be defined by considering the integration of a number of relevant EMLs meta-models. In some respects, the ENV12204 is already such integrated meta-model. It was defined mainly on the basis of CIMOSA and IEM (Mertins, Jochem, 1999). However, the meta-models of these languages were not made explicit in a single representation language and the correspondences among these meta-models were not described explicitly. In this process, the conflicts that arose were solved intuitively without being made explicit. This has the drawback that no explicit trace has been kept of the relationship existing between the constructs of the original languages and those present in the integrated ENV12204 meta-model.

The strategy for resolving conflicts during integration has to be defined and depends on the objective of UEML. A reasonable objective of UEML could be the interoperability of a set of existing enterprise modelling tools. In this case, an adequate global strategy would be to only include a construct in the UEML meta-model if there exist corresponding constructs in the meta-models of at least two languages to be integrated. Hence, if a construct or attribute is specific to a tool, it would not be useful to make it available to

the other tools, since it would not make any sense for the other tool. If the objective is rather to obtain a logically integrated model of the enterprise, a strategy where more completeness of the UEML meta-model is sought would be more appropriate.

In any case, an explicit definition of the correspondences, both among constructs of the original languages meta-models and between these constructs and the corresponding constructs in the UEML meta-model is useful. The approach of mapping rules proposed above seems adequate for this. The explicit definition of the correspondences is important not only for explaining the constructs of the UEML by making reference to the constructs of other existing and known languages, but also because they are the basis for e.g. a specification of model exchange and translation mechanisms to be implemented in EM tools (in the case of a “model exchange among tools” scenario) or specification of mechanisms for query processing on a logically integrated enterprise model (in the case of an “integrated enterprise model” scenario).

4. ADDITIONAL METHODOLOGICAL HINTS FOR THE DEFINITION OF UEML

A very important step, as discussed in section 3.3 is the definition of the objective of UEML at the very beginning. The meta-models integration strategy and the content of the UEML itself will depend on the settled objective.

A second step is the identification of candidate languages to be “integrated” into UEML. A strong candidate is the ENV12204 pre-standard. This language is currently being substantially revised by CEN TC310 WG1. One of the changes is an explicit and better definition of its meta-model. This new meta-model could serve as a base for the UEML meta-model and be augmented or improved by considering other languages.

A good way of working could be to work with languages on a pair wise basis, rather than considering them altogether.

To remove complexity, the integration process could be first performed on subsets of the considered languages. These subsets could be the core of the languages (set of simple or atomic constructs). After integrating these core constructs, additional composite or complex constructs could be considered for integration.

Software tools should be used whenever possible to support the definition of the UEML meta-model. Some of the tools used in the database engineering and database integration area seem appropriate for this.

5. CONCLUSION

In this paper we have investigated the analogy between the problem of defining the meta-model of a Unified Enterprise Modelling Language (UEML) and the problem of database integration. Some commonalities exist between the two problems. Based on a general methodology proposed by Parent and Spaccapietra, (2000), some methodological hints for the definition of the meta-model of UEML are identified. Some initial work of the authors is also cited.

As the database integration problem has been studied for some time now, a large body of literature describing the issues to be resolved, possible solutions to them and supporting tools to solve them is available. Many of them can be reused within the context of the UEML meta-model definition. This paper is only a preliminary study of the link between these two problems. Additional work is obviously needed to better identify relevant solutions and tools that can be reused for UEML meta-modelling.

6. ACKNOWLEDGEMENTS

The author would like to thank Jean-Luc Hainaut and Philippe Thiran for enlightening discussions on the database integration problem and its similarity to meta-models integration. We also thank the anonymous reviewers for their appropriate suggestions for improvement of the paper and Gaëtan Dellannay for initial proofreading.

7. REFERENCES

- AMICE, (1993), CIMOSA: Open System Architecture for CIM, Springer-Verlag.
- Booch *et al.*. (1999), "The Unified Modeling Language User Guide", Addison-Wesley.
- CEN, (1996), ENV 12204: Advanced Manufacturing Technology - Systems Architecture – Constructs for enterprise modelling, TC310 WG1 (currently under revision).
- Férier, L. Heymans, P. Petit, M. (2000), Some Hints for a Clarification of CEN ENV 12204, Invited paper at the Workshop on Evolution in Enterprise Engineering and Integration, Berlin, May 24-26.
- Fox, M.S. (1992), The TOVE project: Towards a common-sense model of the enterprise. In Proceedings of the International Conference on Object Oriented Manufacturing Systems, Calgary Alberta.
- Goossenaerts, J. Gruninger, M. Nell, J.G. Petit, M. and Vernadat, F.B. (1997), Formal Semantics of Enterprise Models, in K. Kosanke and J.G. Nell (Eds.), Proc. of ICEIMT'97, International Conference on Enterprise Integration and Modeling Technology, Springer-Verlag.
- Gruber, T.R. (1993), A translation approach to portable ontology specifications, Knowledge Acquisition, 5(2).

- Hainaut, J-L. Henrard, J. Hick, J-M. Roland, D. Engelbert, V. (2000), The Nature of Data Reverse Engineering, In Proc. of Data Reverse Engineering Workshop, March 2, as part of Reengineering Week 2000, Zurich, Switzerland, available at http://www.info.fundp.ac.be/~dbm/publication/2000/dre2000_jlh.pdf
- Mertins, K. Jochem, R. (1999), Quality-Oriented Design of Business Processes, Kluwer.
- Mylopoulos, J. Borgida, A. Jarke, M. Koubarakis, M. (1990), Telos: A Language for Representing Knowledge about Information Systems, ACM Transactions on Information Systems, Vol. 8(4).
- Parent C. Spaccapietra, S. (2000), Database Integration: the key to data interoperability, In Papazoglou, M.P. Spaccapietra, S. Tari, Z. (Eds.), Advances in Object-Oriented Data Modeling, MIT Press.
- Petit, M. (1999), Formal Requirements Engineering of Manufacturing Systems: a Multi-formalism and Component-based Approach, PhD Thesis, Computer Science Department, University of Namur, Belgium.
- Schlenoff, C. Gruninger, M. Tissot, F. Valois, J. Lubell, J. Lee, J. (2000) The Process Specification Language (PSL) Overview and Version 1.0, Specification, National Institute of Standards and Technology, Gaithersburg, MD, USA, available at <http://www.mel.nist.gov/psl>.
- Vernadat, F.B. (1996), Enterprise modeling and integration: principles and applications, Chapman & Hall.
- Workflow Management Coalition, (1999), Interface 1: Process Definition Interchange Process Model, Document Number WfMC TC-1016-P, Version 1.1, available at <http://www.wfmc.org/>