Supporting Enterprise Integration through a Unified Enterprise Modeling Language

Giuseppe Berio[1], Víctor Anaya[2], and Ángel Ortiz[2]

[1] Dipartimento di Informatica, Università di Torino, C.so Svizzera 185,10149 Torino, Italy
berio@di.unito.it
[2] Research Center on Production Management and Engineering, Technical University of Valencia, Camino de Vera s/n 46022, Valencia, Spain
{vanaya, aortiz}@cigip.upv.es

Abstract. This paper presents a Unified Enterprise Modeling Language (UEML) and a set of mappings which support translating models represented under an Enterprise Modeling Language into another Enterprise Modeling Language. These mappings also permit to integrate (i.e.compose) different models represented in different enterprise modeling languages into a unique model. Additionally, UEML is a meta-model that represents the (abstract) syntax of a Unified Enterprise Modeling Language (UEML 1.0). This paper presents the foundations of the work performed during the UEML project funded by IST Programme of the European Commission 5th Framework. The rationale, the basic principles and the undertaken approach followed to define UEML 1.0. These basic principles and the approach are justified in terms of data integration and database schema integration.

1 Introduction

In order to reduce costs and lead times, and to improve the quality (and the range) of the offered services or products, enterprises take part in organisational paradigms, for instance, EE (extended enterprise) and VE (virtual enterprise) organisations, where they cooperate by exchanging flows of information & knowledge, material & services, and money ([3]).

Since the 80's, two main approaches to Enterprise Integration (EI) have matured: (i) the integration of *software applications* that support the functionality of business processes; ii) the definition of *Enterprise Models* (EMs) by means of *Enterprise Modelling Languages* (EMLs) and integration of such models, for understanding, validating and effectively achieving EI.

Enterprise modelling is the art of externalising the internal knowledge of an enterprise by the generation of EMs ([9]). Thus, EMs empower both the EI and an enterprise analysis (for instance economic analysis, organisation analysis, qualitative or quantitative time analysis, cost and resource needs). In this sense, enterprise modelling supports the phases and tasks within the context of *methodologies for enterprise engineering and integration*, fundamental components of Enterprise Reference Architectures ([6]) such as CIMOSA ([1], [7]), PERA ([12]), GRAI ([5]).

Enterprises manage and use enterprise models through *Enterprise Modelling Tools* (EMTs). The current landscape shows the existence of a lot of enterprises that used to implement in-house enterprise modelling tools or to buy several tools for supporting different types of analysis and other usage. Each EMT has a proprietary EML, and may be coupled with a proprietary *enterprise modelling process* ([10]).

When two enterprises use different EMLs and they want to exchange EMs there are some problems which facets are ([16]):

- Differences in the (abstract) syntax of the EMLs: a EML used by a company differs *syntactically* from that employed by other enterprises even within the same business domain; for instance, two enterprises that model with IEM ([17]) and EEML ([2]) languages respectively.
- Coverage and expressivity of the EMLs: EMLs have different focus, coping with different purposes.
- Differences in semantics of "similar" constructs: sometimes enterprises use the same constructs but they associate distinct meanings.

On the other hand, whenever the same EML is used with *various and distinct purposes or because of the subjective nature of modelling*, different models represented in an EML are probably difficult to be integrated. Problems arised are:

- Scope: when two artifacts seem to represent the same meaning but this happens only partially
- Coverage and granularity: models seem to embrace the same domain but it is not true.
- Modelling conventions
- Synonym terms: distinct terms share the same meaning.
- Homonym terms: same terms with different meanings depending on the context..
- Encoding: values encoded in different formats (e.g, dates in dd/mm/yy vs mm/dd/yy).

In our opinion, these two dimensions (the first, mostly related to *languages* and the second, mostly related to *models*) have to be approached separately because each one comprises many kinds of complex problems and also because they differ in the scope (very broad for languages because languages should provide constructs for describing varieties of situations in most of enterprises, narrower for models because models usually apply to specific enterprises and situations[1]).

When enterprises pretend to share or combine some of their enterprise models, or whenever various kinds of analysis should be performed on these models, one possi-

[1] We do not consider the problem of "generic reusable models" also called "partial models" in the EM literature.

ble approach is to enable (partial) *translations* of models across distinct languages and tools of the enterprises. These translations should be *model-independent*.

Whenever an enterprise takes part in a network of enterprises the number of coexisting EMLs, is likely to increase. Therefore, *translations* between couples of languages, also called *peer-to-peer translation* (bidirectional arrows in Figure 1), are possible, though difficult and costly to carry out, and not suitable within networks of enterprises where fast changes of partners are usual. The other disadvantage of peer-to-peer translations is related to the "maintenance of a global consistency" due to a clear lack of *global, unique and consistent vision* about the knowledge shared between the enterprises (each EML does not cover all enterprise aspects as it is shown in Figure 1 where three EMTs i.e. MOOGO, eMAGIM and METIS embrace respectively resource monitoring, decision support and enterprise planning).

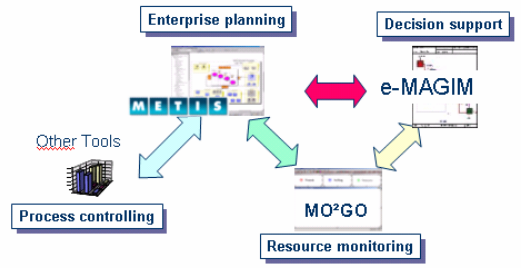

Figure 1. Language-to-language translations

Therefore, to reduce dramatically the number of *interfaces* (peer-to-peer translations) needed to communicate a set of enterprises, and to increase the achievement of a "global consistency", it is really useful to define an *intermediate federator language* (depicted in Figure 2) which eventually allows to represent a *unique, consistent and modular vision* on the shared (or integrated) knowledge of the whole set of enterprises. Thus, such a language, generally called UEML (Unified Enterprise Modelling Language), does not substitute existing EMLs, as its target is to provide effective support for enterprise model translation and integration. In this sense, a UEML should be equipped with *standard translation mechanisms* to and from existing EMLs. Furthermore, this UEML permits enterprises to retain their EMLs without forcing them to use the UEML itself.

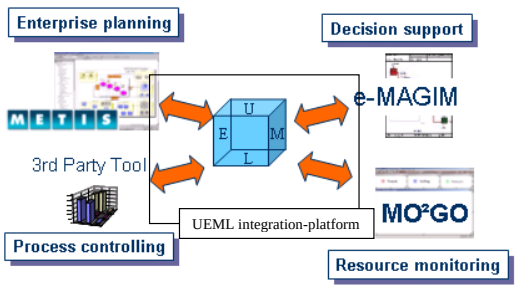

Figure 2. Translations of EMs by using a UEML

However, the UEML is only one of the needed components for effectively achieving model translation and integration. Specifically, this UEML is a way to approach integration problems within the *language dimension*. A list of current and real business problems in which a UEML can play a central role is ([19]):

- Integration of information systems encoding fragmented non sharable enterprise knowledge.
- Achieving a coordination of business processes.
- Achieving a common view in terms of business operations.
- Poor interoperability of process modeling and management tools.
- Insufficient coverage by most languages of required modeling views.
- Diverse visual representations.

2 Towards (a) UEML

How to build effectively such a UEML? There are various answers to this question but some facts need to be taken into account:

a. most of the EMLs are not equipped with formal definition of the semantics of their constructs; therefore it is not obvious (or maybe impossible) to state and to prove (formally) that a language construct is represented (or translated) into a construct belonging to some other language;

b. whenever EMLs are equipped with some formal semantics, the semantic preserving relationships are not unique between distinct languages; therefore, establishing these relationships is difficult as in the previous case (a); the main difference between (a) and (b) is that in (b) it may be possible to prove formally that these relationships are really semantics preserving;

c. whenever some semantic preserving relationships between EMLs have been formally proved, this is not enough for making these semantic preserving relationships as acceptable in term of the modelling domain; in other words, a formal proof of some semantic preserving relationships is not enough for justifying that two distinct constructs are similar/equivalent (or not similar/equivalent) in term of the modelling domain (sometime referred to as "real world semantics"([20])).

Figure 3 below provides two relationships between two EMLs where the construct "Resource" in GRAI ([5]) may be related to constructs "Role" and/or "Resource" in EEML ([2]).

Figure 3. Two EMLs can be related with distinct relationships

The UEML will be therefore the result of *reasonable choices* concerning relationships between languages. These choices may evolve, i.e. changing or increasing in the number, because it may appear interesting both new relationships between EMLs and new EMLs. As a consequence, any possible choice should be well identified and documented, together UEML itself. This is the reason for defining a *"Strategy for UEML"*[2], Section 4, which represents an attempt for providing a well structured context for making choices about the UEML. The basic principles underlying this "Strategy for UEML" are explained in Section 3.

3 Defining (a) UEML: key points

As any EML, UEML is equipped with a structure (abstract syntax) represented by the set of constructs and links between them, and some meaning associated to each construct (i.e. construct semantics) ([18]). When translating knowledge represented in one EML to another EML, it is important to define some commonality between constructs (referred in the remainder as the semantic correspondences between constructs) belonging to each language. In this sense, UEML should comprise, at least, constructs common to the most representative existing EMLs.

UEML handles the integration problems enumerated in Section 1, essentially within the language dimension, in the following ways:

- Differences in the (abstract) syntax of the languages: UEML is founded in the *agreement* concerning the structure (i.e. the Constructs) of the knowledge (i.e. the Concepts underlying the Constructs) being exchanged or translated. The way for expressing this (abstract) syntax is a *class model* and specifically (just for convenience) a *UML class model* ([8]) which is conveniently called *UEML meta-model* (because it models the syntax of a modelling language).
- Coverage and expressivity of the EMLs: the decision to be taken is related to what kind and to which extent of knowledge should be shared by using UEML; in this case, the following "equation" has been defined

  UEML[3] = Common Concepts + (some of) Non Common Concepts;

  This "equation" well represents the UEML as a federator component usually containing what it is shared among various components (see also Section 4, step f).

[2] The name "Strategy for UEML" is not the most appropriate. However, in this paper, we maintain the name originally given in the UEML project. Probably, in further developments, the name may be modified to "Methodology for UEML".

[3] We prefer to distinguish carefully between the construct and the concept underlying the construct which explicitly refers to "meaning" of the construct: in fact, for instance, while two constructs may be distinct, the underlying concepts can be the comparable (or equivalent).

– Differences in semantics of "similar" constructs: this is the most difficult type of problem and it is related to the definition of common concepts. A part of the complexity comes up because the meaning (i.e. underlying concepts) of most EMLs constructs is provided by text expressed in *natural language*: natural language is ambiguous, words are vague and interpretation depends on both the context and the person reading/writing the text. Then, using these texts for finding semantic relationships between constructs belonging to distinct languages is hardly difficult and quite subjective. To cope with similar problem, previous works concerning techniques for *database schema integration* ([15]) suggest to use examples (i.e. data or instances belonging to entities or relational table etc.); the purpose is to identify and verify clearly the "intentional relationships between concepts" by using the evidence of available examples of such concepts. The stated analogies [11][18] between languages and databases are now explicitly shown in Table 1 below (including some samples).

Table 1. Analogies between modeling languages and databases

Database Glossary	**Modelling Language Glossary**	**Samples of Model, Model artefacts and Meta-model**
Data (instances)	Model artifacts	"Mount", "Employee", "Robin"
Database (a set of related instances)	Model (a set of related model artefacts)	{"Mount", "Dismount", "Mechanical Part", "Employee", "Robin", "Edgar"}
Database Schema	Meta-model (representing a way for providing an abstract syntax of a language i.e. an abstract syntax for its constructs)	Activity, ObjectClass, Role, Object

Therefore, in the UEML project, we defined a complex *scenario* (comprising a set of models and related model artifacts) playing the role of databases and instances, by using three distinct modelling languages i.e. IEM, EEML and GRAI. Meta-models for each of these languages were also built. Afterwards, as it happens within the database world, we were interested in comparing these meta-models based on models and model artefacts,part of the scenario. Based on such comparisons, in the simpler case of two constructs with their underlying concepts C1 and C2 respectively in EML1 and EML2, we found the following types of *semantic correspondences* (or relationships)

1) $C1 = C2$
2) $C1 \supseteq C2$
3) $C1 \subseteq C2$

4) Otherwise[4], in the set of models $C1 \cap C2 \neq \varnothing$.

For instance, the first semantic correspondence (1) can be paraphrased as: by looking into the available models, represented in EML1 and EML2 respectively, the model artifacts represented through the concept C1 are also represented by C2 and vice-versa.

In the general case, a common concept C can be introduced if $C1 \cap C2 \neq \varnothing$ and, in this case, if possible, *mappings* might be used for better characterising their semantic correspondences through two predicates Φ_1 and Φ_2 such that

$$\Phi_1(C1)\ \rho\ \Phi_2(C2) \qquad (\mathbf{1})$$

where ρ is '=' or '$\supseteq$' or '$\subseteq$'.

The formula (1) is what can also be found in the *peer-to-peer data integration approach* [14].

Without loosing in generality, a common concept C can be defined by the following set of semantic correspondences, which characterises, in the data integration literature, the *GLAV approach* [13]:

$$C \supseteq \Phi_1(C1). \qquad (\mathbf{2})$$

$$C \supseteq \Phi_2(C2). \qquad (\mathbf{3})$$

These equations provide C as the best approximations of $\Phi_1(C1)$ and $\Phi_2(C2)$. This approximating concept C is closer to $\Phi_1(C1)$ and $\Phi_2(C2)$ as the inclusion '$\supseteq$' becomes the equivalence '=' i.e.:

$$C = \Phi_1(C1). \qquad (\mathbf{4})$$

$$C = \Phi_2(C2). \qquad (\mathbf{5})$$

Therefore, common concepts in UEML can be introduced by using the kind of equations defined above.. All these equations can also be used for realizing *standard mechanisms for translations* from an EML to UEML and vice-versa. However, this point requires some attention because these equations are verified on specific scenarios (i.e. specific models and model artifacts). Nevertheless, if these scenarios deal model-independent translations, this approach is able to find and to justify them.

The approach is also able to manage situations in which one concept in a language is related to more than one concept in another language. Indeed, more equations (1) involving a same concept are possible. The main problem is how to consider this kind of situation for making a UEML. The early work concerning this problem states that the key point is if the equations (1) whenever involving a same concept should be used simultaneously or singularly. This early work identifies three situations:

- scenario models are not correct (this situation may not happen if models are built by experts in the retained EMLs);

[4] In this case, language meta-models may be reworked, introducing *semantic enrichments* as for database schema integration ([15]).

- the concept involved in more equations (1) of a first language is overloaded (i.e. that concept can be used for modeling “real world things” which are represented through distinct constructs in another language i.e. those “real world things” are probably distinguishable);
- distinct constructs in the second language are “distinct views” over some “real world things” which have a unique representation in the first language.

4 Strategy for UEML

The UEML project was a pilot project to demonstrate the feasibility of a UEML according to the objectives described in Section 1. Therefore, the results of the UEML project were limited to the definition of a first version of the UEML (UEML 1.0) dealing with three EMLs: IEM ([17]), EEML ([2]) and GRAI/Actigrams ([5], [4]), each of them supported by EMTs, respectively MOOGO, METIS and eMAGIME. These tools have been supplied by three of the core UEML project members: Fraunhofer IPK Berlin, Germany, Computas, Norway, and Graisoft SA, France. The Strategy for UEML has been applied using these three EMLs. Besides, standard translation mechanisms have been used for implementing standard exchanges between the three tools.

The ”Strategy for UEML”, that is, the procedure followed to define UEML, is composed of the following steps (depicted in Figure 4):

a. It is defined a scenario (a specific description of a problem which plays the role of the set of models and related model artifacts).
b. The scenario is modelled by using IEM, GRAI and EEML This task is performed by experts on each of such three languages (however, modelling remains a subjective task).
c. Definition of the meta-models of the three languages using UML class diagrams: IEM, GRAI and EEML. These meta-models represent the constructs and the links among them. Each meta-model states what is possible to express with each language (i.e. the language abstract syntax).
d. The meta-models of IEM, EEML and GRAI have been verified by UML experts.
e. The models built up on step (b) are composed of a set of model artefacts. Following the principles described in Section 3, the semantic correspondences between constructs of each pair of the three modelling languages, have been stated.
f. Common and also (some) non-common concepts of the three EMLs have been identified (it should be noted criteria defined in Section 3 work well for identifying common concepts; the remaining concepts can be classified as non common but again in the context of the specific scenario; this means that two concepts can be classified as distinct until new scenarios contradict their “non commonality”; this also means that the result is always correct because we may include in the UEML meta-model more distinct constructs that some other scenarios will reveal their commonality).

g. The first version of UEML language has been defined by using the UML class diagram (i.e. UEML 1.0 meta-model). This version takes into account both the common and the non-common concepts,
h. The table of semantic correspondences between the UEML 1.0 meta-model and each of the three meta-model languages.
i. A final verification (using the scenario itself) of the UEML 1.0 meta-model and the found semantic correspondences with the three language meta-models has been performed.

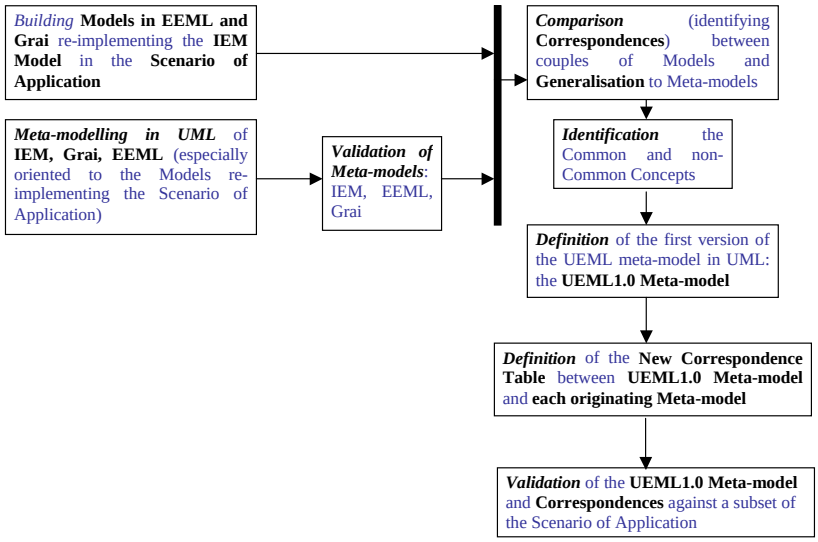

Figure 4. Strategy for UEML

The resulting UEML 1.0 meta-model (without attributes) is shown in Figure 5 below.

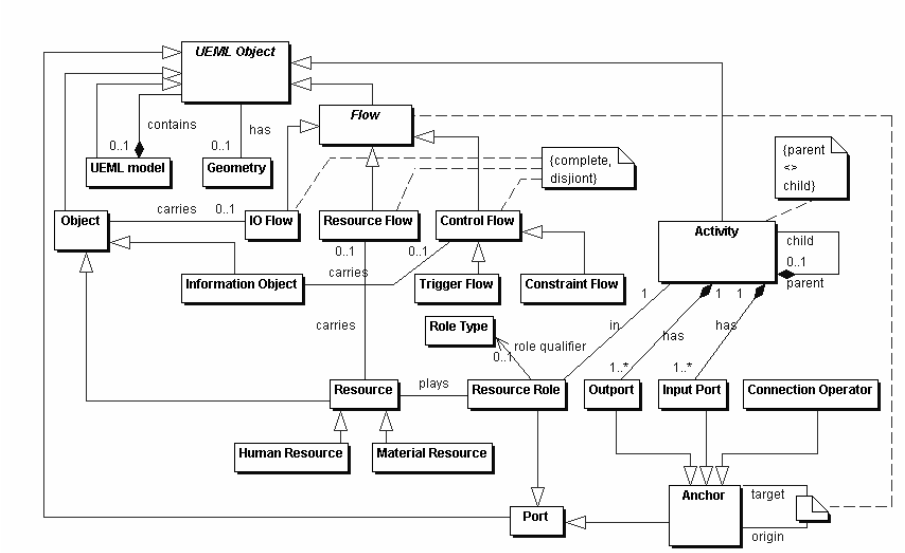

Figure 5. UEML 1.0 metamodel

The Table 2 (output of step h) shown below provides the semantic correspondences between the UEML 1.0 meta-model and the three language meta-models. Such a table is based on the principles advocated previously in Section 3. In fact, each row represents the equations characterising each common concept.

Table 2. Semantic correspondences

C	Φ1 (C1)	Φ2(C2)	Φ3 (C3)
COMMON CONCEPT	**GRAI**	**IEM**	**EEML**
ACTIVITY	Extended activity	Action state	Task
ROLE	Not explicit	IEM Object state	Role
RESOURCE	Resource	Resource class	Resource
INPUT/OUTPUT FLOW	Input / Flow Ouput / Flow	Successor / ProcessElement	Flow (with is control flow = false)
CONSTRAINT FLOW	Control / Flow (with trigger=false)	No direct	Flow (with is control flow = false and Role)
CONTROL FLOW	Control / Flow (with trigger=true)	ControlSuccesor / ProcessElement	Flow (with is control flow = true)
RESOURCE FLOW	Resource / Flow (with tri-gger =	ResourceSuccessor / Resource State	Role (with Task)

	false)		
CONNECTION OPERATOR	Logical operator	Connection Element State	DecisionPoint (not (Inport or Outport))
PORT	Connector	Port	Decisionpoint (Inport or Outport)

5 Conclusions

It has been shown the convenience of defining and using a UEML for supporting the EI. We have discussed and justified the basic principles and the adopted approach ("Strategy for UEML"). We have also presented the results of the undertaken approach, specifically the UEML 1.0 meta-model and semantic correspondences between the UEML 1.0 meta-model and the used EMLs i.e. IEM, EEML and GRAI.

The benefits of the "Strategy for UEML" are:

- to provide a well defined context for making "decisions" on how to make a UEML by reusing existing EMLs;
- to be based on existing theories and approached in database schema integration and data integration;
- to suggest mechanisms for realizing model-independent translations through a UEML between distinct EMLs.

The approach however needs to be refined because some points are not fully explored. Specifically, the definition of common and non common concepts, the use of the equations, the mechanisms for standardised model-independent translations, the relative complexity. Furthermore, alternative approaches to UEML should also be taken into account much more than in the UEML project. During the UEML project a number of needs where enterprise modelling plays a central role have been elicited (in targeted workshops) but poorly used for further improvements of the UEML 1.0 meta-model. Future UEML probably should also be driven by these needs even if they require a lot of work for their refinement.

References

1. AMICE: CIMOSA: Open System Architecture for CIM, Research Reports ESPRIT, 1, Project 688/5288 AMICE. Springer-Verlag, 2nd revised and extended edition, 1993.
2. EXTERNAL. Extended Enterprise Resources, Networks and Learning, EC project, IST-1999-10091, 2000.
3. Ortiz, A., Franco, R.D., Alba, M.: "V-CHAIN: Migrating from Extended to Virtual Enterprise within an Automotive Supply Chain". PROVE'03 Proceedings. Processes and Foundations for Virtual Organizations.
4. Doumeingts, G., Vallespir, B., Chen, D.: Decision modelling GRAI grid. In, P. Bernus, K. Mertins, G. Schmidt (Eds.) Handbook on architecture for Information Systems, Springer-Verlag, 1998.

5. Doumeingts, G., Vallespir, B. Zanettin, B., Chen, D.: GIM, GRAI Integrated Methodology - A methodology for designing CIM systems, Version 1.0, Unnumbered report, LAP/GRAI, University of Bordeaux 1, France, 1992.
6. GERAM. Generalized Enterprise Reference Architecture and Methodology Version 1.5, IFAC-IFIP Task Force on Enterprise Integration, 1997.
7. Kosanke, K., Vernadat, F.B., Zelm, M.: CIMOSA: CIM open systems architecture - evolution and application in enterprise engineering and integration. Computers in Industry, special issue, 40(2-3), 1999.
8. OMG: UML 1.5 specification, OMG, www.uml.org, 2002.
9. Vernadat, F.B.: Enterprise modelling and integration: principles and applications. Chapman & Hall, 1996.
10. Petit M., et al: D1.1: State of the Art in Enterprise Modelling, UEML-IST–2001-34229, www.ueml.org, 2002.
11. Petit, M.: Some methodological clues for defining a UEML. In Proc. of ICEIMT 2002 (Kosanke K., Jochem R., Nell J., Ortiz Bas A. (Eds.)), April 24-26 Polytechnic Univeristy of Valencia, Valencia, Spain, Kluwer. IFIP TC 5/WG5.12, 2002.
12. Williams, T.J. et al: The Purdue Enterprise Reference Architecture, Computers in Industry, 24(2-3), 1994.
13. Calvanese, D., De Giacomo, G., Lenzerini, M.: Description logics for information integration. In Computational Logic: From Logic Programming into the Future (In honour of Bob Kowalski), Lecture Notes in Computer Science. SpringerVerlag, 2002.
14. Calvanese, D., Damaggio, E., De Giacomo, G., Lenzerini, M., Rosati, R.: Semantic data integration in P2P systems. In Proc. of the Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing, 2003.
15. Parent, C., Spaccapietra, S.: Database integration: the key to data interoperability. In Papazouglu M.P. Spaccapietra S. Tari (Eds), Advances in Object-Oriented Data Modeling. MIT Press, 2000.
16. Klein, M.: Combining and relating ontologies: an analysis of problems and solutions. Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, USA, August 4-5, 2001.
17. Jochem, R., Mertins, K.: Quality-Oriented Design of Business Processes. Kluwer, Boston, 1999.
18. Berio, G., Petit, M.: Enterprise Modelling and the UML: (sometimes) a conflict without a case. In Proc. of Concurrent Engineering Conference 03, July 26-30, Madeira Island, Portugal, 2003.
19. Jochem, R.: Common Representation through UEML-Requirements and Approach. International Conference on Enterprise Integration and Modelling Technology. Enterprise Inter- and Intra-Organizational Integration. Kluwer, Valencia, Spain, 2003.
20. Ushlod, M.: Where are the semantics in the semantic web? AI Magazine 24(3), 2003.

Acknowledgment

The authors would like to thank all the UEML IST-2001-34229 core members for their scientific contribution to the work. This work is partially supported by the Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract N° 508011, <http://www.interop-noe.org>).