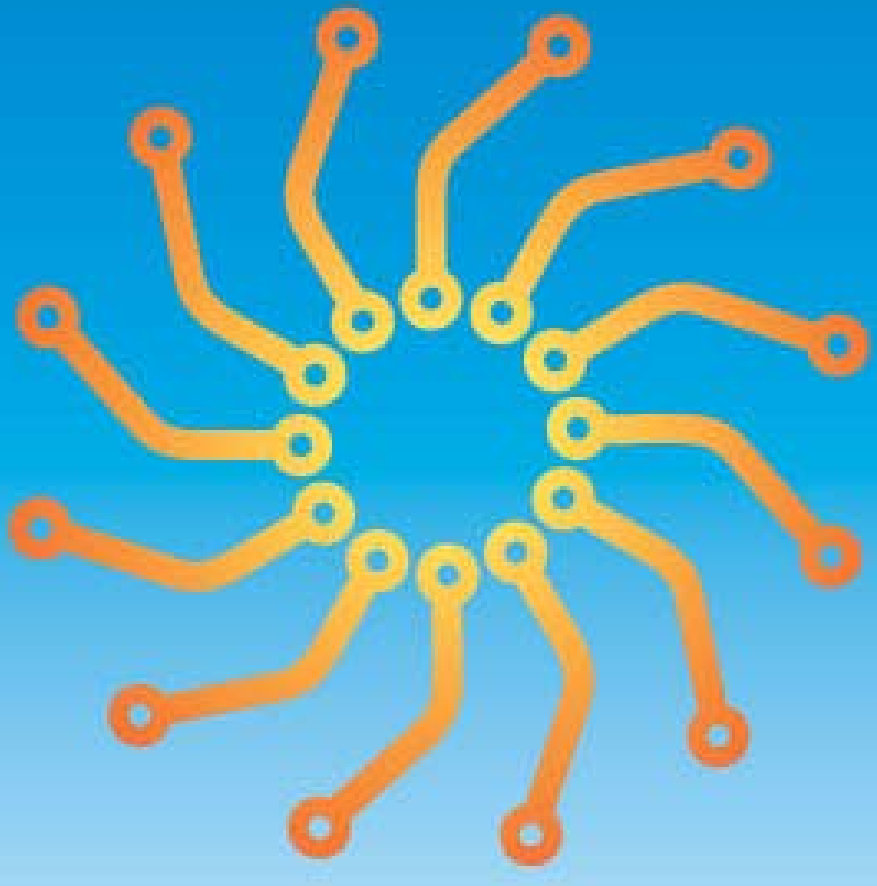
INCOM 2004



11th IFAC Symposium on Information Control Problems in Manufacturing

Salvador da Bahia – Brazil, April 5 – 7 2004

A UNIFIED ENTERPRISE MODELLING LANGUAGE FOR ENHANCED INTEROPERABILITY OF ENTERPRISE MODELS

Hervé Panetto[1], Giuseppe Berio[2], Khalid Benali[3], Nacer Boudjlida[3], Michaël Petit[4]

[1]*CRAN UMR 7039, University Henri Poincaré Nancy I, F-54506 Vandoeuvre-les-Nancy Cedex*
Herve.Panetto@cran.uhp-nancy.fr
[2]*Dipartimento di Informatica, Università di Torino, Torino, Italy*
berio@di.unito.it
[3]*LORIA UMR 7503, F-54506 Vandoeuvre-les-Nancy Cedex*
{Khalid.Benali, Nacer.Boudjlida}@loria.fr}
[4]*Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium*
mpe@info.fundp.ac.be

Abstract: There is a serious backwardness in awareness, acceptance and wide use of the Enterprise Modelling (EM) technology in industry because enterprises cannot capitalise from previous modelling efforts. This situation hinders true enterprise integration, interoperability, and enterprise knowledge sharing. A Unified Enterprise Modelling Language, based on meta-modelling of existing EM Languages, would serve as an Interlingua between EM tools providing the business community with a common visual, template based language to be used on top of most commercial enterprise modelling and workflow software tools.

Keywords: enterprise modelling, meta-modelling, interoperability, enterprise system

1. INTRODUCTION

Today's "business trends are clearly towards the need for managing organizational and operational changes within companies in order to face global competition and fluctuating market conditions" (Vernadat, 1996). This situation poses a number of integration (and interoperability) problems that enterprises have to tackle: integration of markets, integration between several development and manufacturing sites, integration between suppliers and manufacturers, integration of design and manufacturing, integration of multi-vendor hardware and software components. Enterprise Integration is therefore, not anymore only a problem of interconnecting physical and software applications but it also requires global business integration, aiming at the use of the existing or new enterprise resources in order to better achieve the overall business objectives. "Things to be integrated and coordinated need to be modelled. Thus, Enterprise Modelling (EM) is clearly a prerequisite for enterprise integration".

According to Vernadat (1996), *enterprise modelling* is the set of activities or processes used to develop the various parts of an enterprise model to address some desired modelling finality. It can also be defined as the art of "externalising" enterprise knowledge, i.e. representing the enterprise in terms of its organisation and operations (e.g. processes, behaviour, activities, information, object and material flows, resources and organisation units, and system infrastructure and architectures). The finality is to make explicit facts and knowledge that add value to the enterprises or can be shared by business applications and users. The prime goal of enterprise modelling is not only to be applied for better enterprise(s) integration but also to support analysis of an enterprise, and more specifically, to represent and understand how the enterprise works, to capitalize acquired knowledge and know-how for later reuse, to design (or redesign) a part of the enterprise, to analyse some aspects of the enterprise (by e.g. economic analysis, organization analysis, qualitative or quantitative analysis,...), to simulate

the behaviour of (some part of) the enterprise, to make better decisions about enterprise operations and organization, or to control, coordinate and monitor some parts of the enterprise.

Within the initiative on Computer Integrated Manufacturing (CIM), Enterprise Modelling was born in the United States at the beginning of the 80's and emerged through large CIM projects (e.g. ICAM or CAM-I). In the mid-80's, Europe launched several projects on Enterprise Modelling giving birth to several enterprise modelling languages (including notably GRAI (Doumeingts, *et al.*, 1998) and CIMOSA (AMICE, 1997; Kosanke, *et al.*, 1999)). As a result, in the 90's many commercial tools dealing with EM or business process modelling appeared on the marketplace, e.g. ARIS ToolSet, FirstSTEP, METIS, Enterprise Modeller, KBSI, CimTool, MO^{2}GO, e-MAGIM and many others.

Because most resources in modern enterprise were computerised systems, the problem of enterprise integration was also been approached with *workflow systems*, each one with its own modelling environment (Action Workflow, COSA, FlowMark, Lotus Notes, Teamware Flow, Ensemble, WorkParty, ...) and, in the late 90's, with EAI (Enterprise Applications Integration) tools. In computerised systems, the situation is currently worse than before. This is mainly due to the "Internet based systems" which, while offering very powerful, low cost and open infrastructures, fall short of integration. In fact, the many related standards (e.g., XML and its customisations) do not directly address the "problem of meaning".

This intensive production of tools has led to a *Tower of Babel situation* in which the many tools, while offering powerful and distinct functionalities, are unable **to interoperate** and can hardly or not at all communicate and exchange models. This is a serious drawback for awareness, acceptance and wide use of the EM technology since enterprises cannot capitalise from previous modelling efforts. This situation hinders true **enterprise integration, interoperability, and sharing enterprise knowledge**.

2. ENTERPRISE INTEGRATION AND ENTERPRISE MODELLING

Enterprise Modelling is an *engineering discipline* closely related to computerised systems. As such, it requires the combined use of Enterprise Modelling Software Environments (EMSE), Enterprise Modelling Languages (EML), and Enterprise Engineering Methodologies (EEM).
According to this point of view, there exists a lot of fragmented approaches to enterprise modelling (including Methodologies, Languages and Tools).
They cover different subsets of the different components of the enterprise engineering world. For instance, the ENV12204 (CEN, 1995) standard of CEN provides an Enterprise Modelling Language, but does not address any of the other components (no tool, methodology or meta-modelling capacity)); GERAM (GERAM, 1997) and ISO/IEC 15288 define methodological guidelines for Enterprise Engineering but without any modelling language; the Workflow Management Coalition (WfMC) provides a modelling language (WPDL) but without methodological support for using this language for modelling processes; the same happens with other EAI tools (such as the so called Integration Brokers).

They also cover different parts of the *enterprise life-cycle*. For instance, ebXML and WfMC focus on software design while approaches like CIMOSA mainly concentrate on enterprise requirements and design. Additionally, some approaches link enterprise models to Enterprise Operation Tools (EOT). These links allow the produced models to be used, for instance, for process *enactment and control* (e.g. in the WfMC approach, WPDL models are used by a workflow engine to control the execution of ongoing work). Models may be also linked to enterprise software applications like ERP (Enterprise Resource Planning) systems (e.g. ARIS and mySAP.com). Some other approaches only aim at modelling for understanding and analysing and they do not provide explicit links to operational systems or to other models in the life-cycle (e.g. ENV12204).

Enterprise modelling approaches may also have very different objectives and needs. As a meaningful example, we may compare the aims of IEM and GRAI. IEM allows representing business processes and it provides specific concepts adapted for assessing quality management procedures, but it cannot directly be employed for an operational implementation of the business processes. In fact, for quality management, it is not necessary to fully define an implemented process. The description has only to be sufficient to enable determining whether the process steps conform to defined quality procedures. This later objective requires, for example, representing outcomes of each process step and their usage as inputs for other process steps. On the other hand, the main objective of GRAI modelling is to define the control system of an enterprise. This requires a very good understanding of the relationships between the business processes at the various control levels (operational, tactical, and strategic). Quality procedures do not guarantee that an enterprise has good performance but only that its products conform to some quality criteria, whereas enterprise control tries guarantee that an enterprise takes into account market data and reflects them in the enterprise internal behaviours.

In enterprise modelling, there seems to be a tendency for approaches combining, in a more or less integrated way, *several sub-languages* or views (see e.g. CIMOSA, GRAI, and ARIS). A

combining approach allows taking advantage of the strengths of each of the sub-languages and offers the advantage that the resulting combined method or methodology offers the modeller the capacity to meet more modelling objectives. Models built with the distinct languages have to be related in some way and the *languages have to be integrated*. This need for integrating distinct modelling languages and relating models has also been recognised in domains like *software formal methods* for achieving effective and "practical" solutions to complex problems.

However, the integration of several sub-languages (often called *views*) is currently always performed within a single tool (i.e. in a single approach which creates many overlapping with other existing approaches). No tool (at least in the enterprise modelling domain) currently supports the combination of its own models with models created with a language supported by another tool.

A *completely integrated language* allowing the creation of models combining all needed aspects of the reality is probably unachievable and the supporting tools for that language would be too complex to build. Therefore, the only reasonable approach seems to be to create a modelling environment from "legacy" enterprise modelling tools (and languages) allowing to reuse existing models and to leverage these existing models and tools into an *integrated environment*. This integrated environment should also be complemented with a process for extending, in a controlled way, a set of limited constructs belonging to a *core language*. As we will see in the remainder (section 3.3), a sample of this process has been defined during the UEML project.

3. THE UEML

The UEML project[1] was set up in an attempt to contribute to the solving of the problems of multiple EMLs. The long term objective of UEML is the definition of a core language called Unified Enterprise Modelling Language, which would serve as an *Interlingua* between EM tools. This language will:

- Provide the business community with a common visual, template based language to be used on top of most commercial enterprise modelling and workflow software tools;
- Provide standardised mechanisms for sharing and exchanging enterprise models among projects, overcoming tool dependencies;
- Support the implementation of open and evolutionary enterprise model repositories to leverage enterprise knowledge engineering services and capabilities.

[1] UEML IST-TN 2001 34229, www.ueml.org

In order to prepare this long term objective, the UEML project was initiated with the objective to create and manage a working group aiming to:

- Create a European Consensus on a core set of modelling constructs and facilitating interoperability in the frame of on-going standardisation efforts in this domain.
- Build a demonstrator portal with services and contents to support and promote, testing, industrial validation, and to collect comments.

The first objective of the project was to analyse the market potential of a UEML, to accurately define the specifications of an embryo of such a language and to demonstrate and disseminate the concepts.

3.1 The need for UEML

Two main efforts related to the definition of a common core language for enterprise modelling are PSL (PSL, 2002) and ENV12204[2] that however do not currently provide a satisfactory answer to critical and also practical problems. PSL is a *logic-based approach* that does not clearly address the problem of the basic *mapping* between a generic EML and PSL itself. This mapping should e.g. state, for instance, that the concept of *Activity* belonging to an EML corresponds to the concept of *Activity* in PSL. Being a *declarative language*, PSL allows discovering *inconsistencies* between distinct models provided in distinct EMLs. However, it neither prevents nor *solves* these inconsistencies. ENV12204 is merely a set of useful concepts for understanding the *domain of enterprise modelling* (or even the set of things that need to be represented by any enterprise modelling language). However, its *syntax* is not well defined and therefore it cannot be used as an *exchange format* between distinct tools. It also does not define mappings between existing EMLs and itself.

The usefulness of a UEML would therefore reside in the availability of a *well-defined syntax* and *well-defined mappings* (possibly *standardised*) between various EMLs and UEML. We believe that the definition of mappings between languages and UEML is important but quite *independent* from the UEML definition itself. Thought, they should be precisely defined and shared (through, for instance, standardisation), they should base on *reasonable hypotheses* and will never be fully (and formally) provable.

Other approaches which attempt to solve the *problems of exchange and interoperability* between computerised systems do not deal clearly with the enterprise modelling area. They can be classified as ways for enabling *business level communication* between distinct *computer-based systems* and therefore as bottom-up approaches. For instance, ebXML, WPDL, EAI techniques are useful for defining communication at *business level* among

[2] A new version of ENV 12204, EN ISO 19440 is expected in early 2004.

enterprise software. These approaches are really useful for *programming software layers* such as *middleware* (e.g. CORBA) and *customising software architectures*. This description of purely software aspects can be considered as a kind of *high level programming* that anyway requires *enterprise modelling as a prerequisite*.

However, another prerequisite for the exchange of models (or to make models interoperable) through a common language to be meaningful is to clearly understand the *semantic links* existing between the models themselves. This understanding is fundamental because without it, an exchanged model could be understood in a totally different way by the receiving tool, and thus misinterpreted.

3.2 The need for meta-modelling

In order to understand the links between distinct languages, *meta-modelling* is an important issue. Meta-modelling allows defining the *syntax of a language*[3]. The product of meta-modelling is usually called a *meta-model*.

Meta-models need to be described by using *meta-modelling techniques* (i.e. languages for making meta-models). Approaches like XML (DTDs and Schemas), MOF, Telos, can be used. These techniques are *content-independent* (applicable for the definition of any language). Other meta-modelling techniques are *content-dependent (*and sometimes *domain-specific)*: for instance, XMI is an exchange format based on the meta-model of UML (UML, 2003) in XML designed for enabling exchange of UML models.

Accordingly, a *UEML could be defined as a content-dependent domain-specific meta-model* through a content-independent meta-model. The UEML might just use content-independent meta-modelling techniques[4] as a *way for its definition*.

Currently, several Meta-Modelling Languages (and also tools) exist but none of them are specifically targeted for the definition of EMLs. The reason is that these Meta-Modelling Languages were often developed to design and implement *Information Systems, Knowledge Base Systems and computer-based infrastructures* (environments) allowing to program meta-models.

3.3 The UEML approach

In the UEML project, a UEML meta-model was built on the basis of three existing languages (namely IEM, EEML and GRAI). An illustrative scenario was defined in which models of a common situation are stored in distinct software tools and exchanged. First, models of this scenario where elaborated in the three distinct languages and. the exchange was performed manually by specialists of these languages. More precisely, given a first model in IEM, specialists of GRAI and EEML provided the "semantically equivalent" model in their own languages. Afterward, IEM, EEML and GRAI constructs have been meta-modelled using UML class diagram. This resulted in three so-called "original meta-models". At the same time, the links between the concepts of every original meta-model and their use in the models of the Scenario were defined to illustrate how a unique real-world phenomenon is modelled with the three original meta-models.

With the aim to define a common meta-model for core constructs, we compared and "unified" the three meta-models through an incremental approach. We compared the three meta-models peer-to-peer to find any correspondence between a concept in one meta-model and a concept in another one.

Once the peer-to-peer correspondences (and absence of correspondences) had been defined, a set of common concepts were identified (Table 1) and further elaborated into the first version of the UEML meta-model 1.0 (Fig. 1). This meta-model represents the common concepts underlying the three original EMLs. This meta-model being remarkably different from the three meta-models by the use of an appropriate higher level of abstraction and considering some discrepancies among the three original meta-models, we informally re-defined new correspondences between the UEML meta-model and each original meta-model. Finally, the UEML meta-model and the new correspondences were validated against a subset of the Scenario.

3.4 Defining mappings among EMLs

The clear definition of the meta-models of existing EMLs and of UEML with meta-modelling techniques is necessary but not sufficient to achieve a meaningful exchange of models. The correspondences among constructs between two distinct languages have to be precisely defined by comparing semantics of these constructs. However, this is a difficult task because:

- EMLs are often based on informal semantics, i.e. some natural descriptions of constructs meaning.
- The way in which EMLs are used in specific context and situations may change.

Therefore, as suggested in Sect. 3.1, mappings between languages should rely on reasonable hypotheses should be clearly stated and become the base for building the language, and possibly be standardised further.

---

[3] A meta-model may also be used to define part or all of the semantics of the language. But this is often not recommended.

[4] It should be noted that the notion of meta-modelling technique is *relative*. In fact, it is often true that a language can be used as a meta-modelling technique for another (sometimes, the same) language.

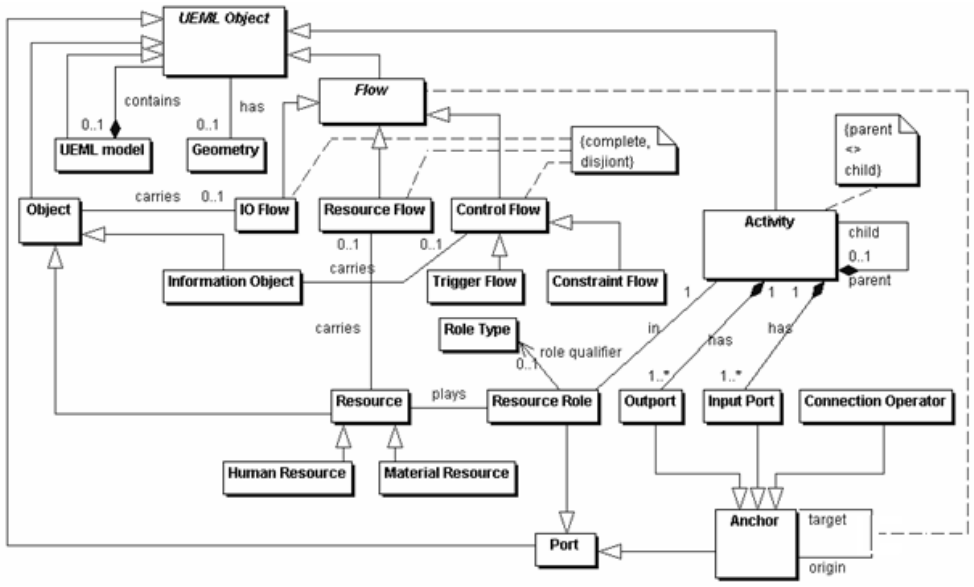


Fig. 1: The UEML 1.0 meta-model

Table 1: Extract of the common concepts of UEML

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 (control flow= false)
CONSTRAINT FLOW	Control/Flow (trigger=false)	No direct	Flow (control flow= false and linked to Role)
CONTROL FLOW	Control/Flow (trigger=true)	ControlSuccessor/ProcessElement	Flow (control flow= true)
RESOURCE FLOW	Resource/Flow (trigger=false)	ResourceSuccessor/Resource State	Role (linked to Task)
CONNECTION OPERATOR	Logical operator	Connection Element State	DecisionPoint (not (Inport or Outport))
PORT	Connector	Port	Decisionpoint (Inport or Outport)

Mappings can be defined, more or less precisely, in various languages. For example, they can be expressed informally in natural language or through the use of a meta-modelling language. From a technical viewpoint, XSLT is a proposal to define transformation of XML documents based on a set of transformation rules expressed on the basis of XML schemas. The advantage of this approach is that software tools are already operational to interpret these mappings and apply them on XML documents. This approach could be considered as a means of implementing correspondences among EMLs, provided that these have XML syntax. Defining relationships at the language level can also be done in an "a priori" manner when new methodologies and methods are under definition. Therefore, a UEML can be a good starting base for placing under control the process of defining new methods and methodologies as well as the rules applied in a specific methodology.

4. CONCLUSION AND OUTLOOK

4.1 Future perspective for enterprise modelling

This analysis of the state of the art in enterprise modelling (Petit, *and al.,* 2002) demonstrates the need to define and develop a UEML approach to solve the current problems faced by workers in the enterprise modelling domain.

But such UEML approach can only be successful and effective under two conditions:

- Providing a global approach of interoperability among enterprise modelling software going further than only providing a common format of exchange;
- Making clear and effective the link between enterprise modelling and Enterprise Applications and Software.

4.2 UEML as a global approach to enterprise modelling

As stated earlier, a common exchange format, if deemed successful, cannot be described independently of mappings to and from existing EMLs. Furthermore, this requires the explicit definition of meta-models of the involved languages and of the mappings among their concepts. However, in order to avoid that UEML, as a common format, becomes yet another language among the large set of existing ones, it requires a larger view of interoperability among EM tools. The UEML language and approach must be flexible to be able to cope with future proprietary emerging languages and with the evolution of existing EMLs. The long term objectives of a

UEML approach would then be to provide the necessary concepts and tools to achieve the following:

- *Interoperability* between already existing supporting *tools as well as newly developed tools*,
- Well-founded integration basis between distinct *enterprise modelling languages*,
- Consistent global models on which also distinct *methodologies* can be integrated,
- Improvement of *existing methodologies* and definition of *new methodologies*.

These objectives pose a number of requirements on the UEML approach:

- The availability of meta-modelling concepts, methods and tools to properly define EMLs (existing ones, new emerging ones, UEML, their extensions and particularisations for specific purposes or applications);
- The availability of concepts, methods and tools to properly define relationships among distinct EMLs and a UEML together with relationships between models created with different EMLs and UEML;
- The concepts and tools to properly define methodologies associated with EMLs, including UEML;
- The specification of an *open architecture* in which all these elements can be implemented to provide an evolutionary multi-language platform for enterprise modelling centred on UEML.

This platform would allow creating *coherent, global and logically centralised* (integrated) models of the enterprise but which may be distributed within different enterprise modelling applications at a *physical level*.

4.3 From enterprise modelling to enterprise systems

We consider that Enterprise Engineering or Enterprise Modelling make sense provided that we are able to link the tools developed in this field with the Enterprise Applications and Software. Enterprise Modelling must be considered as the way to design the *architecture* and the *model* of the enterprise independently from existing or future Enterprise Applications and Software. We see the various levels as shown in the Fig. 2.

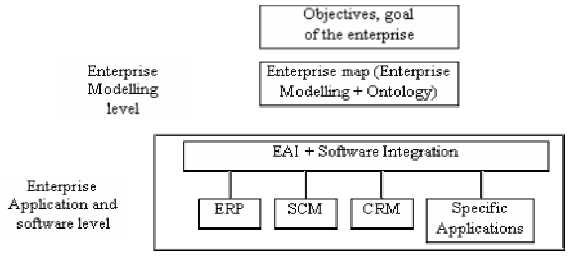


Fig. 2: Enterprise Modelling and Enterprise Application Levels

To our opinion, the future within ten years will be capabilities to generate, from the enterprise modelling level, the specifications allowing to choose or customize Enterprise Applications and Software (EAS) or to derive specifications of software applications for the enterprise models. Moreover, we believe that the EAS such as ERP, SCM or CRM will no longer have a centralised structure. Rather, they will have a modular and distributed structure. Each module will be chosen and customized according to specifications generated from the enterprise models. On the one hand, such a structure might *decrease* the time and cost of EAS development, and *increase* the adequacy of EAS to the needs of the enterprise. This should result in an increased acceptance of EAS by end-users, higher *adaptability* of EAS to the enterprise structure and improved *Return on Investment* of EAS development.

REFERENCES

AMICE. (1993). CIMOSA: Open System Architecture for CIM, *Research Reports ESPRIT*, **1**, Project 688/5288. Springer-Verlag.

Bernus P., Mertins K., and Schmidt G. (Eds) (1998). *Handbook on Architectures of Information Systems*. Springer Verlag, ISBN 3-540 64 453 9.

CEN (1995), CEN/CENELEC ENV 12204. *Advanced Manufacturing Technology, Systems Architecture, Constructs for Enterprise Modelling*, CEN, Brussels.

Doumeingts G., Vallespir B. and Chen D. (1998). Decision modelling GRAI grid, Chapter in: P. Bernus, K. Mertins, G. Schmidt (Eds.) *Handbook on architecture for Information Systems*, Springer-Verlag.

GERAM (1997). Generalized Enterprise Reference Architecture and Methodology Version 1.5, *IFAC-IFIP Task Force on Enterprise Integration*.

Kosanke K., Vernadat F.B., and Zelm M. (Eds.) (1999). CIMOSA: CIM open systems architecture - evolution and application in enterprise engineering and integration. *Computers in Industry, special issue*, **40(2-3)**.

Petit M, *and al.* (2002). *D1.1: State of the Art in Enterprise Modelling*, UEML TN IST 2001 34229, www.ueml.org

PSL (2002). Industrial automation system and integration - *Process specification language: Part 11: PSL-Core*. ISO/CD 18629-11.

UML (2002). *UML 1.5 specification*, OMG.

Vernadat F.B. (1996). *Enterprise modelling and integration: principles and applications*. Chapman & Hall.

ACKNOWLEDGEMENT

The authors would like to thank all the UEML core members for their scientific contribution to this work. This work was funded by the European Commission IST 5th framework programme.