Interoperable Language and Model Management Using the UEML Approach

Andreas L. Opdahl
University of Bergen
P.O.Box 7800, N-5020 Bergen, Norway
+47 – 5558 4140
Andreas.Opdahl@uib.no

Giuseppe Berio
University of Torino
Corso Svizzera 185, 10149 Torino, Italy
+39 – 011 6706810
Giuseppe.Berio@di.unito.it

ABSTRACT
The paper presents the approach currently used to define a new version of the Unified Enterprise Modelling Language (UEML). UEML is intended as an intermediate language that is able to incorporate and support integrated use of a wide variety of existing modelling languages. A central idea behind the UEML approach is separation of reference, a technique that supports language and model interoperability by (1) breaking modelling constructs and model elements down into ontologically primitive concepts and (2) mapping each concept onto a common ontology. The paper also discusses the UEML approach in relation to long-term, global, integrated model management. The aim is to direct future development of the UEML approach and of UEML, including its tools and organisation.

Categories and Subject Descriptors
H.1.0 MODELS AND PRINCIPLES – General, I.6.4 SIMULATION AND MODELING - Model Validation and Analysis, I.6.5 SIMULATION AND MODELING - Model Development (Modeling methodologies).

General Terms
Management, Documentation, Design, Standardization, Languages, Theory.

Keywords
Modelling language management, model management, interoperability, separation of reference, UEML, Bunge’s ontology, BWW model.

1. INTRODUCTION
Ongoing work within the Interop Network of Excellence (www.interop-noe.org) is developing version 2 of a Unified Enterprise Modelling Language (UEML2). The aim of UEML is to act as an intermediate language between existing enterprise modelling languages and, thereby, to facilitate interoperability between a wide variety of enterprise modelling languages and models. Central in the development of UEML2 is a language-integration approach that facilitates maximally tight integration of enterprise modelling languages and enterprise models.

This paper presents and discusses the UEML approach along with one of its central ideas – that of describing modelling constructs by separation of reference – with particular emphasis on long-term, global, integrated model management. The aim is to direct future development of the UEML approach and of UEML. An additional aim is to draw critique of the UEML work from a research community that focusses specifically on long-term, global, integrated model management.

The following Section 2 presents the background of UEML, before Section 3 explains the UEML approach and separation of reference. Section 4 then discusses the UEML approach in relation to long-term, global, integrated model management, considering physical, empirical, syntactical, semantic, pragmatic and social issues. Finally, Section 5 concludes the paper.

2. BACKGROUND
Model-driven technologies have recently come to the fore through initiatives such as the OMG’s MDA (www.omg.org/mda) and ModelWare (www.modelware-ist.org) and through developments towards model-driven enterprise systems, model-supported enterprise application integration, ontology-driven agents on the semantic web etc. These model-driven technologies use models of enterprises, their planned information systems and their problem domains to generate, tailor, adapt and otherwise deploy new ICT solutions. Advantages of model-driven technologies include more agile information systems, end-user tailorability and shorter development times. Unfortunately, the different technologies are driven by a wide variety of modelling languages. Continuously, new languages are being proposed that offer improved expressiveness or functionality. Standardisation efforts such as those of the OMG tend to focus on syntactical rather than semantic issues. An intermediate language is therefore needed to support integrated use of enterprise, IS and problem-domain models expressed in different languages while retaining their semantics. The Unified Enterprise Modelling Language (UEML) is an ongoing effort to develop such an intermediate language.

The idea of a Unified Enterprise Modelling Language emerged during the ICEIMT’97 conference (Goosenaerts, Gruninger, Nell, Petit & Vernadat 1997), with the aim of providing an underlying formal theory for enterprise modelling languages. A major motivation was enterprise integration in the face of a wide variety of existing languages (Vernadat 2002). The UEML Thematic Network (UEML TN) (2002-2003) was funded by the EU’s FP5. It had three main activities (Jochem 2002, Panetto, Berio, Benali, Boudjlida & Petit 2004, Mertins, Knothe & Zelm 2004, Berio, Anaya & Ortiz 2004): requirements collection and analysis, feasibility study and demonstrator...
development. It developed and demonstrated a common abstract syntax and exchange format, which incorporated three important industrial European enterprise modelling languages: EEML (EXTERNAL 2000), GRAL (Doumeingts, Vallespir, Zanettini & Chen 1992) and IEM (Jochem & Mertins 1999).

UEML development has since continued within the Interop Network of Excellence (www.interop-noe.org, 2003-2006), funded by the EU’s FP6. It also has 3 main activities: requirements, language selection and definition approach (Berio, Opdahl, Anaya & Dassisti 2005). Starting from high-level requirements established in the UEML TN, more focused requirements were collected using a novel requirements elicitation template. Next, languages to incorporate into UEML2 were selected using a set of quality criteria, which were linked to the collected requirements and evaluated using information collected from Interop partners using a language template. Currently, the selected languages are being incorporated construct-by-construct into UEML2 based on information collected from Interop partners using a construct template.

3. THE UEML APPROACH

Collectively, we will refer to the requirements, the language template and the construct template as the UEML approach in this paper, although focus is on describing modelling-language constructs using the construct template.

3.1 The construct template

The UEML construct template provides a standard, format for describing modelling constructs (e.g., Opdahl & Henderson-Sellers 2004, 2005, Dallons, Heymans & Pollet 2005, Heymans, Saval, Dallons & Pollet 2005). The template was used for compiling the initial version of UEML2. It provides a common structured format for describing modelling constructs, guiding information collection, supporting distributed work and ensuring that the resulting descriptions are consistent. It clearly and systematically distinguishes how a construct is presented, e.g., lexical and syntactical issues, from what a construct represents, i.e., reference, an important aspect of semantics, and from how a construct is used, e.g., pragmatics, although pragmatics has been less investigated in Interop so far.

The first paper-based version of the construct template was distributed to selected INTEROP partners in November 2004. Initial attempts were made in the first half of 2005 to describe BPMN, colored Petri nets, GRL, ISO/DIS 19440, UEML 1.0, selected diagram types from UML 2.0, XPDL and YAWL (Berio, Opdahl, Anaya & Dassisti 2005). The template was revised twice during this process. In the autumn of 2005, colored Petri nets, GRL and UML 2.0 class and activity diagrams were chosen for inclusion into the first demonstration version of UEML2, which is currently being prepared.

3.2 Separation of reference

A central idea behind the construct template and the UEML approach is separation of reference, which is used to describe at a detailed level what a modelling construct represents (Opdahl & Henderson-Sellers 2004, 2005). Through referential separation, modelling constructs can be systematically described in a way that prepares them for semantic integration, as elaborated in (Opdahl 2006) and outlined below.

The UEML approach and the template uses separation of reference relative to Bunge’s ontological model (Bunge 1977, 1979) and to the Bunge-Wand-Weber representation model of information systems (the BWW model, Wand & Weber 1988, 1993, 1995). Each modelling construct to be described is broken into ontologically primitive parts, i.e., into parts that map one-to-one with an ontology concept that is primitive relative to Bunge’s ontology and the BWW model. Examples of primitive ontology concepts are a particular class of things in the enterprise (e.g., ProductionEquipment), a particular type of property or relationship (e.g., having a responsibility), a particular state (e.g., being idle) or a particular event (e.g., being assigned a new responsibility).

In other words, using separation of reference, each modelling construct is described in terms of: (1) Which classes of things the construct is intended to represent. (2) Which properties of those things, if any, the construct is intended to represent. (3) Which states of those things, if any, the construct is intended to represent. (4) Which events between those states, if any, the construct is intended to represent. (5) Whether the construct is intended to represent the class or instance level or either. (6) Whether the construct is intended to assert facts or to express goals, beliefs, knowledge etc. (i.e., construct modality).

3.3 The common ontology

The classes, properties, states and events used to describe modelling constructs are maintained in a common ontology. Initially, this ontology was populated with classes, properties, states and events drawn from Bunge’s ontology and the BWW-model. As more modelling languages and constructs are incorporated into UEML, the ontology grows dynamically as more classes, properties, states and events are added.

The common ontology is organised hierarchically. Ontology classes are organised using class-subclass (ProductionEquipment is-subclass-of Equipment) and part-whole (Equipment is-composed-of EquipmentParts) relationships. Properties are organised using property precedence (being human precedes having a responsibility). States are similarly organised using super-/substate relationships and events using super-/subevent relationships.

All the modelling constructs are thereby interrelated at the most detailed level possible via the common ontology: If two modelling constructs are identical, they will map onto the exact same ontology concepts. If two modelling constructs do not overlap at all, they will map onto completely distinct ontology concepts, i.e., onto concepts that are not even hierarchically related. In the third and most common case, two modelling constructs may map onto some identical ontology concepts, some ontology concepts that are hierarchically related and some ontology concepts that are distinct. But in all cases, the hierarchically organised common ontology makes it possible to determine the exact representational (or referential) relationship between any pair or group of modelling constructs.

In this way, existing languages become integrated when their modelling constructs are mapped onto the common ontology. This integration is maximally tight in an ontological sense, because the modelling constructs are mapped onto primitive ontology concepts. This paves the way for comparison, consistency checking, update reflection, view synchronisation and, eventually, model-to-model translation across modelling language.
3.4 Meta-meta model and prototype tool

The meta-meta model in Figure 1 is based on earlier work by Opdahl & Henderson-Sellers (2004, 2005). The top part of the meta-meta model is for managing the relationships between languages, their diagram types and their modelling constructs. The bottom part shows the structure of the common ontology. The middle part is for breaking down modelling constructs and mapping them onto the common ontology.

Based on the meta-meta model, a prototype tool for managing construct descriptions, UEMLBase, has been developed using the OWL plug-in for the Protégé tool. The intention is that the tool should replace the paper-based construct template used so far. Construct descriptions created using the UEML template are currently being negotiated between the partners and entered into UEMLBase, revealing a first version of the common ontology.

4. DISCUSSION

This section will discuss the UEML approach in context of long-term, global, integrated model management, inspired by the modelling quality framework introduced by Lindland, Sindre & Solvberg (1994) and developed further by Krogstie (e.g., 2001). Hence, we will discuss physical, empirical, syntactic, semantic, pragmatic and social issues separately. The aim is to direct future development of the UEML approach and of UEML.

4.1 Physical quality

“The basic quality goals on the physical level is externalisation, that the knowledge \( K \) of the domain \( D \) of some social actor has been externalised by the use of a modelling language, and internaliseability, that the externalised model \( M \) is persistent and available enabling the audience to make sense of it” (Krogstie 2001).

As for externalisation, an explicit aim of the UEML approach is to support incorporation of a wide range of modelling languages intended to represent widely different types of phenomena in and aspects of enterprises. In this sense, the approach is able to express a wide range of knowledge, i.e., it is highly externalisable. On the other hand, the UEML work focusses on enterprises and closely related domains, such as information systems and their problem domains. Physical externalisation raises questions about whether and how the UEML approach can be used for other types of domains, such as software design, which are equally important targets of long-term, global, integrated model management, a possibility that should be investigated further.

As for internaliseability, physical persistence immediately raises the issue of repository-based versus transformation-based model management. Whereas it is clear that UEML – as any integrated or intermediate modelling language – must be properly managed as a persistent artefact, it is not equally clear whether UEML models should be maintained centrally in large integrated repositories or dispersed as partial models that are translated into other partial models when needed. An obvious advantage of partial models is that they offer a direct bridge to other families of models and languages, described and managed using other approaches than the UEML's. An equally obvious advantage of repositories is that they offer more elaborate services, e.g., for consistency checking. Translation approaches also lead to accumulative information loss when partial models are translated across languages in multiple imperfect stages. Perhaps the UEML approach should evolve to support both styles of model management. Partial models can be checked out of and into the central repository, but partial models can also be translated into other partial models. An intermediate decentralised solution is to have multiple repositories, connected by exchanging partial models. Decentralised repositories may even overlap to some extent.

Physical availability adds another dimension to the model-management problem, where model availability can be defined as the proportion of time a model is in a functioning condition. Availability is a central parameter when selecting between a centralised, decentralised, distributed or somehow combined architecture for UEML. It also raises the issue of redundant or multiple overlapping repositories. The availability aspect of physical internaliseability can be broadened to also account for related qualities that affect model use, such as reliability and performance. The choice between central, decentralised, distributed and combined UEML model management resembles the choice of information systems architecture in general and should be informed by existing theory.

4.2 Empirical quality

“Empirical quality: deals with predictable error frequencies when a model is read or written by different users, coding (e.g. shapes of boxes) and ergonomics of computer-human interaction for documentation and modelling-tools” (Krogstie 2001). Empirical quality is a central concern of the presentation part of the UEML approach, which includes lexical and pragmatic issues. But these issues so far remain less developed, and the pragmatics issues addressed are only simpler ones, such as social conventions for diagram layout. Instead, many usability issues will be relegated to a variety of types of model-based or model-driven front-end tools that are interfaced with UEML tools. The importance of empirical quality means that the UEML approach and tools should support simple development and management of domain specific languages (DSPs) and coordinated management of language dialects. Extending the UEML meta-meta model and tool to account for presentation issues is therefore necessary.

4.3 Syntactic quality

“Syntactic quality is the correspondence between the model \( M \) and the language extension \( L \) of the language in which the model is written” (Krogstie 2001). Again, syntax is a central concern of the presentation part of the UEML approach, which aims to support integrated management of multiple modelling grammars (aka presentations).

Syntactic quality in a UEML context does not mean finding a single “good” or “best” syntax (that might be the aim of a UEML core language), but incorporating a wide variety of syntactical presentations of the classes, properties, states and events in the common ontology. This means that a UEML repository must support checking in and checking out of partial models expressed using a wide variety of presentations, and it must support managing these presentations. In consequence, the UEML approach should support consistency checking between the presentation and representation parts of the meta-meta model (as soon as it has been extended to account for presentation).
Figure 1: The UEML meta-meta model.
If a non-centralised UEML architecture is chosen, a UEML repository should provide cross-language model-to-model translators for any pair of (overlapping) languages that have been incorporated into UEML. For our purposes, we can distinguish between two levels of translation: Syntax-level translation converts between models represented in standard file formats such as XMI (www.omg.org/xmi) or GXL (www.gupro.de/gxl). Structure-level translation converts between abstract structures, typically graphs. Whereas the UEML approach is already well equipped to support structure-level translation, it is not yet equipped to support translations between file formats, because the presentation part of the template does not account for file formats or mappings between them. One natural way to extend the UEML approach in this direction is to introduce sequential presentations in addition to diagrammatic ones. As it should be possible to store the same language in a variety of file formats, the same language could have several alternative sequential presentations. Indeed, it might even be advantageous to let the same language have alternative diagrammatic presentations, to account for minor syntactic dialects when there is no representational difference. The idea of having both sequential and diagrammatic presentations can be extended further into including even 3-dimensional and dynamic presentations, as envisaged in (Opdahl 2003).

4.4 Semantic quality

"Semantic quality is the correspondence between the model $M$ and the domain $D$. The framework contains two semantic goals: Validity which means that all statements made in the model are correct relative to the domain and completeness which means that the model contains all the statements which is found in the domain" (Krogstie 2001). Semantic quality is the central aim of the UEML approach, which has been developed specifically to maintain and integrate languages and models referentially – a central aspect of semantics – in situations where many different grammars are used to represent the same classes, properties, states and events.

A central semantic quality issue is to ensure that the common ontology is understood in a sufficiently similar way across spatially, temporally and culturally distributed UEML communities. Should concepts in the common ontology come to be understood differently by different stakeholders, UEML might quickly unravel. This means that both the common ontology and the mappings of modelling constructs onto the ontology must be well documented. It also means that UEML and its common ontology should be deliberately anchored in a core of well-defined, widely used and well-understood modelling languages that can be expected to evolve slowly and, thereby, stabilise the corresponding concepts in the common ontology. In addition, the language level – which is most dependent on the common ontology – is likely to evolve more slowly than the model level.

Nevertheless, the UEML approach must develop mechanisms for identifying and resolving ontological misunderstandings immediately when they occur. Such mechanisms are also needed to support merging of common ontologies that have evolved independently in different communities, e.g., in different enterprises, business segments etc.

4.5 Pragmatic quality

"Pragmatic quality is the correspondence between the model $M$ and the audience's interpretation of it (I)" (Krogstie 2001). Separation of reference only addresses one aspect (or layer) of meaning, i.e., that of reference. But it is well known that language is not only used to “point at things” in a domain. Language is also used, e.g., to declare or otherwise express social facts and mental states (e.g., Austin 1962). Pragmatic quality is therefore related to the **paradigm dependence** of modelling languages, most of which are used for particular purposes within a particular community with a particular tradition etc. Modelling constructs might therefore mean different things to – and be used differently by – different communities of users. Paradigm dependence is amplified in a long-term, global, integrated setting. Although the presentation part of the UEML approach addresses a few simpler pragmatic issues, this level of pragmatics proper is not dealt with.

It is not obvious that the UEML approach should attempt to deal broadly with pragmatics. An argument by Jørgensen (2004) suggest that high pragmatic agreement is not the same as high pragmatic quality. Jørgensen argues that ambiguity of meaning is an essential characteristic of a live enterprise model, i.e., of any model that is managed and used over time. The reason is that an unambiguous model that has become complete for its purpose and universally understood is no longer a very interesting model. Jørgensen instead promotes interactive models that are deliberately kept incomplete and ambiguous in order to ensure that they remain alive. If Jørgensen's argument holds, it means that some degree of disagreement and confusion over the pragmatic meaning of a model is not a problem, because an ambiguous model encourages communication (about both model and domain) and enforces closer scrutiny (of both model and domain): ambiguity does not decrease, but increases, the value of a model. Ambiguity is beneficial only to a certain degree, however, because a model that is too ambiguous repels its users. And it is essential to restrict ambiguity to the pragmatic level of human communication, while the semantic level of reference remains precise, because technical agents (software tools, intelligent agents...) depends on the referential level without the aid of pragmatics. It remains a challenging task for further UEML work both to investigate Jørgensen's argument and to leverage the potential benefits of pragmatically ambiguous but referentially precise modelling-language and model management.

4.6 Social quality

"[T]he goal defined for social quality is agreement among participants’ interpretations $I$" (Krogstie 2001). Hence, we have already touched central social-quality issues in the two previous sections. An additional central issue is fostering a UEML community that has effective relationships with communities formed around other language families. On the informal side, shared work and communication spaces should be available on the web for learning about, discussing, maintaining and evolving UEML and its approach. On the formal side, UEML and its approach should be managed by a board and promoted as a standard within relevant organisations.

4.7 Other issues

A host of other issues do not fit clearly within the structure of the model quality framework of Lindland et al's and Krogstie's. A particularly important and interesting concern is the balance between model sharing – a necessity for inter-organisational
cooperation, e.g., along value chains – and asset protection – the importance of not sharing valuable information too widely. There may be interesting cryptological challenges here, e.g., encrypting different parts of a partial model so they are available to different partners at a fine-grained level, possibly allowing certain transformations, consistency checks even on the encrypted parts of models without revealing their hidden contents.

5. RELATED WORK

5.1 Unified Modeling Language
The original work on UML focussed on developing a unique modelling language for object-oriented software. While UML is grounded in software, many of its problems are or will eventually be shared by UEML. However, enterprise modelling is a different domain from software modelling. Although UML provides some constructs that can be used to represent enterprises, numerous other phenomena in and aspects of enterprises are not covered at all. Also, the UML does not always clearly distinguish the software domain from other domains. Finally, it is possible that informally combining constructs from different languages is particularly welcome in enterprise modelling situations.

5.2 Meta-Object Facility
The Object Management Group’s (OMG’s, www.omg.org) preferred way of representing languages (or meta models) is the Meta-Object Facility (MOF 1.3 & 2.0, www.omg.org/mof). The behavioural aspects of language constructs are represented in MOF using an object-oriented approach, and (keeping things simple) many of the relationships found in MOF can also be found in the meta-meta model presented here. However, whereas the UMLElement approach is grounded in Bunge’s ontology and the BWW model, MOF appears to focus abstractly on organising the internal structure of languages while emphasising syntactical aspects. The UMLElement approach also provides dynamic and modal concepts.

5.3 Object Management Group
Current OMG-backed modelling-related techniques – such as ECA, OSM-RFP, BPM-RFP and the Business Motivation Metamodel (www.omg.org) – inherit the weaknesses of the MOF and can potentially be beneficially incorporated into UEML. An added value provided by the UMLElement approach is the better structuring of the resulting meta models and the common ontology, from which correspondences with existing languages can be found. Additionally, in UMLElement a language is incorporated as is. UMLElement can benefit from the cross and direct usage of distinct languages defined in several communities, often better elaborated for given requirements (and without the need to redefine constructs).

5.4 Domain-specific languages
The UMLElement approach can be used to create and manage – and thereby promotes – Domain Specific Languages (DSL). In this respect, the approach provides guidelines for designing languages and meta models. It should also support reuse. In contrast, in the current DSL literature, the design principles are often hidden (OOPSLA 2004).

5.5 Ontologies
The relation between the UMLElement work and ontologies deserves further attention. Being based on Bunge’s ontology and the BWW model, the UMLElement approach involves several ontologies. One is the Bunge/BWW ontology, which grounds the other ontologies. Another is the common ontology, which extends the Bunge/BWW ontology. Other ontologies still result from representing modelling languages in terms of the meta-meta model and mapping them onto the common ontology. When the meta-meta model becomes extended to account for presentation, yet more types of ontologies will result.

Ontology research might therefore inform management of the various ontologies involved in the UMLElement approach, as happened when Protégé-OWL was used for UMLElementBase prototype development. Other ontology research might, e.g., inform formalisation of UMLElement. The UMLElement approach is intended for incorporating languages belonging to different paradigms, used by communities and intended for different purposes. Seeing ontologies as a kind of language, the approach may therefore be used to incorporate and manage conventional ontologies. In other cases, co-existence may be a better solution, instead providing transformational bridges between UMLElement and other languages/ontologies – or language/ontology families (Euzenat & Stuckenschmidt 2003). Such ideas are found current proposals (Clark, Evans, Sammut & Williams 2004) and are also related to the OMG’s QVT proposal (www.omg.org, Gardner, Griffin, Koehler & Hauser 2003), although it seems that families of languages in ontologies refer to languages that are of the same type, such as families of increasingly expressive description logics.

5.6 Model management
Model management can be seen as a generalisation of schema management in databases (Bernstein 2003). Like in the UMLElement work, models and languages in this field are only partially formalised. The two fields are also similar in that model management can be seen as a way to compare, compose, integrate or just use together several models. On the other hand, the two fields are different because while schema management can be guided by large amounts of existing instance data, model instance data may be harder to get by. The ideas of both schema integration (Petit 2002) and data integration (Berio 2005) were exploited when defining the first version of the UMLElement. The approaches used to define versions 1 and 2 of UMLElement might be both complementary and combinable in the longer run.

6. CONCLUSION
The experiences from the UMLElement work in Interop indicate that the UMLElement approach for describing modelling constructs is sufficiently powerful to support integrated use of a broad variety of languages and models. The experiences also indicate that the UMLElement approach can form the core of a new theory of integration and integrated use of languages and models. For this to happen, however, the approach must be broadened and developed, formalised and documented further, and the prototype tool must be developed further. The resulting theory and tools also need to be empirically validated and evaluated in real case studies, while incorporating an increasingly wider selection of modelling languages. The discussion and related work sections have pointed to several additional paths for further theoretical and empirical work.

7. ACKNOWLEDGMENTS
This work is partially funded by the Commission of the European Communities under the sixth framework programme: the Interop Network of Excellence, IST-508011. The authors thank all the partners involved in WP5/DEM in the Interop network.
8. REFERENCES


