Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

The Unified Enterprise Modelling Language – Overview and Further Work

Victor Anaya*, Giuseppe Berio, Mounira Harzallah***, Patrick Heymans****, Raimundas Matulevičius****, Andreas L. Opdahl*****, Hervé Panetto******, Maria Jose Verdecho***

** CIGIP, Universidad Politecnica de Valencia, Spain; (e-mail: {vanaya, mverdecho}@cigip.upv.es).*
*** Dipartimento di Informatica, Università di Torino, Italy; (e-mail: berio@di.unito.it)*
**** LINA, University of Nantes, France; (e-mail: mounira.harzallah@univ-nantes.fr)*
***** PReCISE, Computer Science Faculty, University of Namur, Belgium; (e-mail: {phe, rma}@info.fundp.ac.be)*
****** Dept. of Information Science and Media Studies, University of Bergen, Norway; (e-mail: Andreas.Opdahl@uib.no)*
******* CRAN, Nancy-University, CNRS, France; (e-mail: Herve.Panetto@cran.uhp-nancy.fr)*

Abstract: The Unified Enterprise Modelling Language (UEML) aims to support *integrated use of enterprise and IS models* expressed in a variety of languages. To achieve this aim, UEML provides a *hub* through which different languages can be connected, thereby paving the way for connecting the *models* expressed in those languages. UEML offers a structured approach to *describing enterprise and IS modelling constructs,* a *common ontology* to interrelate construct descriptions at the semantic level, a *correspondence analysis approach* to estimate semantic construct similarity, a *quality framework* to aid selection of languages, a *meta-meta model* to organise the UEML and a *set of tools* to aid its use and evolution. This paper presents an overview of UEML and points to paths for further work.

1. INTRODUCTION

Emerging information and communication technologies are increasingly *model-driven*, in part in an attempt to produce solutions that are both *adaptable* and *integrated*. But model-driven information systems are often driven by models that cannot easily be interrelated because they are expressed using languages that are *not interoperable*. The models therefore easily become *inconsistent* as they evolve, and model-driven technologies may end up *reinforcing*, rather than alleviating, existing interoperability problems. The situation creates a need for theories, technologies and tools that allow information systems to be adapted and evolve, each driven by the most suitable languages for their purposes and context, while allowing the information systems and the models that drive them to be used in an integrated manner.

The Unified Enterprise Modelling Language (UEML) refers to an on-going attempt to develop theories, technologies and tools for *integrated use of enterprise and IS models* expressed in different languages. By this we mean keeping the existing models as they are and, in addition, establishing correspondences between them in an explicit and usable way. Examples of useful services are consistency checking, automatic update reflection and model-to-model translation *across modelling language* boundaries. UEML is thereby intended to act as a *hub* connecting different languages along with the different models expressed in those languages. To this end, UEML comprises:

- a structured approach to *describe enterprise and IS modelling constructs,*
- an *evolving common ontology* to describe the semantics of modelling constructs,

- a *correspondence analysis approach* that uses the common ontology to determine semantic correspondences between constructs,
- a *quality framework* to define and evaluate the quality of enterprise and IS modelling languages in order to aid language selection for specific purposes,
- a modular *meta-meta model* to organise the overall UEML approach and
- a *set of tools* to aid its use and evolution.

The purpose of this paper is to present an overview of UEML and discuss paths for further work. The paper is organised as follows: Section 2 presents UEML's *background* and its *vision*. Section 3 explains how languages and constructs are *described* in UEML, whereas Section 4 shows how descriptions of constructs are tied together by a *common ontology*. Section 5 discusses how *correspondences* between languages and constructs can be established and used, e.g., to support model-to-model translation across languages. Section 6 shows how enterprise and IS modelling languages are *classified and selected* in UEML according to specific goals. Section 7 presents the *meta-meta models* that holds the UEML approach together, whereas Section 8 reviews the various *prototype tools* supporting its use and evolution. Section 9 discusses UEML in its present state, before Section 10 concludes the paper and offers paths for further work.

2. BACKGROUND

The idea of a Unified Enterprise Modelling Language first emerged during the ICEIMT'97 conference (Goossenaerts, Gruninger, Nell, Petit & Vernadat 1997), with the aim of providing an underlying formal theory for enterprise modelling languages. A major motivation was the "Tower of Babel" situation that was assumed to hinder proliferation of enterprise modelling in industry (Vernadat 2002). The first development version of a unified enterprise modelling language was delivered by the *UEML Thematic Network* (UEML TN) (2002-2003), funded by the EU's FP5 (Jochem 2002, Panetto, Berio, Benali, Boudjlida & Petit 2004, Mertins, Knothe & Zelm 2004, Berio, Anaya & Ortiz 2004). UEML development has since continued within the Interop-NoE (2003-2007) Network of Excellence, funded by EU's FP6, producing two more development versions, UEML 2.0 and 2.1 (Berio, Opdahl, Anaya & Dassisti 2005a, 2005b, 2006).

The following scenarios illustrate the UEML vision:

- *Exchanging information contained in enterprise and IS models* across modelling language boundaries. This is the central motivation behind UEML, which explains its focus on *interoperability between modelling languages* as a prerequisite for integrated use of the models that are expressed in those languages.
- *Creating new problem- and/or domain-specific methods* by combining elements from existing modelling techniques. UEML aims to make it easier to combine modelling languages and associated techniques and tools depending on the problem at hand, an ambition resembling that of *method engineering*. In particular, UEML aims to support *local tailoring/adaptation* of languages and constructs to fit local practices and needs, possibly producing new *domain-specific languages* as a result.
- *Systematic, quality-driven, reuse of existing enterprise and IS modelling languages.* Combining techniques and tools across modelling languages has the side benefit of making the languages available for the domains where they are most suited, without limitations posed by modelling tools and other technologies.
- *Defining a core language for enterprise and IS modelling.* As UEML stabilises, it may become possible to extract a core set of modelling construct to use as the starting point for a new enterprise/IS modelling language. Such a *UEML core language* should be composed of those constructs that have proven most useful for practical, integrated model use. However, the core language scenario should be understood as a longer term objective that is beyond the scope of this paper.
- Facilitating a *web of languages* and of *models* is another long-term objective. Whereas much research and development effort has gone into techniques and tools for integrated management of structured data (e.g., relational database theory) and of semi-structured data (e.g., XML and other web technologies), there is a lack of theory and technology for integrating information resources in the form of diagrammatic *models*. UEML could contribute to growing a *web of languages* and *of models* in a way that resembles the touted semantic web of semi-structured data (Berners-Lee, Hendler, Lassila 2001).

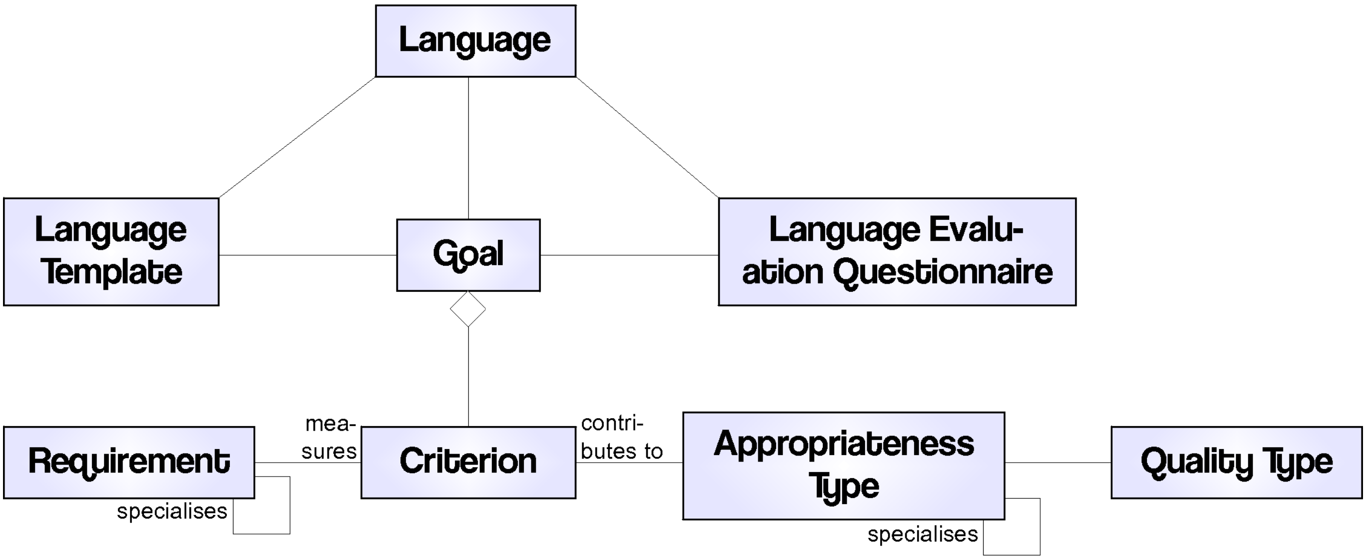

Figure 1: The main classes of the UEML representation meta-meta model, used to describe the semantics of modelling constructs.

Although UEML is intended as a hub for connecting different languages and different models expressed in those languages, it will not necessarily be the only means of making enterprise and IS languages and models interoperable. Other theories, technologies and tools may be better suited for certain integration needs and should possibly be made usable alongside UEML.

3. LANGUAGE AND CONSTRUCT DESCRIPTION

UEML facilitates integrated model use by making *semantic correspondences* between the modelling constructs of different languages clear. Making the languages interoperable is seen as a first step towards also making the models expressed in those languages interoperable. A central part of UEML is therefore a standard, integrative and evolvable approach to *describing enterprise and IS modelling constructs.* By *standard* we mean that the approach provides a structured path to describing modelling languages, diagram types and constructs. By *integrative* we mean that, as soon as the languages, diagram types and constructs have been described according to the approach, they have also become prepared for assessment of semantic correspondences, possibly across languages. And by *evolvable* we mean that UEML will be able to grow and adapt by incorporation and modification of additional modelling languages and constructs without becoming overly complex and thus unmanageable.

The descriptions of individual modelling constructs are particularly important, because it is this level that connects different modelling languages. Hence construct descriptions are more complex than descriptions of languages and diagram types. Specifically, in UEML, two distinct descriptions need to be made for each construct (Opdahl 2006):

- *Presentation* (or *concrete syntax*), which deals with the presentation of the modelling construct as part of model diagrams or in serialised form, e.g., in an XML file.
- *Representation* (or *semantics*), which accounts for which enterprise phenomena the construct is intended to represent (in particular covering *reference*, a central aspect of *semantics*).

Whereas a construct can have many presentations, it can have only one representation. This paper will focus on the representation part, which has so far been more developed than presentation.

In UEML, semantics is described by a *representation mapping* of each modelling construct into a *common ontology*, based on earlier work by Opdahl & Henderson-Sellers (2004, 2005). The UEML approach uses *separation of reference* to break individual modelling constructs into their *ontologically atomic parts,* along the following six axes:

1. *Which class(es) of things is the construct intended to represent?* Most modelling constructs somehow represent one or more *classes of things*. Even when the *primary* purpose of a construct is to represent certain properties, states or transformations, the construct implicitly also represents a property of, state of or transformation in, *one or more classes of*

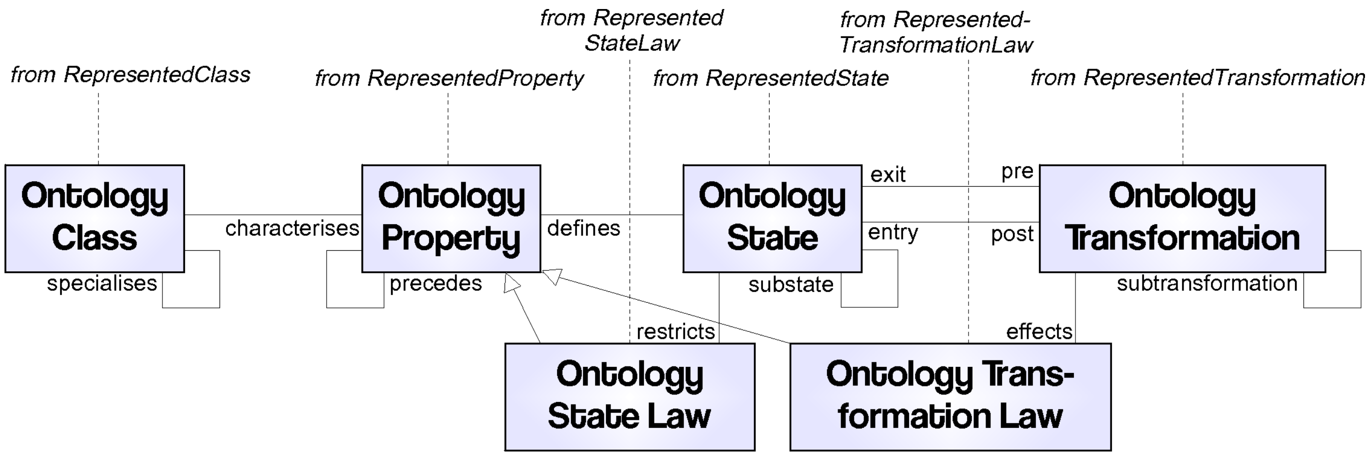

Figure 2: The main classes of the common UEML ontology, into which construct descriptions are mapped.

things. (A transformation may correspond either to an atomic event or a complex process.)

2. *Which properties is the construct intended to represent?* Most modelling constructs somehow represent one or more *types of properties*, which may either be *intrinsic properties (*belonging to only one thing) or *relationships* (properties that are *mutual* to several things). Some intrinsic properties are *laws* that restrict other properties. Even if the primary purpose of a construct is to represent classes, states or transformations, it represents classes, states or transformations that involve *one or more types of property*.
3. *Which states is the construct intended to represent?* Some modelling constructs are intended to represent a more or less restricted state in one or more classes of things. The *state law* that restricts the state can be described in terms of the properties of those classes. Whereas most modelling constructs represent one or more properties and, at least, one or more classes, not all constructs are intended to represent a state.
4. *Which transformations is the construct intended to represent?* Some constructs are intended to represent a simple or complex transformation of one or more classes of things from one state to another. The *transformation law* that effects the transformation can be described in terms of the states of those classes. Again, not all constructs are intended to represent a transformation. Although some constructs are apparently not intended to represent behaviour at all, other constructs represent particular *states, transformations*.
5. *Which instantiation levels is the construct intended to represent?* A modelling construct represents classes, properties, states and transformations at either the instance or type level or both.
6. *Which modality (or mode) is the construct intended to represent?* We usually think of enterprise and IS models as *assertions* of *facts* about a domain, e.g., assertions that something is or is not the case in the enterprise. But some model elements may instead state that someone *wants* something to be the case, or that someone is not *permitted* to do something, or that someone *knows* something is the case¸ or that something *will be* the case some time in the *future*.

Hence, whereas the two first axes deal with *structure*, the next two deal with *behaviour*. Together, these four axes describe the semantics of a modelling construct by describing a *state of affairs*, or a *scene,* played by several classes, properties and, perhaps, states and transformations together. The final two axes supplement the scene with information about the construct's intended *use*, i.e., its instantiation level and modality/mode.

The UML class diagram in Figure 1 shows the key concepts used to describe modelling languages and constructs in UEML. The upper part of the diagram depicts *modelling languages*, along with their *diagram types* and *modelling constructs*. The lower part shows how each individual construct is described by a *scene* of interrelated classes, properties, states and transformations.

4. THE COMMON ONTOLOGY

To tie modelling-construct descriptions together, UEML uses a common ontology into which the represented classes, properties, states and transformations of each construct are mapped. The common ontology thereby comes to interrelate the construct descriptions at the semantic level.

The UEML ontology is organised into four *taxonomies*: The classes in the ontology are organised in a conventional *generalisation hierarchy*. Properties, on the other hand, have their places in a *precedence hierarchy*, where a property precedes another if every thing that possesses the second property necessarily also possess the first. (For example, *associated-with* precedes *having-content*, because everything that is *having-content* is also *associated-with* that content.) There are similar generalisation hierarchies of states and of transformations too. Classes, properties, states and transformations – including the state and transformation laws – all have *attributes*. For example, they all have unique names and there are cardinality constraints and role names on the associations between classes and properties.

The four taxonomies are interrelated. Classes are related to the properties that *characterise* them. Properties are related to the states they *define.* States are in turn *entered* and *exited* by transformations. Certain types of properties are laws that restrict other properties. State laws *restrict* states, whereas transformation laws *effect* transformations. The resulting organisation of the UEML ontology as four distinct, but interrelated taxonomies makes it possible to evolve the ontology over time without increasing complexity more than necessary. New classes, properties, states and transformations will always have a clearly identifiable location where they can be added to the appropriate taxonomy.

The UML class diagram in Figure 2 shows the key concepts of the common ontology, based on the earlier work of Opdahl & Henderson-Sellers (2004, 2005). For every construct incorporated into UEML, each represented class, property, state and transformation is mapped into an ontology concept in the ontology. Figure 2 therefore structurally resembles the lower part of Figure 1.

The UEML ontology was first populated with a set of initial classes, properties, states and transformations derived directly from Mario Bunge's ontological model (Bunge 1977, 1979) and the Bunge-Wand-Weber representation model of information systems, the so-called BWW model (Wand & Weber 1988a, 1988b, 1993, 1995). Since then, it has evolved and grown as new constructs have been added. Currently, UEML incorporates a selection of academic and industrial modelling languages, such as ARIS (Dossogne & Jeanmart 2007), BMM (Tu 2007), BPMN (Dossogne & Jeanmart 2007), coloured Petri nets, GRL (Dallons, Heymans & Pollet 2005, Heymans, Saval, Dallons & Pollet 2005, Matulevičius, Heymans & Opdahl 2006, 2007a, Tu 2007), IDEF3 (Harzallah, Berio & Opdahl 2007), ISO/DIS 19440, KAOS (Matulevičius, Heymans & Opdahl 2006, 2007a, 2007b), UEML 1.0 and selected diagram types from UML 2.0. In consequence, the most general concepts in the common ontology are *ontologically committed*, in the sense that they have grown out of Bunge's ontology and the BWW model, whereas the more specific ones have emerged through language and construct analyses.

5. LANGUAGE AND CONSTRUCT CORRESPONDENCES

To support integrated use of models, UEML must offer ways to exploit the mappings to *identify and manage correspondences* among language constructs and among model elements. Correspondences between any pair of constructs can be examined by comparing their mappings into the common ontology. If two modelling constructs are identical, they will map into the exact same ontology concepts. If two modelling constructs do not overlap at all, they will map into concepts that are not closely related in their respective taxonomies. However, the most common situation will most likely be where the modelling constructs map into some common ontology concepts, into some concepts that are closely related and into some that are not.

Three kinds of correspondences have been identified. Each of them can be precisely formulated in terms of the ontology classes, properties, states and transformations into which the constructs in the correspondence map:

- *Equality* occurs when two or more constructs represent the exact same state of affairs, as explained in Section 3. If two constructs are equal, one can always replace the other without loss of information, e.g., for model-to-model translation.
- *Containment* occurs when the state of affairs represented by one construct has the state of affairs represented by another as a part. When one construct contains several others, the former can be replaced by a combination of the others during model-to-model translation.
- *Generalisation* occurs when one modelling construct represents a state of affairs that generalises the state of affairs represented by another. When one construct generalises another, the general construct can replace the special one in a model-to-model translation (with some loss of information), but the inverse replacement is only appropriate under specific circumstances.

Of course these simple kinds of correspondences are not independent. For example, constructs that are equal will trivially contain and generalise one another. There are also *complex correspondences*, e.g., when one construct

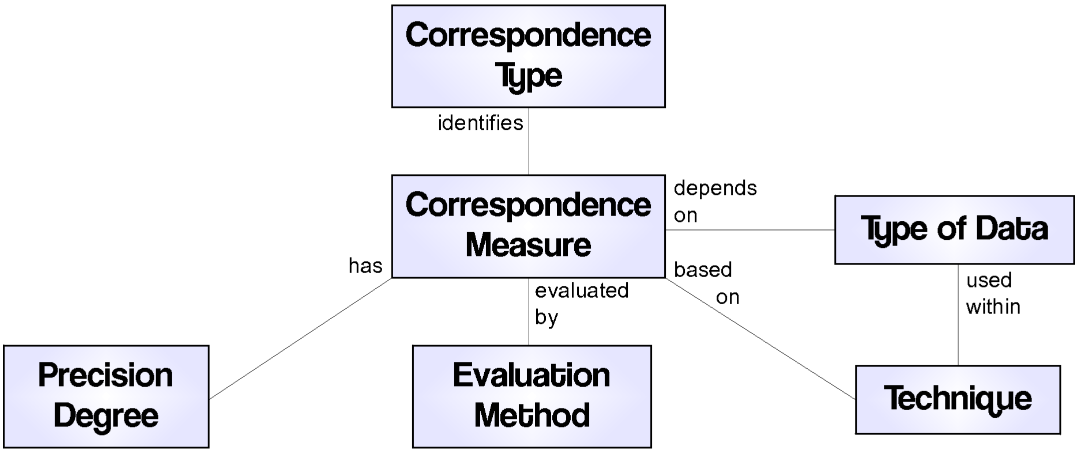

Figure 3: The main classes of the UEML correspondence analysis framework, used to identify and manage correspondences between modelling constructs.

represents a state of affairs that *generalises* a *part* of the state of affairs represented by another, thus combining containment and generalisation. There are also *overlapping* constructs, each of which contains part, but not all, of the other. However, a complete typology of correspondences and how they combine stills needs to be worked out.

Correspondences are also characterised by different *degrees of precision*. For example, it is possible to only take into account how each construct is mapped into ontology concepts, ignoring how the concepts are *related* within the construct description. More precise correspondences can be identified by taking into account both ontology concepts and the relations between them, but ignoring the *roles* that the concepts may play in the relations. Finally, both the ontology concepts, the relations between them and the roles played can be taken into account.

Because correspondences are generally ways to assess to what extent constructs are similar or dissimilar (under a precision degree), it might be possible to characterise correspondences by using correspondence measures (CM). In this sense, a CM is a function:

$$CM: UEMLC \times UEMLC \rightarrow \Re_{+}$$

Where UEMLC represents the set of constructs incorporated into UEML. CM results from explicit selections ranging on the following five parameters: correspondence type, precision degree, technique, type of data and evaluation method, as shown in Figure 3. The form of the function is related to the *correspondence type* to be identified (e.g. equality). There are three well-known forms of function that can be used, i.e., Jaccard, Recall and Precision (Gower & Legendre 1986). The *type of data* taken as input by the function is varying depending on the *precision degree* required. Type of data constrains the specific *technique* that can be used for effectively evaluating function results as well (using, e.g., structure-based, graph-based or attribute-based techniques) (Lin 1998, Rodrýguez & Egenhofer 2003, Blanchard, Kuntz, Harzallah & Briand 2006). Finally, the measure should be meaningful. To this end, an *evaluation method* should be defined to evaluate the measurement accuracy. According to Budanitsky (1999), accuracy of the measure can be evaluated by comparing the measurement results with correspondences found in three alternative and distinct ways: 1) theoretical investigation 2) human judgement and 3) knowledge about a particular application.

Correspondence measures can also be used to *validate* the representation mappings and the common ontology, when correspondence measures derived automatically from the common ontology is compared to expert estimates of the same correspondences. Deviations indicate that the representation mapping for a construct is wrong and/or that there are weaknesses in the common ontology. There may be concepts missing from the common ontology, or there may be taxonomical relations between ontology concepts missing, e.g., a missing generalisation relation from a sub- to a superclass. If left undetected, missing taxonomical relations can lead to redundancies in the common ontology when the same subclass is added several times because it cannot be retrieved as a specialisation of its superclass. In this way, correspondence measures can also aid *elimination of redundancy* in the common ontology.

Correspondence measures as representative of correspondences are useful as *high-level guides* for model-to-model translation and other cross-language services. The representation mappings and common ontology provide the details for how to translate between modelling constructs belonging to different languages, as soon as the pair of modelling constructs to translate between have been decided. But it offers less help with deciding which constructs in one language to translate into which constructs in another. The correspondence measures can help by suggesting, for each construct in a language, which constructs in the other language that are most suitable as targets for, e.g., translation, leaving the final choice to be made by the model manager. When the construct-to-construct correspondences at the language-level have been established in this way, the representation mapping and common ontology are there to support the detailed construct-level mappings.

6. LANGUAGE QUALITY FRAMEWORK

Together, the representation mappings, common ontology and correspondence measures contribute towards integrated use of models expressed in different languages. But there is also a need to select suitable languages to include in the UEML in the first place. For example, to quickly enrich the common ontology, it may be better to incorporate a much used and relatively complete language early on than a narrower language used only by specific communities. Later, when using UEML, there is also a need to select suitable languages for particular purposes among the many available. UEML therefore includes a *language quality framework* (Anaya, Berio & Verdecho 2007) that aids language selection by:

- defining the *concept of quality* of a modelling language;
- supporting *methodical, goal-dependent evaluation* of the quality of enterprise and IS modelling languages.

The current quality framework has adapted and extended the SEQUAL quality framework (Krogstie 1998, 2005), which provides a model of the quality of models, later extended to also account for the quality of languages. SEQUAL identifies 8 *quality types* for characterising what quality is: physical quality, empirical quality, syntactic quality, semantic quality, perceived semantic quality, pragmatic quality, social and organisational quality. For example, *semantic quality* is the correspondence between the model and the domain. SEQUAL also identifies several *types of appropriateness*, each indicating a language aspect that must be considered when assessing whether a language is appropriate for a particular purpose (Krogstie 1998, 2005). For example, *comprehensibility appropriateness* reflects the ease with which the language its model can be understood by a certain audience. In SEQUAL, each quality type is related to one or more appropriateness types and vice versa. For example, domain appropriateness is used to assess physical and semantic qualities. Therefore, the different types of appropriateness provide the context for evaluating the related quality types.

In addition to SEQUAL, the UEML quality framework has been inspired by two additional quality frameworks: Moody's framework (2003) and ISO/IEC 9126 international standard for assessing software product quality (ISO/IEC 2001). These two frameworks have been adapted and aligned with SEQUAL's appropriateness types through a generalisation hierarchy (Berio, Opdahl, Anaya & Dassisti 2005b).

The resulting *appropriateness types* in UEML's quality framework remain too general to allow concrete evaluations (Anaya, Berio & Verdecho 2007). Therefore, the framework also covers *requirements* and *criteria*. Requirements are collected from users (actors or experts), asking them how enterprise modelling should contribute towards enterprise integration and interoperability, based on a requirements base established in the previous UEML Thematic Network (UEML-TN 2002-2003). Criteria are the *operational*, or *measurable*, counterparts of requirements. Each criterion can in turn be related to one or more *appropriateness types*, making it clear to which quality types the criterion contributes. The framework provides two complementary ways of collecting data for evaluating criteria. The *language template* is used to gather general and factual information about a language, such as its notations and meta models, whereas the *language-evaluation questionnaire* comprises both questions derived from current criteria and an associated glossary (Verdecho & Matulevičius 2007).

The framework also covers *language descriptions*, which cover, e.g., a language's owner and version; g*oals,* which are aggregations of criteria for the purpose for evaluating language quality; *metrics-for-goal,* which are selected metrics relevant to a specific goal (metrics are needed to perform criteria assessment); *metric evaluations,* which are specific evaluations (for instance, a value) of a single metric on a specific language; *combined metrics evaluations*, which are combined evaluation of several metrics evaluations for a given language and a given goal (an explicit combined metrics evaluation makes explicit how several single metrics are combined, e.g., with a weighted formula, to evaluate the quality of a language with respect to a given goal; additionally, it is useful because the same metrics evaluation can be used several times if needed).

The UML class diagram in Figure 4 shows the key concepts used to evaluate the quality of modelling languages in

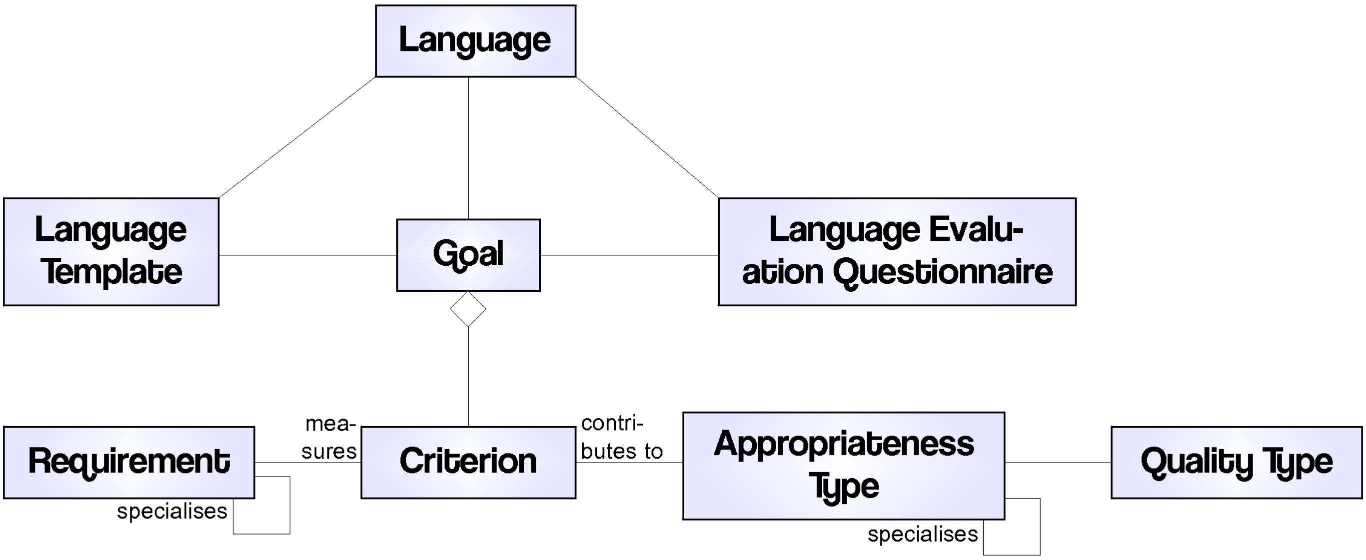

Figure 4: The main classes of the UEML language quality framework, used to classify and select modelling languages.

UEML. The associated *quality evaluation method* gives a clear picture of how to evaluate and select one or more enterprise and/or IS modelling languages for a specific purpose. The first task is to define the goal as an aggregation of criteria and then select suitable metrics for each criterion. A list is made of languages to be evaluated. The language template is used to collect factual information about each language, and the language-evaluation questionnaire is used to collect subjective opinions. Hence, whereas only a single filled-in language template is needed for each language, multiple filled-in questionnaires from language users are usually needed. Once the selected criteria are assessed by using selected metrics and storing these assessments as metrics evaluations, combined metrics evaluations are calculated and stored. Finally, languages must be suitably selected based on the results stored as combined metrics evaluations. Before its use, an enterprise may undertake a customisation of the quality framework: This simply means to define additional requirements, appropriateness types, criteria and metrics.

7. META-META MODELS

The UML class diagrams of the language and construct description approach (Section 3), of the common ontology (Section 4), of the correspondence analysis approach (Section 5) and of the quality framework (Section 6) are all *meta-meta models.* They are meta-meta models because models of modelling languages are meta models and because Figures 1-4 are models of how to model aspects if modelling languages (thus of how to model meta models). The UML diagrams are intended as illustrations only. For example, Figures 1-2 do not show attributes and omit several association classes and abstract classes. More detailed meta-meta models can be found in (Opdahl 2006).

Whereas the representation mappings connect Figures 1 and 2, the meta-meta models of the correspondence analysis and language quality frameworks in Figures 3 and 4 are currently connected to Figure 1 only through the *language description* in Figure 4. Further work should establish a single combined, yet modular, meta-meta model that covers all constituents of the UEML approach, the *overall UEML meta-meta model.*

8. TOOLS

UEML is supported by a set of prototype tools realised using a selection of existing technologies. There are currently five tools in the set:

- *UEMLBase Repository* is a Protege-OWL realisation of the representation and ontology meta-meta models of Figures 1-2, translated into OWL.
- *UEMLBase Editor* is an emerging set of Eclipse GMF-based editors for browsing and updating the contents of the UEMLBase repository.
- *UEMLBase Manager* is a Java-plugin for Protege-OWL that provides merging, reporting and other housekeeping functions for the repository.
- *UEMLBase Verifier* is a set of Prolog rules and a Prolog rule checker that support formal verification of the contents in the UEMLBase repository, for example to check cardinality constraints and ensure that the construct descriptions are concrete. Prolog was chosen instead of newer technologies, such as

SWRL, because of its high availability, robustness and general versatility (Mahiat 2006).

- *UEMLBase Correspondence Analyser* uses the repository to compute similarity measures between UEMLBase constructs, based on the meta-meta model in Figure 4, thus paving the way for consistency checking, automatic update reflection, model-to-model translation across languages, as well as other integrated model uses.

Each tool strives to be consistent with the meta-meta models of Figures 1-4, although they realise more specific implementation models, such as OWL, Eclipse EMF, Java classes and Prolog facts. Hence, the meta-meta models is used to support interoperability within the UEML tool set.

9. DISCUSSION

The paper has presented the main constituents of the UEML approach and explained how they are related. Languages, possibly selected with the aid of the quality framework, are described using separation of reference according to Section 3. The descriptions of the states of affairs are then mapped into the common ontology of Section 4. It thereby becomes possible to establish correspondences between different constructs in terms of their mappings into the common ontology as in Section 5. The selection of modelling languages is guided by the quality framework of Section 6. In the long term, the most used and useful concepts in the common ontology can be used to form a *core UEML language* for enterprise and IS modelling. In the long tern, UEML could also contribute towards developing a *web of languages and of models* in a way that resembles the touted semantic web of semi-structured data (Berners-Lee, Hendler & Lassila 2001), which is currently emerging in areas such as *e-science* and *e-government* (Shadbolt, Hall & Berners-Lee 2006).

From an initial set of around 25 concepts taken more or less directly out of Bunge's ontology and the BWW model, the common UEML ontology has grown to comprise 110 concepts. Most of them have resulted from analyses of individual modelling constructs using separation of reference. (A few initial higher-level remain to organise and structure the four taxonomies.) As part of the Interop-NoE work, 130 constructs from the following 10 languages have been mapped into this ontology: ARIS, BMM, BPMN, GRL, IDEF3, ISO/DIS 19440, KAOS, coloured Petri nets, UEML 1.0 and selected diagram types from UML 2.0. However, they are not all described in equal detail and none of them are yet fully validated. The languages, constructs, mappings and ontology have all been stored in the UEMLBase Repository, supported by the Editor, Manager, Verifier and Correspondence Analyser tools.

The standardised approach to language and construct description has turned out to have several advantages, in particular at the modelling construct level. The structured descriptions become complete, consistent, cohesive and, thus, more learnable and understandable. It therefore becomes easier to compare them to one another. The structured approach also offers systematic and detailed advice on how to proceed when analysing individual language constructs. It encourages highly-detailed construct description, which leads to languages that are integrated at a fine level of detail. It supports ontological analysis in terms of *particular* classes, properties, states and events, and not just in terms of the *concepts* in general.

The UEML approach has *positive network externality*, in the sense that incorporating an additional construct or language becomes:

- *more valuable* the more constructs and languages that have already been incorporated, because the additional language becomes interoperable with a larger number of other languages;

- *less costly* because reusing an enriched common ontology and existing representation mappings provide good reference examples and because the cost of maintaining tools and infrastructure can be shared by more UEML users.

Similar positive network externality effects can be expected at the *model level* beside the language level discussed here.

Early experience with the construct description approach indicated that it was difficult to use because it was based on a novel, unconventional way of thinking about the semantics of modelling constructs. It was sometimes hard to find the appropriate classes, properties, states and events in the common ontology to use when describing a construct. Also, it was sometimes hard to determine exactly which part of a language that constitutes a modelling construct. As part of the Interop-NoE, tools and tutorials were developed that have seemingly resolved many of these problems. Also, early drafts of the common ontology have become available along with exemplary representation mappings. As a result, the first draft of several of the most recent language incorporations could be made by students with little direct supervision.

The framework for selecting and evaluating the quality of modelling languages according to specific goals also provides high benefits for users that need to decide about languages to use for practical purposes. First, it gets the *voice of the customer* through the consideration of the requirements of the users making them to appear in the front end of the framework. Then, these requirements are related to criteria that make them operational and applicable to the language evaluation.

10. CONCLUSION AND FURTHER WORK

UEML is an ambitious, long-term effort that will require several years of cooperation between academia and industry. The overall challenge for further work is to extend the theory and tools developed by the Interop-NoE network to support *practical integrated use of models and languages*. Although several limited paper-and-pencil trials have demonstrated the feasibility of the approach (Berio, Opdahl, Anaya, Dassisti, 2005b; Matulevičius, Heymans, Opdahl, 2007; Harzallah, Berio, Opdahl, 2007), detailed methods for integrated model use still need to be developed and implemented.

For UEML-supported integrated model use to be tested in large-scale, realistic settings, the common ontology and representation mappings must be verified, validated and improved. The current ontology and mappings have been contributed by several Interop-NoE research teams working in a distributed manner. The most immediate challenge is to improve the ontology and mappings in two directions. Firstly, the Editor and Verifier tools are being extended and improved. Secondly, the Correspondence Analyser tool is used to compare correspondences calculated from the common ontology and the representation mappings with correspondence estimates provided by human experts. The comparisons are used to identify weaknesses in the representation mappings. For example, when two constructs are considered similar by human experts, but not by the Correspondence Analyser, the reason might be that one or more ontology concepts have been duplicated. Accordingly, when the Analyser, but not the human experts, deem two constructs similar, the reason may be weaknesses in the generalisation hierarchies in the ontology. In this way, verification not only supports improving the representation mappings but also controls the quality of the common ontology.

As for the overall UEML approach, an obvious path for further work is to connect the meta-meta models for language and construct description and for the common ontology with the one for the quality framework. Also, the combined meta-meta model must be extended to account for the presentation part of language and construct description and for construct correspondences. In addition to tying together the overall approach, this work can be expected to reveal further possibilities, such as deriving quality and appropriateness metrics for languages, not only at the language level, but also at the construct level from the detailed UEML ontology and mappings.

These and other possible future developments have been organised in a *UEML roadmap* comprising several research directions, each detailed by specific actions (Opdahl & Berio 2006b): 1. Language breadth – include more languages; 2. Ontological depth – refine the common ontology; 3. Ontological clarity – elaborate the common ontology language; 4. Presentation – extend the support for presentation issues; 5. Mathematical formality – define UEML semantics formally; 6. Tool support – develop prototype tool with GUI and validation support; 7. Model management – provide support for model management in addition to language management; 8. Validation – structural and behavioural language and model validation; 9. Dissemination – make UEML known in industry and academia and as a standard; 10. Community – establish and maintain a committed and cohesive community for managing and evolving UEML and its approach. Additional directions that deal specifically with the language quality framework are: 1. Continuing the development of the quality framework by introducing new criteria and extending the questionnaire accordingly; 2. Continuing the accommodation of existing quality frameworks by specialising appropriateness; 3. Gradually developing supporting tools based on the meta-meta model, starting from the current simple support for filling-in the questionnaire to complete functionality to define and evaluate metrics; 4. Launching use of the quality framework and especially by performing evaluations of languages for developing a core language. For example, more specific quality frameworks can be used to systematically introduce new appropriateness measures and to specialise existing ones. The roadmap still needs to be extended to account better for correspondence analysis.

The UEML approach may even be useful outside enterprise and IS modelling, e.g., for software modelling. Significantly, only the language quality framework is specific to enterprise modelling. The other major UEML parts might be used for a wider set of modelling domains.

ACKNOWLEDGMENT

The authors are indebted to all the partners of Interop-NoE and, in particular, to the researchers participating in its *Domain Enterprise Modelling*. The authors are indebted to the many students who have contributed to the work, including Emmanuel Blanchard, Aurelie Dossogne, Cedric

Jeanmart, Alf Harry Karlsen, Jeremy Mahiat, Christophe Tu, Torbjørn Vefring and Tomas Zijdemans. This work was partially funded by the Interop Network of Excellence and by the Interuniversity Attraction Poles Programme, Belgian State, Belgian Science Policy.

REFERENCES

Anaya, V., Berio, G., and Verdecho, M.J. (2007). Evaluating Quality of Enterprise Modelling Languages: The UEML Solution. *Proc. I-ESA 2007*, Funchal, Portugal.

Berio G., Anaya V., and Ortiz A. (2004). Supporting Enterprise Integration through a Unified Enterprise Modeling Language. In *Proc. of EMOI 2004 (Enterprise Modelling and Ontologies for Interoperability),* Grundspenkis, J., Kirikova, M. (eds.), joint with CAiSE*04, Riga Technical University, 3:165-176.

Berio, G., Opdahl, A., Anaya, V. and Dassisti, M. (2005a). Deliverable DEM1. Publicly available at www.interop-noe.org.

Berio, G., Opdahl, A., Anaya, V., and Dassisti, M. (2005b). Deliverable DEM2. Publicly available at www.interop-noe.org.

Berio, G., Opdahl, A., Anaya, V., and Dassisti, M. (2006). Deliverable DEM3. Publicly available at www.interop-noe.org.

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. Scientific American Magazine - May, 2001

Blanchard E., Kuntz P., Harzallah M. and Briand H. (2006). A tree-based similarity for evaluating concept proximities in an ontology. In Proc. 10th Conf. of the International Federation of Classification Society, pp.3–11. Springer.

Budanitsky, A. (1999). Lexical semantic relatedness and its application in naturallanguage processing, Technical report, Univ. of Toronto

Bunge, M. (1977). Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World. Boston:Reidel.

Bunge, M. (1979). Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems. Boston:Reidel.

Dallons, G., Heymans, P. and Pollet, I. (2005). A Template-based Analysis of GRL, in Proc. of EMMSAD'05 (CAiSE*05), Tenth International Workshop on Exploring Modeling Methods in Systems Analysis and Design, pp. 493-504.

Dossogne A. and Jeanmart C. (2007) Evaluation of ARIS and BPMN using the UEML approach. Master thesis, University of Namur.

Goossenaerts, J., Gruninger, M., Nell, J.G., Petit, M. and Vernadat, F. (1997). Formal Semantics of Enterprise Models. In *Proc. of ICEIMT'97*, K.Kosanke and J.G Nell. (Eds.), Springer- Verlag.

Gower, J.C. and Legendre, P. (1986). Metric and euclidean properties of dissimilarity coefficients', J. of Classification, 3, 5–48.

Harzallah M., G. Berio, et A.L. Opdahl (2007). Incorporating IDEF3 into the Unified Enterprise Modelling Language (UEML). In Proc. VORTE 2007, joint with EDOC07.

Heymans, P., Saval, G., Dallons, G. and Pollet, I. (2005). A Template-Based Analysis of GRL: Book chapter, in Advanced Topic in Database Research - Volume 5. Idea Group Publishing.

INTEROP-NoE (2003-2007). Interop Network of Excellence. www.interop-noe.org, 2003-2007.

ISO/IEC Standard 9126 (2001). Software product quality, International Standards Organisation (ISO). International Electrotechnical Commission (IEC).

Jochem, R. (2002). Common representation through UEML – requirement and approach. In *Proc. of ICEIMT 2002,* Kosanke K., Jochem R., Nell J., Ortiz Bas A. (Eds.), Polytechnic University of Valencia, Valencia, Spain, April 24-26, Kluwer. IFIP TC 5/WG5.12.

Krogstie, J. (1998). Using a Semiotic Framework to Evaluate UML for the Development for Models of High Quality. Siau K., Halpin T., (eds) Unified Modelling Language: System Analysis, Design and Development Issues, IDEA Group Publishing, pp. 89-106.

Krogstie, J. (2005). Evaluating UML Using a Generic Quality Framework. Encyclopaedia of Information Science and Technology. M. Khosrow-Pour Editor, IDEA Group Publishing.

Lin D. (1998). An information-theoretic definition of similarity. In Proc. 15th International Conference on Machine Learning, pp. 296–304. Morgan Kaufmann.

Mahiat, J. (2006). A Validation Tool for the UEML Approach. Master thesis, University of Namur.

Matulevičius R., Heymans P., Opdahl A. L. (2006). Comparison of Goal-oriented Languages using the UEML Approach. In *Interoperability for Enterprise Software Applications*, Panetto H., Boudjlida N. (eds), pp 37-48. ISTE.

Matulevičius R., Heymans P., Opdahl A. L. (2007a). Comparing GRL and KAOS using the UEML Approach. In *Enterprise Interoperability II. New Challenges and Approaches*, Concalves, R. J., Muller, J. P., Mertins, K., Zelm, M. (eds.), pp 77-88. Springer-Verlag.

Matulevičius R., Heymans P., Opdahl A. L. (2007b). Ontological Analysis of KAOS Using Separation of Reference. In *Contemporary Issues in Database Design and Information Systems Development*, Siau K. (ed.), pp. 37-54. IGI Publishing.

Mertins, K., Knothe, T., Zelm, M. (2004). User oriented Enterprise Modeling for Interoperability with UEML. In *Proc. of EMMSAD'04*

(Evaluating Modeling Methods for Systems Analysis and Design), pp.25-36, joint with CAiSE*04, Riga – Latvia, June 7-8.

Moody DL (2003) Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice. Proc. ECIS'2003, Naples, Italy.

Opdahl, A.L. (2006). The UEML Approach to Modelling Construct Description. Proc. 2nd International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2006).

Opdahl, A.L. and Berio, G. (2006a). Interoperable Language and Model Management using the UEML Approach. Proc. G@mma 2006 (International Workshop on Global Integrated Model Management), pp. 35-42.. ACM Press.

Opdahl, A.L. and Berio, G. (2006b). A Roadmap for the UEML. Proc. 2nd International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2006).

Opdahl, A.L. and Henderson-Sellers, B. (2004). A Template for Defining Enterprise Modelling Constructs. Journal of Database Management 15(2).

Opdahl, A.L. and Henderson-Sellers, B. (2005). Template-Based Definition of Information Systems and Enterprise Modelling Constructs. In Ontologies and Business System Analysis, Peter Green and Michael Rosemann (eds.). Idea Group Publishing, 2005.

Panetto H., Berio G., Benali K., Boudjlida N. and Petit M. (2004). A Unified Enterprise Modelling Language for enhanced interoperability of Enterprise Models. In *Proc. of the 11th IFAC INCOM2004 Symposium*, Bahia, Brazil, April 5-7.

Rodrýguez M., Egenhofer M (2003). Determining semantic similarity among entity classes from different ontologies. IEEE Transactions on Knowledge and Data Engineering 15(2), pp. 442–456.

Shadbolt, N., Hall, W. and Berners-Lee, T. (2006). The Semantic Web Revisited. *IEEE Intelligent Systems,* May/June 2006.

Tu, C. (2007). Ontological Evaluation of BMM and i* with the UEML Approach. University of Namur, Master thesis.

UEML-TN (2002-2003) Unified Enterprise Modelling Language Thematic Network. 2002-2003.

Verdecho M.J. and Matulevičius, R. (2007). Language Evaluation Questionnaire for Enterprise Modelling Languages. Unpublished.

Vernadat, F. (2002). UEML: Towards a Unified Enterprise Modelling Language. International Journal of Production Research, 40(17):4309-4321, Taylor & Francis Group.

Wand, Y. and Weber, R. (1988a). An ontological analysis of some fundamental information systems concepts. In Proc. Ninth International Conference on Information Systems, DeGross, J.I. and Olson, M.H. (eds.), Minneapolis/USA, November 30–December 3, 1988, pp. 213–225.

Wand, Y. and Weber, R. (1988b). An Ontological Model of an Information System. IEEE Transactions of Software Engineering,16 (11):1282-1292, IEEE Press.

Wand, Y. and Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. Journal of Information Systems, 3:217–237.

Wand, Y. and Weber, R. (1995). On the deep structure of information systems. Information Systems Journal, 5:203–223.