Network of Excellence - Contract no.: IST-508 011

www.interop-noe.org

Deliverable DEM 1

UEML2.1

<table>
<thead>
<tr>
<th>Classification:</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Responsible:</td>
<td>UoT, UiB, UpV</td>
</tr>
<tr>
<td>Editors:</td>
<td>Giuseppe Berio (UoT)</td>
</tr>
<tr>
<td>Main Authors:</td>
<td>Giuseppe Berio (UoT), Andreas Opdahl (UiB), Victor Anaya (UpV) and Michele Dassisti (POLIBA)</td>
</tr>
<tr>
<td>Internal review:</td>
<td>Hervé Panetto (UHP)</td>
</tr>
<tr>
<td>Contributors to annexes:</td>
<td>University of Torino (UoT) IPK University of Bergen (UiB) Polytechnic University of Valencia (UpV) Politecnico di Bari (POLIBA) University of Namur (UoN) University of Stockholm (KTH) SINTEF University of Nantes (LINA &amp; IRCYNN) University of Nancy (UHP &amp; LORIA) University of Throndeim University of Bordeaux I</td>
</tr>
<tr>
<td>Tasks:</td>
<td>Task 2; Task 4</td>
</tr>
<tr>
<td>Status:</td>
<td>Final</td>
</tr>
<tr>
<td>Date:</td>
<td>30th November 2005</td>
</tr>
<tr>
<td>History</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>First version: Giuseppe Berio, <em>University of Torino (UoT)</em> on 15&lt;sup&gt;th&lt;/sup&gt; September 2005</td>
<td></td>
</tr>
<tr>
<td>Amendments: All Partners (<em>during the INTEROP project meeting held in Bologna</em>) on 4&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Addition of Section IV.1.1: Andreas Opdhal, <em>University of Bergen (UiB)</em> on 10&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Addition of Section IV.1.2 Victor Anaya, <em>Polytechnic University of Valencia (UpV)</em> 10&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Improvement and readability: Hervé Panetto, <em>University of Nancy (UHP &amp; LORIA)</em> on 14&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Improvement: Giuseppe Berio, <em>University of Torino (UoT)</em> on 16&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Reorganisation of Section IV.1.2 Victor Anaya, <em>Polytechnic University of Valencia (UpV)</em> 18&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Glossary and Section III.1.1: Andreas Opdhal, <em>University of Bergen (UiB)</em> on 18&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Section IV.1.1: Andreas Opdhal, <em>University of Bergen (UiB)</em> on 20&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Consistency verification: Giuseppe Berio, <em>University of Torino (UoT)</em> on 21&lt;sup&gt;st&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>INTEROP quality check: Kurt Kosanke, <em>CIMOSA Association (CIMOSA)</em> on 30&lt;sup&gt;th&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Clarification to the quality check reviewer: Giuseppe Berio, <em>University of Torino (UoT)</em> on 31&lt;sup&gt;st&lt;/sup&gt; October 2005</td>
<td></td>
</tr>
<tr>
<td>Changes (readability, structures, terms and reorganisation of Sections) proposed by the quality check implemented in the document: Giuseppe Berio, <em>University of Torino (UoT)</em> on 3&lt;sup&gt;rd&lt;/sup&gt; November 2005</td>
<td></td>
</tr>
</tbody>
</table>
# Table of Content

Table of Content ................................................................................................. 3  
Executive Summary ............................................................................................. 5  
Part I General ...................................................................................................... 6  
I.1 The organisation of the work ........................................................................ 6  
I.2 Main results .................................................................................................. 6  
Part II Overview .................................................................................................. 8  
II.1 Introduction .................................................................................................. 8  
II.2 Vision, application scenarios and challenges ............................................ 10  
II.2.1 Vision ..................................................................................................... 10  
II.2.2 Application scenarios ............................................................................ 12  
II.2.3 Challenges ............................................................................................. 13  
II.2.4 Summary ................................................................................................. 14  
II.3 Research objectives towards the vision, recognising the challenges .......... 15  
II.3.1 Addressing the research objectives ...................................................... 15  
II.3.2 Summary ................................................................................................. 16  
Part III Results ..................................................................................................... 20  
III.1 The UEML template approach ................................................................. 20  
III.1.1 The principles underlying the UEML template approach ................. 20  
III.1.2 The meta-meta model .......................................................................... 21  
III.1.3 Instantiating the meta-meta model ....................................................... 23  
III.1.4 The UEML template approach: an open-ended approach ................. 23  
III.1.5 The UEML template approach and the application scenarios .......... 24  
III.1.5.1 Model integration and transformation scenarios .......................... 24  
III.1.5.2 UEML core language scenario ....................................................... 25  
III.1.5.3 New methods and methodologies scenarios .................................. 26  
III.2 Users’ needs and language quality ........................................................... 27  
III.2.1 Requirements and their organisation ............................................... 27  
III.2.1.1 Requirements elicitation and collection ......................................... 27  
III.2.1.2 User needs organisation ................................................................. 29
III.2.2 Language quality ................................................................. 30
III.2.3 Summary ............................................................................. 34
III.3 Experimental application ....................................................... 35

Part IV Future Plans and conclusions ........................................... 37
IV.1 Future plans .............................................................................. 37
IV.1.1 The UEML template approach ............................................... 37
IV.1.2 The quality of languages and its use ...................................... 40
IV.1.3 Correspondences between languages .................................... 42
IV.2 Conclusions ............................................................................. 43

List of annexes ............................................................................... 44
Technical glossary ......................................................................... 45
References ..................................................................................... 47
Executive Summary

The document consolidates and summarises the most important results on UEML (Unified Enterprise Modelling Language) achieved in the first 18 months of the INTEROP project and provides an improved presentation of the results. Included are: an advance on UEML 2.0 (this is why the document is titled UEML 2.1) and a synthesis of the vision and challenges concerning UEML.

The central part of the document provides a detailed analysis of and discussions on the three main topics around which the work on UEML is developing. 1) the topic of “research approaches”, focusing on the fundamental research and analysing “how to model enterprise modelling languages”, 2) the topic of “requirements for enterprise modelling”, focusing on problems (including application domains) and solutions in the context of enterprise modelling for interoperability and integration, and 3) the topic of “enterprise modelling languages”, focusing on existing languages that might be used for enterprise modelling or which are related to enterprise modelling, and on how to select/classify (i.e. organise) existing languages, taking into account the requirements mentioned before.

Finally, the document discusses the refined research objectives derived from the results of the performed work. These research objectives will be useful for guiding current activities within and also beyond the INTEROP project.
Part I General

I.1 The organisation of the work

The work has been based in the following steps:

- Analysis of the Deliverable D 5.1 issued on month 18\textsuperscript{th} (*);
- Identification of the relevant material;
- Development of the additional sections concerning the vision, the challenges and the research objectives (already identified and their refinement);
- Presentation of the content of the additional sections an amendments during the meeting held in Bologna on 4\textsuperscript{th} October 2005;
- The first draft of the part II has been submitted to work package internal review;
- The consolidated draft has been submitted to the INTEROP quality check.

(*) Some important reports included in the Deliverable 5.1 have not been included in this deliverable because they are publicly accessible or used in other relevant initiatives of INTEROP. Their impacts on UEML 2.1 is anyway synthesised and referred in the part II text of this deliverable. These are:

- A report about the compliance between UEML 2.0 and UEML 1.0, including strengths and benefits of both (published);
- The state of the art about mapping languages (used in task group 3);
- Various experiment dealing with the usage of requirements form directly building a UEML core language (published).

I.2 Main results

One objective of the performed work has been to clarify and to make explicit the vision underlying UEML 2.0 resulting in the following statement:

UEML brings together theories and mechanisms for “modelling enterprise modelling languages”, theories and mechanisms for characterising and finding correspondences between constructs in distinct enterprise modelling languages and, finally, strategies for selecting/classifying such languages according to users’ needs.

This vision is consistent with the main application scenarios of UEML which also represent the real benefits: to support integration of enterprise models and integrated usage of enterprise models expressed in distinct enterprise modelling languages and in, long term, to support the identification of core constructs (i.e. fundamental constructs) for enterprise modelling and to develop new tools, methods and methodologies applying enterprise modelling with other kind of models (i.e. making available built-in mechanism for integration of enterprise models and integrated usage of enterprise models).

A new set of research objectives has been reported in the final section of this deliverable, that has been defined by analysing the performed work taking into account the vision. These research objectives are used to guide the current activities inside INTEROP, but they are relevant for the future work beyond INTEROP as well.
This deliverable also provides a reorganisation and presentation of the main results (according to the vision) about UEML 2.0 already documented in the Deliverable 5.1 issued on month 18th. These results are:

- The UEML principles and the UEML template approach, that is a very general approach for describing enterprise modelling languages (or for “modelling enterprise modelling languages”, i.e., meta-modelling).
- The definition of a (new) requirements collection and elicitation method for driving the selection of languages that should be considered relevant to UEML. Requirements are derived from user needs.
- An extension of an existing quality framework for modelling language evaluation that can be used to make sound and transparent any selection of languages for UEML. Specifically, it comprises a first set of quality criteria and their relationships with collected requirements. These relationships are extremely important to ground UEML on user needs.
- An experimental application of the UEML template approach (for initial validation) and the selection of a set of enterprise modelling languages based on a subset of the defined quality criteria. The results of this experimental application have been documented and are also part of the main results concerning UEML 2.0.
Part II Overview

II.1 Introduction

Information is becoming and will be in the future represented under several distinct forms. Indeed, following the history, information has been represented in running systems as programming language files, data formats and so on. EAI (Enterprise Application Integration) has been used and is being used to put together systems storing data and programs in several forms. The aim of EAI is indeed to reuse existing running systems by integrating their applications and data.

Nowadays, the clear tendency is to move from low levels of represented data and applications (in the running systems) to more conceptual artefacts, like models. For instance, in the Model Driven Architecture\(^1\), models are the central concepts and they become executable and have to be managed similar to any other software artefact.

Enterprise models are, by definition, kinds of models and represent information\(^2\) about the enterprises (both static/dynamic and descriptive/prescriptive), in several forms due to the variety of applications to be supported. Therefore, enterprise models are to be used either in a stand alone mode or in relations with other models (possibly closed to running systems, comprising applications and data). The latter requires to support model integration and integrated usage of models (here the terms are not standardised but they cover all manipulations and operations on models, such as transformation, translation, integration, composition, updating, coordinated use of models and so on).

Therefore, it is important to focus on mechanisms for establishing relationships between models.

The starting point for establishing relationships between models can be based on relationships between languages. However, due to the broad applications of enterprise modelling and modelling in general, the required modelling capabilities are continuously evolving and the languages used for expressing these models are very distinct in their nature and features.

The work on UEML is aimed on characterising these relationships (in the remainder named correspondences) between enterprise modelling languages. However, finding correspondences between languages may also lead to discover basic modelling concepts or constructs, which can be used for activities like standardisation and the development of new methods, methodologies and tools (based on several kinds of models as well). These aspects of the work on UEML (Unified Enterprise Modelling Language) are really facet of the same problem i.e. enabling model integration and integrated use of models. The former should enable to support various the activities requiring model integration or integrated usage of models, the second should enable to reduce, in the long term, the complexity of these activities.

In the first release of UEML, i.e. UEML 1.0, outcome of the previous UEML project (www.ueml.org) (Panetto et al. 2004), the main results were related to the development of a “pivotal language” (sometimes referred to as an exchange format) that can be used to perform simple exchanges between tools supporting enterprise modelling languages.

The current version of UEML, named UEML 2.1, takes a more general approach (INTEROP Deliverable 5.1, Section II.5), (Berio, 2005) by providing a precise way “to describe (or model) enterprise modelling languages”, named the UEML template approach. This approach requires a detailed (ontological) analysis of the constructs found in enterprise modelling languages and allows

---


\(^2\) The term “information” is used here to indicate data and behaviour (i.e. information about data and behaviour).
to formally define **correspondences** between constructs in distinct languages and thereby a UEML-based core enterprise modelling language (termed **UEML core language** in the remainder).

The UEML template approach is also complemented by a continuous collection and elicitation of requirements (performed with various categories of users\(^3\)). Based on these requirements, clear strategies are defined for selection/classification of existing languages as relevant for UEML. Such selection and classification of existing enterprise modelling languages is clearly a key pre-requisite for the development of a useful and used UEML. While selection/classification should be the primary aim of requirements, they can also be used to develop new languages (or just new constructs) and further used in UEML.

Therefore, what is currently called UEML, mainly due to historical reasons, brings together theories and mechanisms for describing enterprise modelling languages, (embryonic) theories and mechanisms for characterising and finding correspondences between constructs in distinct languages and, finally, user needs driven strategies for selecting/classifying such languages.

All these theories and mechanisms provide basic mechanisms (i.e. the correspondences between constructs in distinct modelling languages) to enable integration and integrated use (in several ways, e.g. integration, transformation, of models used in enterprise activities) of enterprise models (stored in distinct enterprise modelling tools supporting distinct languages) and, on the other hand, when applied to selected languages, to discover core concepts for enterprise modelling (represented throughout a UEML core language).

This report presents consolidated results of the work on UEML that has been performed during the first 18 months within the **INTEROP project**. The different sections and annexes present, according to vision, objectives and application scenarios, insights on the various aspects that have been discussed and the solutions that have been provided or are envisioned.

Section II.2 contains a detailed description about the vision, the application scenarios and the challenges that should be addressed. Section II.3 presents the objectives that have been defined in the first phase of INTEROP and how they have been addressed. The same section also explains the relationships between the challenges and the objectives. Section III.1 is fully devoted to the UEML template approach. The section also provides an overview on how the various application scenarios advocated in section II.2 can be addressed by the UEML template approach. Section III.2 provides insights about the work performed on the selection/classification of modelling languages. Section III.3 discusses the experimental application of the UEML template approach and the results about the first selection of modelling languages. Based on this experimental application, section IV.1 presents the research objectives that should be addressed in the second INTEROP phase (already started\(^4\)). Some of them, because very challenging, will be eventually beyond the INTEROP project. However, the current and future work, within and beyond INTEROP, should be oriented in the directions these objectives underlie. Finally, Section IV.2 concludes this deliverable by summarising the main points and provides some additional final comments.

---

\(^3\) User is intended in broadest sense; a better and neutral term could be Actor.

\(^4\) From April 2005 to November 2006.
II.2 Vision, application scenarios and challenges

II.2.1 Vision

According to the Introduction, UEML aims to support integration and integrated use of enterprise models expressed in distinct enterprise modelling languages. The following statement summarises the UEML vision:

UEML brings together theories and mechanisms for “modelling enterprise modelling languages”, theories and mechanisms for characterising and finding correspondences between constructs in distinct enterprise modelling languages and, finally, strategies for selecting/classifying such languages.

There are several reasons for thinking in this direction and the most important are:

- Problem throughout enterprise models often requires the combined use of specific models expressed in different languages (existing and future), which are employed because they provide the best solution for a particular part of the problem;
- Defining a new unifying or universal language does not take into account that models are originally expressed in distinct languages (in fact, model integration and integrated use of models are also relevant problems when all models are expressed in the same language); additionally, it is risky, especially if it is directly built on a set of requirements (even if they are issued by experts), because the value and degree of precision of these requirements may make the resulting language huge, unstable and poorly structured and requires anyway an important effort for using that set of requirements (INTEROP-Deliverable 5.1, 2004, Sections II.2.2.2, II.2.7), (Berio & Riquet, 2004), (Bergoltz et al., 2005), (Billington et al., 2003), (Heymans & Petit, 2004) report specific studies within and outside INTEROP; though, it is better to recognise the existence of several languages and to focus on establishing and managing correspondences between them that can further be used for integrating models or to support the integrated use of these models;
- Selecting and classifying languages based on their strengths and weaknesses and finding correspondences between these selected languages might lead to the definition of a set of enterprise modelling constructs that can be qualified as core constructs employed in most of the “best” languages; the advantage of this approach is that any core construct is not introduced by following an informal analysis/thinking about enterprise modelling and enterprise modelling languages. Instead, the core construct is derived from a sound base and a deep analysis of languages and because of correspondences, the evidence of the core construct is implicitly assessed;
- Finally, standardisation initiatives (e.g. OMG; ISO/DIS 19440) try to achieve model integration and integrated use of models by making everyone use the same language (or set of standardised (sub)languages). Although this goal will remain unrealistic in the foreseeable future, and standardisation attempts do not emphasise the correspondences between their own standard and other proposals. They offer little support for model integration and integrated use of models across standards. The work on UEML should also attempt to support the latter kind of integration. It could serve as an input to current standards initiatives, thus influencing them, or it could be a support for new standardisation initiatives.
Following these arguments, UEML should not define, a new language for modelling enterprises, but should focus on characterising and finding correspondences (i.e. mappings, relationships) between distinct classified and selected enterprise modelling languages. Moreover, based on these correspondences, the UEML initiatives would also focus on characterising core concepts/constructs for enterprise modelling and notations for these concepts (i.e. what has been called a “UEML core language” in the Introduction).

The interest of correspondences between distinct languages (i.e. *language level correspondences*) is among the most important aspects in treating models (i.e. in integration and integrated use of models). These correspondences should be **model-independent** i.e. they should be valid independently of models. However, integration and integrated use of enterprise models require additional information (e.g. information closely related to models, possibly leading to “model-dependent correspondences” i.e. correspondences at the model level) that are not related to languages and their correspondences (Berio, Mertins and Jaekel, 2005).

The example depicted in Figure II.2.1 below illustrates the idea, in a typical *model integration scenario* (i.e. in a scenario in which at least two models coexist), of what correspondences between languages can be. The example also includes “model-dependent correspondences” i.e. correspondences focusing on models and model-elements).

![Figure II.2.1: Example of model-independent correspondences.](image)

The figure depicts a correspondence linking a construct called *Activity* with a construct called *Action* in an “equivalence relationship”. In two distinct models, these two distinct constructs are used to represent two artefacts named respectively *My Production* and *The production*. Based on the correspondence, it is possible to state that *My production may be The production*; however, to conclude that *My production is The production* additional information is required (for instance, the two model elements refer to the same physical products, otherwise they only refer to, may be, a similar production process but not to the same production processes; other examples belongs to typical scenarios in database and schema integration where two distinct relational tables or two distinct entities showing the same set of attribute names do not need to include the same data) Therefore, despite the “equivalence” between Activity and Action, it may happen that *My production is not The production*.

In other words, correspondences among languages, in an integration scenario, lead to possible relationships between models represented in these languages. Additional information are required to find (or to state) the actual relationships between the given models. Nevertheless, establishing correspondences between modelling constructs at the language level is a powerful aid for model integration and integrated use of models: both activities become easier to perform at the model level, when they are aided by clear correspondences at the language level. The language-level correspondences provide which model-level elements that have to be checked for correspondence.
and which ones do not have to be considered. Similar examples of using correspondences can be drawn for model translation and, more generally, model transformation.

Usually, the situation is more complex than in Figure II.2.1, because the correspondences between constructs in different languages are seldom exact equivalences. More often, constructs correspond because they overlap, or because they are redundant, to a certain extent. In such cases, language level correspondences remain a powerful aid at the model level as described above. But another important aid also becomes visible: the language-level correspondences also show us small potential overlaps between model elements, which might introduce inconsistencies in the overall model and which might otherwise not have been discovered.

If several other constructs in other languages refer to the same concept as it is or the Activity and Action in Figure II.2.1 above, this concept can be qualified as a “core concept” for the modelling of enterprises. A notation for the core concept can be introduced as part of deriving a “core language” from UEML (named UEML core language in the Introduction). The advantage of this approach is that the core concepts and constructs are not introduced by following an informal analysis/thinking about enterprise modelling and enterprise modelling languages. Instead, a UEML core language will be derived from a sound base and a deep analysis of languages. The identified correspondences provide the evidence of the existence of core modelling concepts (and if correspondences are also used for working on models, this also provide the evident of the usefulness of the core concepts).

The vision presented in this section has been elaborated by analysing the performed work (INTEROP Deliverable 5.1) and the previous UEML Thematic Network project results (UEML Deliverable 2.1), (UEML-Deliverable 2.3), (UEML-Deliverable 3.1).

II.2.2 Application scenarios

The vision presented in the previous section can be tested by application scenarios (all the aim of integration and integrated use of models). These scenarios are abstract scenarios and concern models, but in various ways. They can be described as follows (INTEROP Deliverable 5.1, 2005, Section II.2.3):

- **Transformation and integration of enterprise models** represented in different (enterprise) modelling languages, which constitute the base for exchanging meaningful information (represented as models) between distinct (enterprise) modelling tools;
- **Required global consistency** between various enterprise models (i.e. the relationships between enterprise models should be fully represented) especially if these enterprise models are represented in distinct enterprise modelling languages.

Other application scenarios are in methodologies and foundations for enterprise modelling. Specifically:

- To define **new methods and methodologies** for enterprise modelling based on coupling several languages (here the emphasis is on **building integrated models**);
- To provide new methods, methodologies and tools based on a well-designed “**UEML core language**” (here the emphasis is on reducing the risk of modelling similar or overlapping situations by using very distinct constructs thereby **reducing the complexity of integrating and transforming models**).
The activity on UEML should not really go deeper in understanding how to put in place the various application scenarios. However, in the first phase of INTEROP a very extensive state of the art about transformation and integration of models has been performed, which will be used in further evaluations. Other applications scenarios are analysed in specific INTEROP activities.

The application scenarios presented in this section have been elaborated by analysing the performed work and the previous UEML Thematic Network project results (UEML-Deliverable 2.3, 2003).

II.2.3 Challenges

Underlying the vision and the application scenarios, there are various challenges that need to be addressed. In the remainder of this section, the identified challenges are presented. These challenges are used to explain the research objectives defined in the first phase of INTEROP and how these research objectives have been addressed.

C1) The very big challenge of UEML is about the usage of enterprise modelling languages that are characterised by very different features and foundations.

There is the need to focus on languages (opposite to focus on models): specifically, on what elements can be distinguished inside every language. These elements are (based on Harel & Rumpe, 2004, but using other terms):

- **syntax** (which includes both the internal structure of the language, its *abstract syntax*, to its visual presentation, i.e., the *concrete syntax*);
- **ontology** (which represents the things in the *domain* that the syntax should be able to represent);
- **representation mapping** (the relationship between the abstract syntax and the ontology).

All languages are necessarily associated with these three elements; however, how these elements are provided is very different. For instance, many enterprise modelling languages are equipped with a syntax provided by a meta model while the representation mapping onto the domain is expressed by natural language (for instance, English). The same situation is also very common in software engineering. Generally speaking, languages tend to describe, in a more or less formalised way, the syntax whenever they are used for interacting with humans; languages tend to describe, in a formalised way, the domain and the representation mapping, whenever there is the need to mathematically evaluate some properties of the models.

All categories of languages are important in modelling enterprises. Indeed, in the first case, languages are more or less similar to conceptual schemas that are required to approach, empirically, complex enterprise problems (i.e. trying to solve them by some methods). In this case, languages are the representation of empirical thinking about enterprises and they are associated to specific methods and methodologies. In the second case, a language is a way to verify correctness of models according to some desired properties.

C2) The structure and the objects of the common ontology shared among languages.

---

5 New methods and methodologies are defined for achieving specific goals and additionally supported by *method-engineering environments* TG6 (INTEROP NWP, 2005), *model-driven interoperability* TG2 (INTEROP NWP, 2005); a UEML core language can be also targeted by standardisation activities (especially in the ISO TC184/SC5 committee).

6 Harel & Rumpe (2004) use the terms “semantic domain” and “semantics” where here it is used the terms “ontology” and “representation mapping”, respectively. See the Technical Glossary for explanations.
According to challenge C1, there is the need to organise and to establish a common ontology for languages of distinct features and nature. Eventually, for correspondences there is the need to compare, at some extent, languages and constructs: an explicit common ontology is needed. This common ontology should represent the basic phenomena of the real world (i.e. enterprises) that are represented throughout languages. Basic phenomena can indeed be composed (in some way) for representing more complex phenomena than languages and their constructs actually allow representing (by using a notation (syntax) i.e. a finite representation for a domain). To be general, the approach to be undertaken might be closely related to approaches in other areas of computer science. For instance, in computational models, languages (notations) are used for representing the concepts of computations that are, for instance, expressed by infinite sequences of states. In the same way, in UEML, there is the need to refer to a set of basic phenomena in concrete domains (i.e. enterprises). Accordingly, enterprise modelling languages become notations for representing these phenomena and their combination.

C3) How to define and to identify the interesting correspondences between represented languages?

In other words, how to characterise (in term of properties) and derive correspondences according to the common ontology? Once time more, the distinct nature of languages raises important points. The most important is: to what extent UEML is able to support the model integration and transformation i.e. which is the limit between the required information about models and the required information about languages? Or even, which kind of correspondences UEML should allow (or just allows) to establish between languages and their constructs?

C4) Acceptance

The proposed vision has been developed inside INTEROP, but now a large external acceptance is required. The vision proposes both correspondences between language concepts/constructs (these are the base for maintain coexisting of several languages) and (with less priority) the synthesis of core concepts/constructs for enterprise modelling. Public acceptance is certainly easier to be obtained for the former, whereas for the latter this is a long term solution and can only be realised through tool vendors.

II.2.4 Summary

Table II.2.1 below summarises the vision and the corresponding scenarios of application. The time scale makes evident that the primary objective should be the definition of correspondences between languages. These correspondences can further be used to effectively define core concepts of enterprise modelling and notation for them (already termed “UEML core language”).

<table>
<thead>
<tr>
<th>Vision</th>
<th>Scenarios of application</th>
<th>Challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthesis of a UEML core language; identification of core concepts of enterprise modelling domain;</td>
<td>To support the definition of new methods, methodologies and tools based on the UEML core language</td>
<td>C4</td>
</tr>
<tr>
<td>Correspondences between languages based on the common ontology: definition, representation and analysis</td>
<td>To support models integration, transformation and translation; to support the definition of new methods and methodologies coupling several languages</td>
<td>C4</td>
</tr>
</tbody>
</table>
Working towards the vision, at least four challenges have to be carefully addressed. The first (C1) is about the distinct nature of enterprise modelling languages to be taken into account. The second (C2) is about the definition of a common semantic domain for these languages. Challenges C1 and C2 are pre-requisites for addressing the definition of correspondences between languages (C3). Finally, acceptance (C4) is the most important challenge, because any proposal towards the vision will contain some elements which are not scientific but just social. These elements are mainly the languages that should be used in the UEML development and the synthesis of the common semantic domain. However, challenge C4 is also addressed by the research objectives guided by the vision itself.

II.3 Research objectives towards the vision, recognising the challenges

This section provides and comments the research objectives defined in the first phase of INTEROP (1\textsuperscript{st} November 2003 - 30\textsuperscript{th} April 2005) according to the challenges and the application scenarios that have been described in Sections II.2.2, II.2.3 above. Some of these research objectives have already been addressed and important work has been performed; because of the dependency between objectives, other research objectives have not yet been acted upon.

O1) Definition of the approach for representing modelling languages according to their three main components: presentation, representation mapping and the common ontology.

This objective is directly related to challenges C1 and C2. Achieving this objective is the prerequisite for meeting the others. Indeed, the key point is how to approach the precise “representation of enterprise modelling languages”. Once the objective has been achieved, a common ontology can be derived from representing selected languages (see <Objective O3> below)

O2) The foundational nature of the proposed approach requires understanding to what extent the approach is suitable for representing enterprise modelling languages and to what extent it can be used.

This objective is directly related to challenges C1 and C2 and C4. Indeed, it is important to evaluate whether and how the proposed approach is working on a broad variety of languages. Working toward this objective is also useful improving the proposed approach and for increasing its suitability and acceptance (challenge C4).

O3) To define a sound and transparent way for selecting languages to incorporate into UEML (and to select languages accordingly).

This objective is mainly related to challenge C4. Even under a restriction of the scope, it is impossible to consider all existing and potentially relevant languages. Selected languages should be used for defining a common ontology and perhaps a core language for enterprise modelling. It is therefore important to fully trace and make transparent how languages are selected. Additionally, selection of languages might become effective if it is justified by (real) user needs. These user needs provide a clear understanding of what is required by several categories of users. Finally, depending on the used languages, it is possible to reuse the common ontology for modelling other languages.
O4) To characterise correspondences between languages and their constructs in terms of a common ontology.

This objective is mainly related to challenge C3. To characterise correspondences between languages and their constructs, definitions and properties of these correspondences should be provided. Definitions and properties should refer to the common ontology for all languages, according to the <Objective O1> stated above. In further work, there will be the need to evaluate the kind of correspondences that can be effectively found in the common ontology.

II.3.1 Addressing the research objectives

O1) Definition of the approach for modelling the modelling languages according to their three main components: syntax, domain and representation mapping.

The so called “UEML template approach” has been retained (details are fully provided in Section III.1 below). This approach is based on:

- Adaptation and adoption of a meta-meta model (represented as a simple UML Class Diagram for describing: the abstract syntax of individual modelling constructs; an ontology that is common to all the languages (in the remainder, termed the common ontology); and representation mappings between them);
- Standardisation of objects that can be used to instantiate the meta-meta model (specifically, the BWW concepts (Wand & Weber, 1988), (Wand & Weber, 1993), (Wand & Weber, 1995) have been used for this purpose (Section III.1.3); these objects represent actually basic phenomena of the real world and are used for describing complex constructs found in the various languages;
- A template, named UEMl template, that is used to instantiate the meta-meta model (this is needed because there are currently no specific computerised tools to instantiate the meta-meta model; however, the experience gained by using the template can be also very useful for developing user interfaces in a computerised tool supporting the approach).

O2) The foundational nature of the proposed approach requires to understand to what extent the approach is suitable for representing enterprise modelling languages and to what extent it can be used by the involved participants.

An experimental application of the proposed approach has been performed (Section III.3). Languages that have been used are (additional information of each of these languages can be found in Annex A5):

- UML 2.0 (Class Diagram and Activity Diagram) is a typical software modelling language but it also covers software requirement specification and it is often used for modelling non-software artefacts.
- ISO/DIS19440 is the evolution of the previous standard ENV 12204. It is closely related to enterprise modelling and its aim is to provide a set of constructs for modelling enterprises especially in the manufacturing domain.
- IDEF 3 is the “business process specification language” within the IDEF suite of modelling methods and languages. IDEF3, as the whole IDEF suite, is closely related to enterprise modelling.
• BPMN is the Business Process Modelling Notation associated to the Business Process Modelling Language (BPML) specification. It provides an abstract formal model and XML syntax for expressing executable processes that addresses complex activities, transactions and their compensation, data management, concurrency, exception handling and operational semantics. BPML itself does not define any application semantics such as particular processes or application of processes in a specific domain; rather it defines an abstract model and grammar for expressing generic processes. This allows BPML to be used for a variety of purposes that include, but are not limited to, the definition of enterprise business processes, the definition of complex Web services, and the definition of multi-party collaborations.

• Coloured Petri Nets are a well-known modelling language for expressing the behavioural aspects of complex systems. They are based on the historical model introduced by Carl Adam Petri. Coloured Petri Nets have been applied and are still applied to represent complex systems, in the various application contexts: computer systems, network protocols, manufacturing systems, knowledge base dynamics, and so on.

• GRL is a language for modelling goals and non-functional requirements. It is mainly originated by the requirement engineering community allowing the formal refinement of goals.

• XPDL is the XML Process Definition Language defined by the Workflow Management Coalition (WfMC).

Additional information about each of these languages can be found in the filled-in “language templates” (Annex A5) and in the various states of the art developed in other previous or running projects (UEML SoTA, 2002), (ATHENA SoTA, 2004).

The selection of these languages has been explained by using the way of working shortly described under the <Objective O3> paragraph below and fully described in Section III.2.2.

O3) To define a suitable way for selecting languages relevant to UEML.

According to what has been said about challenge C1, the category of languages that can be considered under the term enterprise modelling is very broad and often subjective (this also becomes clear by reading the most recent states of the art developed in other previous or running projects (UEML SoTA, 2002) (ATHENA SoTA, 2004), and also from an historical perspective looking at various surveys in enterprise modelling languages (Vernadat, 1997), (Vernadat, 2000), (Bernus & Fox, 2005).

The internal discussion raised the point of language quality. The concept of quality has been already addressed in the area of information systems where modelling is a fundamental activity. By analogy, the same concept of quality can be applied to enterprise modelling languages. Accordingly, quality is closely related to the suitability of the language for modelling the enterprises and, through models, to provide answers to specific questions or problems. Additionally, initiatives aiming to develop a UEML core language, should probably focus on analysis of languages that are successfully employed in enterprise modelling enabling the creation of models that solve problems of enterprises.

The proposed way of working towards the objective relates a set of enterprise modelling user needs (that represent problems and existing solutions concerning enterprises) to a set of quality criteria that has to be developed. The rationale for having both users’ needs and quality criteria is because
user needs are broad, not very precise, often abstract: this makes difficult to use them for direct evaluation of modelling languages (INTEROP Deliverable 5.1, 2005, Section II.2.7). Therefore, quality criteria are (should be) the precise and operational counterpart of users’ needs.

This way of working is in line with the proposed vision. Moreover, the quality of a language can also be evaluated in specific contexts (i.e. on enterprise, a group of enterprises, others groups and communities), including languages for solving specific problems expressed with specific users’ needs (that may lead to a “local UEML”). However, these specific languages may also contribute to core concepts for enterprise modelling, (the quality of languages is only a decisional tool for making choices whenever the potentially interesting languages are numerous, as for the development of core concepts in enterprise modelling).

Finally, there are four important corollaries of the proposed way of working for selecting enterprise modelling languages as relevant for UEML:

- The first corollary is that the common ontology that will be built by using the “high quality languages” i.e., generally speaking, advanced and influencing enterprise modelling languages; therefore, this way of working should contribute to speed-up UEML development and use because most of the common ontology becomes reusable;
- The second corollary is that the way of working provides how UEML should evolve over time; indeed, the proposed way of working can be used to continue the disciplined activity on UEML (toward, for instance, the definition of core concepts/constructs for enterprise modelling);
- The third corollary is that the way of working makes UEML directly related to the users’ needs, one of the most important preconditions for making UEML usable and used and to prepare it for large acceptance;
- The fourth corollary is that languages can be added independently of this way of working, leading to a kind of “local UEML”; additionally, users’ needs not filled by currently existing languages should be realised in new languages; then, these new languages will be evaluated according to their qualities, possibly selected, or, as said before, just added to any “local UEML”.

Details about the quality of languages and requirements are presented in Section III.2 below.

O4) To characterise correspondences between languages and their constructs in the common ontology.

An embryonic study has been performed in (Berio, 2005). The reason of this limited work is the need to formalise the meta-meta model currently represented as a UML Class Diagram (Section III.1.2 below).

The performed study suggests the need to distinguish between basic and complex correspondences. The basic correspondences focus on constructs in distinct languages used to model similar things (i.e. construct to construct correspondences) while complex correspondences should relate sets of constructs in distinct languages. The study is also complemented by an analysis of the state of the art about “mapping languages” (INTEROP Deliverable 5.1, 2005) bringing together the most relevant proposals referring (in some
way) to mappings: mappings over models and meta models in UML, mappings over ontologies and, finally, mappings over EXPRESS models. This analysis can be used as a starting point:

- To reuse the existing and advanced techniques for specifying correspondences (many of these techniques are really operational and based on UML);
- To continuously improve the understanding of the distinction and the relationship between correspondences between languages (model-independent correspondences) and correspondences that relate models (as described in the Introduction, a correspondence between two constructs may be used for transforming or for integrating models, but for understanding if this correspondence should (must) be applied, additional information is required).

II.3.2 Summary

Based on the developed vision, application scenarios and challenges, Table II.3.1 below summarises the relationships between the visions, the application scenarios, the challenges and the research objectives defined in the first phase of INTEROP. The time scale provides a clear understanding on the order, taking into account priorities, the objectives should be achieved.

*Table II.3.1: The research objectives addressing the challenges towards the vision and the application scenarios.*

<table>
<thead>
<tr>
<th>Vision</th>
<th>Scenarios of application</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthesis of a UEML core language; identification of core concepts of enterprise modelling domain;</td>
<td>Efficient mechanisms for modelling in the UEML core language (or for transforming heterogeneous models in the UEML core language);</td>
<td>C3</td>
</tr>
<tr>
<td>Corresondences between languages based on the common ontology: definition, representation and analysis</td>
<td>To support models integration, transformation and translation; to support to definition of new methods and methodologies</td>
<td>C4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identified research objectives</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>O4) To characterise correspondences between languages and their constructs in the common ontology.</td>
<td>C3</td>
</tr>
<tr>
<td>O3) To define a sound and transparent way for selecting languages relevant to UEML.</td>
<td>C4</td>
</tr>
<tr>
<td>O2) The foundational nature of the proposed approach requires to understand to what extent the approach is suitable for representing enterprise modelling languages.</td>
<td>C1, C2, C4</td>
</tr>
<tr>
<td>O1) Definition of the approach for modelling the modelling languages according to their three main components: syntax, common ontology and representation mapping.</td>
<td>C1, C2</td>
</tr>
</tbody>
</table>

---

7 Therefore, covering the three main areas of the whole INTEROP project, respectively Architectures, Ontologies and Enterprise Modelling.
Part III Results

III.1 The UEML template approach

The “UEML template approach” is the way proposed for describing modelling language constructs. The sections below present the principles, the related meta-meta model and the way the meta-meta model is instantiated. These three sections describe the base of the proposed approach. The two final sections provide insights about how the various application scenarios (Section II.2.2) might be realised by using the proposed approach.

III.1.1 The principles underlying the UEML template approach

The UEML principles have been taken into account to define the UEML template approach. These principles have their foundation in the vision.

- **Integrative approach**: UEML way for describing modelling languages should try to incorporate existing languages throughout the definition of an ontology (Section II.2.3) common to all those languages (and therefore called, in the remainder, the common ontology), based on deep analysis of each language construct; additionally, the approach should support the incorporation of specific (set of) modelling constructs across languages i.e. without the need of incorporating all constructs found in a language;

- **Separate presentation, representation and internal structure**: UEML should have clearly separate meta models (created to incorporate languages) for: internal structure (abstract syntax of constructs); presentation (lexical, syntax and related issues); and representation (common ontology and representation mapping). On the presentation side there should be one meta model for each included type of diagram (visual languages) or modelling construct description. On the representation side there should be a common meta model that integrates the domains of the different languages. Each presentation meta model should of course be clearly related to the representation-side common meta model. (Because the various meta models are thus interrelated, it is possible to refer to an “overall meta model”, which is composed of those several sub-meta models). A language definition such as the OMG's UML definition is overly complex in part because it tries to define concrete syntax, abstract syntax, domain ontology and representation mapping all in the same family of meta models, without clearly distinguishing between them.

- **Extendable, tailorable**: It should be possible to incorporate new modelling languages when needed and to tailor the overall meta-model locally to fit specific organisational standards. In other words, the approach should be open-ended.

- **Structured approach to organising and managing the overall meta model**: As more languages are incorporated, a major risk is that the overall meta model becomes messy. Providing principles for extending and evolving the meta model is therefore important. In particular, the approach should ensure reuse of the concepts/phenomena in the representation meta model (and reuse constitutes the base for eventually defining a “UEML core language”).
• **Standardised and template-based:** All diagram type descriptions and modelling construct descriptions should be in a common standardised format (template), which can include diagrams, meta models, text etc.

• **Industrial languages:** Within the limits imposed by the overriding “extended quality framework of modelling languages” (Section III.2.2 below), priority should be given to industrial languages to make sure the performed work is practically relevant. However, while probably not a priority, selection of experimental languages might also be incorporated.

### III.1.2 The meta-meta model

The “UEML template approach” is based on a **meta-meta model** (which is the way to organise and manage the overall meta-model) developed by (Opdahl & Henderson-Sellers, 2004). This meta-meta model is presented as a UML Class Diagram in Figure III.1.1 below: It comprises the set of classes that are used for modelling a modelling language. The meta-meta model corresponds most directly the “Representation” section (in the earlier versions called the “Semantics” section) of the “UEML template” (details about the “UEML template” are in Section III.1.2 and Annex A1). The “Preamble” and “Presentation” (in the earlier versions called “Syntax”) sections of the “UEML template” are not yet covered because the proposed meta-meta model so far only covers abstract syntax, not concrete syntax.

The main features of the meta-meta model are:

- It is based on a limited set of well-defined concepts (defined in Bunge's ontology and the BWW-model):
  - **Classes** of things in the domain. Classes are characterised by properties and they are organised in generalisation/specialisation hierarchies. A modelling construct may represent any number of such classes.
  - **Properties** of things and classes in the domain. Properties belong to things and characterise (or define) classes. Properties are organised in precedence hierarchies (being alive precedes being a mammal; being a mammal precedes being human). A modelling construct may represent any number of such properties. Properties can be mutual to two or more things or classes (i.e., “relationships”) There is also a special part-whole relation property.
  - **States** of things in the domain. States are defined in terms of properties and may be constrained by a particular kind of property, called a state law property.
  - **Events** of things in the domain. Events have a from state and a to state and may be enforced by a particular kind of property, called a transformation law property. Events may comprise other events. Such complex events, containing intermediate states, are also called processes.
- It represents the abstract syntax of a modelling language construct, and maps the abstract syntax onto a common ontology of concepts that are shared between languages and constructs.
- It is represented in UML but this constitutes neither a constraint nor a limit; UML is actually a way to use a notation that is familiar with most of the working-in INTEROP Partners. Indeed, the meta-meta model has been realised in a working Protegé prototype tool.
- While applied to enterprise modelling languages, the meta-meta model allows describing generically any language of a concrete domain through its constructs; therefore, the
The proposed approach is open-ended because it allows reusing classes, properties, states and events.

**The “abstract syntax” of each individual modelling construct**

**“Representation mapping” from abstract syntax to the common ontology, in a sense capturing “semantics”**

Figure III.1.1 above shows as a UML Class Diagram the structure of the meta-meta model, which is organised in two layers; an upper construct layer (mainly related to the abstract syntax of the construct) and a lower ontology layer (i.e. the common ontology). These two layers are related by associations that indicate what phenomena in the domain each construct is intended to represent. Each construct has an individual abstract syntax layer, whose contents are mapped onto the common ontology layer shared by all construct definitions. In consequence, each modelling construct can be defined in detail in the private abstract syntax layer, whereas the common ontology accounts for representational overlaps (or redundancies) between different constructs. In the abstract syntax layer, a ConstructDefinition has a constructName and an instLevel, thus accounting for the first entry in the “Representation” part of the “UEML template”. A ConstructDefinition also consists of one or more RepresentedClasses, zero or many RepresentedProperties and a single RepresentedSegment, thereby accounting for the other three template entries too. Because some modelling constructs may be used to represent the same class and property several times, RepresentedClasses and Properties also have roleNames and minimum and maximum cardinalities. If a RepresentedSegment is a process, state or event, the segment must contain RepresentedStates and RepresentedEvents to specify its behaviour with further detail (for instance, a construct like Process can be represented as networks of states and events).
In the common ontology layer (i.e. the common ontology), there are (ontological) Classes, Properties, States and Events that correspond to all the RepresentedProperties, RepresentedStates and RepresentedEvents from the individual abstract syntax layers of all the modelling constructs defined so far using the template. A Class represents a collection of things that a modelling construct is intended to represent; a Property represents the properties of those things that a modelling construct is intended to represent; an Event and a State represent the changes in those properties and things that a modelling construct is intended to represent. In the common ontology layer, no class, property, state or event can be duplicated, so whenever two modelling constructs are representationally (or referentially) overlapping/redundant, the exact point(s) of overlap can be found in the common ontology. When a new modelling construct is defined using the template, its private RepresentedClasses, RepresentedProperties, RepresentedStates and RepresentedEvents must refer to Classes, Properties, States and Events in the common ontology. In some cases, the appropriate Classes, Properties, States and Events may already be defined in the ontology. In other cases, they must be inserted into the ontology before the new modelling construct can be fully defined.

A generalisation hierarchy between classes and precedence hierarchy between properties ensure that the ontology remains well structured and easily browsable (A property precedes another if no thing can possess the latter property without also possessing the former. For example, being a living thing precedes being a mammal, which in turn precedes being a human being. When a new Class, Property, State or Event is inserted into the ontology, the appropriate generalisation and precedence relationships to existing Classes, Properties, States and Events must of course also be defined).

### III.1.3 Instantiating the meta-meta model

The instances (i.e. the objects) of the meta-meta model are created by filling in the “UEML template” (Annex A1). This UEML template is then a simple way to instantiate the meta-meta model. Some of the entries in the template correspond to the abstract syntax layer, whereas others correspond to the common ontology. The latter entries are filled in by reusing concepts from the common ontology when possible. Furthermore, the template goes beyond the meta-meta model, because it also supports presentational (syntactical) description of modelling constructs. The UEML template has been then used in the experimental application (Section III.3).

### III.1.4 The UEML template approach: an open-ended approach

According to what has been said in Sections II.2.1 and II.3, the initiative with its focus on core concepts for enterprise modelling (including the development of a UEML core language) is useful to provide reusable definitions in the common semantic domain make it easier to describe additional modelling languages to be used as base for defining new languages.

However, the UEML template approach is a general way to model languages, their constructs, with their concrete meaning based on a common ontology. Therefore, the UEML template approach can be used for describing other languages that are not necessarily used for generating part of the UEML core language. Indeed, the provided approach allows

- To reuse the common ontology that has been defined by analysing several languages to develop a UEML core language; because classes and properties defined in the common ontology can be reused through the specialisation/generalisation hierarchy and the precedence network.
To extend in a controlled way the UEML core language, for the same reasons explained in the previous point.

The situation is depicted in Figure III.1.2 below.

![Figure III.1.2](image)

**Figure III.1.2:** The UEML template approach allows to include several languages other than the selected languages for developing a UEML core language.

### III.1.5 The UEML template approach and the application scenarios

#### III.1.5.1 Model integration and transformation scenarios

To illustrate this point, there is the need to provide some correspondences between languages. However, as said in Section II.3.1, correspondences are not characterised till now and only an embryonic study has been performed. Therefore, the example provided in this section, overviews an expected result.

Let suppose that two constructs Action and Activity defined in the languages IEM (UEML SoTA, 2002) and ISO/DIS 19440 respectively (details can be found in Annex 5) have been represented by using the “UEML template” (Annex 1). The result is depicted in Figure III.1.3 below: the for these two constructs is represented by two distinct classes (belonging to BWW model) both representing systems being part of some other systems, combined, according to the BWW model, by a specialisation/generalisation relationship. Therefore, it might be possible to conclude that a correspondence exists between the two initial constructs.

![Figure III.1.3](image)

**Figure III.1.3:** An example (UML Object Diagram) of two constructs in two languages, integrated by using the UEML template representing objects created on the underlying meta-meta model depicted in Figure III.1.1 (the bold arrow indicates a “generalisation/specialisation” link).

Unfortunately, UML which is used for representing the Object Diagram above, does not allow to further instantiate the represented objects. However, this is not critical if this further instantiation is
shown by using dependency links. Accordingly (Figure III.1.4 below), if a model element Activity “my production” is represented in a model, it is possible to represent (or to import) this Activity “my production” as a model element Action in IEM because of the specialisation/generalisation hierarchy (Figure III.1.3 above) and because it has been decided that the correspondence should be applied.

Figure III.1.4: An example (UML Object and Class Diagrams) of a simple exchange based on constructs integrated by using the UEML template approach.

More generally, if Action and Activity are used to represent various model elements, the specialisation/generalisation hierarchy only suggests that these model elements are comparable; however, the specialisation/generalisation hierarchy does not force any equivalence (or similarity) between these model elements.

III.1.5.2 UEML core language scenario

The classes/properties/states/events added to the common ontology can be qualified as revealed (as it was in UEML 1.0 where classes in UEML 1.0 meta model were discovered by using existing languages); on one hand, some of these revealed classes/properties/states/events will become closely related to a UEML core language (Figure III.1.5 below); on the other hand, these revealed classes/properties/states/events can be referred to as a revealed common ontology.

Figure III.1.5: The UEML template approach and the UEML core language.
For illustrating how the usage of the UEML template approach can be related to a UEML core language, the ISO/DIS 19440 language is used as an example (details can be found in Annex A5). Applying the UEML template to the construct Enterprise Activity, the result is a Representative Class called the Activity System (the role) actually representing a specific system (the given name is not important). The represented specific systems are the systems represented by the common BWW model based class GoalSeekingComponentSystemThings. This class represents the things that are systems, part of some other surrounding system, and seeking for a goal. Classes like GoalSeekingComponentSystemThings are revealed classes because they are identified through analysis of constructs belonging to existing languages.

Applying the same procedure to several languages, it might be possible to find out in these distinct languages, various constructs intended for modelling GoalSeekingComponentSystemThings. Therefore, it may be possible to create a UEML core construct (belonging to a UEML core language) called UEML_Activity that represents the class GoalSeekingComponentSystemThings. Figure III.1.6 below shows the proposed usage of the UEML template approach according to Figure III.1.5 above.

![Figure III.1.6: An example on how to generate a UEML core construct starting from an existing language.](image)

It is expected that the number of revealed classes/properties/states/events will increase rapidly at first, but that the common ontology will start to grow slowly as soon as a significant number of languages (or significant languages) have been integrated.

### III.1.5.3 New methods and methodologies scenarios

Despite no specific studies have been performed, it seems quite natural to observe that:

- New relationships between constructs in distinct languages can be defined, because the languages (or part of them) are represented and, based on the common ontology, new relationships can be added between constructs; for instance, it could be possible to associate a construct Status to a construct Class because both Status and Class refer to the class Things;
- New methods and methodologies, in the future, can be based on a UEML core language; based on the availability of these languages and their uniform representation (including their representation mappings), and the possible new relationships between constructs, new
methods and methodologies exploiting synergies between the various languages can thereby be realised;

• The correspondences (especially the basic ones) allow the integrated usage of modelling languages.

III.2 Users’ needs and language quality

The experience gained in the UEML Thematic Network project (UEML Deliverable 3.3, 2003) and the internal discussion have revealed two issues that need to be addressed. These issues concern both users’ needs and language quality:

• How to elicit requirements from user needs and how to organise requirements for further usage?
• How to define and to evaluate the language quality?

In the remainder, the most important details about each of these two issues are discussed.

III.2.1 Requirements and their organisation

Due to the complexity, the size, the scope, and the differences in vocabulary, experience and vision, and the number of users potentially involved, there are two major aspects concerning users, which need to be addressed:

• Users have to be motivated to express their needs and to clarify those themselves, as much as possible; in other words, the requirements elicitation and collection should be distributed as much as possible;
• The requirements should be organised according to their importance for UEML.

III.2.1.1 Requirements elicitation and collection

The requirements elicitation and collection method extends the method defined in the UEML Thematic Network (UEML Deliverable 2.1, 2002). The current elicitation and collection method is based on a “requirement template”.

The “requirement template” (Table III.2.1 below) is simple and based on the idea that the users should explain the needs that they submit in more detail. Specifically, the “requirements template” requires reformulating the needs the users are expressing in several ways and with distinct statements.
Table III.2.1: The requirement elicitation and collection template (“requirement template”).

<table>
<thead>
<tr>
<th>Detail level I</th>
<th>Detail level II</th>
<th>Detail level III</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Better refine your first idea by formulating a new and complete sentence</td>
<td>1.1 Formulate a first new concept of (1.) and/or 1.2. Formulate a second new concept (1.)</td>
<td>1.1.1 Reformulate 1.1 (no replication) and/or 1.1.2 Reformulate 1.1 (no replication) and/or 1.1.3 Reformulate 1.1 (no replication)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.2.1 Reformulate 1.2 (no replication) and/or 1.2.2 Reformulate 1.2 (no replication) and/or 1.2.3 Reformulate 1.2 (no replication)</td>
</tr>
</tbody>
</table>

The first column of Table III.2.1 (Detail Level I) represents the main concept contributing to the development of UEML, while the deployment (Detail Levels II and III) represents a better specification of the “problems” and/or “solutions” underlying the identified concept.

Table III.2.2 below shows an example of the proposed template. It is easy to note that the first level of detail is highly improved in the third level of explanation. It should be also easily noted that the general statement “UEML should be able to capture and show the enterprise goals in the models” is further clarified by existing “solutions” and by addressed “problems” in the higher levels of details.

Table III.2.2: An example of filled requirement elicitation and collection template.

<table>
<thead>
<tr>
<th>Name:</th>
<th>Capture enterprise goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>UEML should be able to capture and show the enterprise goals in the models</td>
<td>1.1 The relation between enterprise goal and the model should be always evident</td>
</tr>
<tr>
<td></td>
<td>1.1.2 Enterprise goals means to take into account the context in which enterprise is operating</td>
</tr>
<tr>
<td></td>
<td>1.1.3 A number of such methodologies have been developed eg. Enterprise Knowledge Development (EKD), the i* methodology, Business Modeling with UML</td>
</tr>
<tr>
<td>1. It is necessary to make explicit what the enterprise goals are and how they influence the enterprise analysis</td>
<td>1.2.1 A methodology for goal modelling must be included into the enterprise modelling component</td>
</tr>
<tr>
<td></td>
<td>1.2.2 UEML should use goals as constraints to the modelling and to decision making</td>
</tr>
<tr>
<td></td>
<td>1.2.3 Similar goal oriented methodologies have been developed and used in Sweden for many years</td>
</tr>
</tbody>
</table>
According to the previous discussion and to the elicited requirements (Annex A3) the “requirement template” allows:

- To describe problems and to refine them as proposed solutions,
- To describe abstract ideas and to clarified them in problems and solutions,
- To describe solutions and identify any problems already solved by the proposed solutions.

The example in Table III.2.2 also shows that a single requirement is to be solved by the “composition” of all the levels of detail. In other words, the example in Table III.2.2:

\[
\text{UEML should be able to capture and show the enterprise goals in the models (the relation between enterprise goals and the model should be always evident (each time a scenario simulation is made or a modelling decision is taken, it has to be related to the enterprise goals))}
\]

is a single requirement, represented with a bracket expression.

The collected and elicited requirements are listed in Annex A3: there, user needs are organised according to the categories described in Section III.2.1.2 below.

III.2.1.2 User needs organisation

The organisation the user needs is extremely important for using them efficiently. The organisation should reflect the main application scenarios for UEML. This suggests “problem-oriented” categories of requirements that are organised into three main groups: core, basic and extended. These groups organise categories from generic to specific ones for the purpose of the objective of UEML, i.e. to support the integration and the integrated use of models. Accordingly:

1. core that groups together the categories strictly related to the conceptual understanding of what UEML should support; these fundamental categories address requirements for putting in place the main application scenarios;
2. basic that groups together the categories related to the existing standards and languages i.e. closely related to the usage of UEML in general contexts; indeed, UEML, to become usable, as a first step, should be compliant or should be able to represent objects in existing standards and languages (or to access these objects);
3. extended that groups together the categories envisioning the usage of UEML by humans and also the application of UEML in the practical specific contexts involving existing systems and existing solutions; indeed, indeed, with the extended requirements, UEML should become usable in the real contexts where users are working and systems are running.

It should be noted that while groups are considered stable, additional categories can be proposed if the already defined categories do not represent some needs sufficiently enough.

For organisational reasons in INTEROP, the activities of adding classes (while it is not mandatory) has to stop. However, as better explained in Annex A8, the way of organising the work in INTEROP provides a good understanding of the activities beyond the end of the project.
The proposed categories are listed and explained in the three tables below.

<table>
<thead>
<tr>
<th>Core Category</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 METHODOLOGICAL</td>
<td>This category identifies requirements concerning distinct methods for enabling the several distinct modelling views to be consistent</td>
</tr>
<tr>
<td>2 ORGANISATIONAL</td>
<td>This category identifies requirements associated with capabilities for managing all the aspects related to an organisation (except evolution).</td>
</tr>
<tr>
<td>3 EVOLUTIONARY</td>
<td>This category identifies how UEML could support evolution and dynamical aspects of the organisation or part of it during its life cycle.</td>
</tr>
<tr>
<td>4 STRUCTURAL</td>
<td>This category groups requirements that refer to the architectural structure of UEML, for supporting the management of enterprises models.</td>
</tr>
<tr>
<td>5 OPENNESS</td>
<td>This category groups requirements related to the capability of UEML to be interfaced with other modelling environments or tools.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Basic Category</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 STANDARD SUPPORT</td>
<td>This category groups requirements related to the capability of UEML to be compatible with multiple standards (ISO, IEC, OMG …).</td>
</tr>
<tr>
<td>2 FACILITY</td>
<td>This category consists of requirements referring to the representation of different state/status of object/activity/process and management of flexible performance systems.</td>
</tr>
<tr>
<td>3 LANGUAGE COMAPTIBILITY</td>
<td>This category identifies requirements according to the capability of UEML to import, export and support data and information from multiple languages (IDEF, ODP, BFML, B2MML …).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extended Category</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 HUMAN INTERACTION</td>
<td>This category refers to the capability of UEML to enable the interaction with various kinds of humans.</td>
</tr>
<tr>
<td>2 ENVIRONMENT SUPPORTS</td>
<td>This category identifies the capabilities to support/include/interface other environments (D.E.S., E.R.P., F.M.E.A., M.E.S., S.C.M., C.R.M. etc.).</td>
</tr>
</tbody>
</table>

The identified categories are used to categorise requirements. Details about the categorisation of requirements are in Annex A3. This categorisation has revealed that users tend to concentrate their needs on the STRUCTURAL category (INTEROP Deliverable 5.1, 2005); this result was expected due to the type of participants (playing the role of users). However, additional requirements found by the analysis of the relevant literature are rarer and tend to be concentrated on METHODOLOGICAL category (INTEROP Deliverable 5.1, 2005).

**III.2.2 Language quality**

Evaluating the qualities of languages requires the definition of well-defined and/or measurable quality criteria. For this reason, requirements cannot directly become criteria (Section II.3.1). However, on one hand, the analysis of requirements may suggest quality criteria; on the other hand, the same analysis of requirements might suggest (additional) languages that should be evaluated according their quality.

Accordingly, the internal discussion about the language quality has addressed over the following points:

- Language quality should be defined and criteria for assessing it should be defined;
• Criteria should be related to user needs;
• Criteria should be associated with explicit or implicit methods for evaluating those criteria. Methods should require limited information about the modelling languages to be evaluated;
• The list of the languages that may be evaluated according to the quality criteria should be defined;
• The way to select, according to the evaluated criteria, languages (in the list mentioned above) has to be transparent.

In the remainder, following the points listed above, the main aspects and solutions of the proposed way of working are fully described and explained.

The **quality framework** of (Krogstie et al., 1995) has been applied and extended (Annex A6). This framework essentially provides the neutral definition of the several **quality types** of a language (for instance, **syntactic quality** represents to what extent a language syntax corresponds to the concepts to be represented): neutral means that the quality framework can be applied to languages that are used for modelling without any regards to the specific domains. Each of these quality types is related to what is called **appropriateness** (an example of appropriateness is the **domain appropriateness** that is used to assess **physical** and **semantic qualities**; domain appropriateness is essentially the relationship between the domain of modelling and the way of modelling using that language). The various types of appropriateness provide the context to evaluate the related quality types. The quality framework helps in assessing the coverage degree of the quality criteria, according to the various quality types. Quality criteria can be defined starting from elicited requirements or just independently. However, as explained in Section II.3.1, quality criteria should be related to elicited requirements. The meaning of the relationships between requirements and quality criteria is fully explained in Annex A7. An example of these relationships is provided and shortly discussed below (Table III.2.4).

Figure III.2.1 below represents the recommended relationships between requirements and the quality framework (this figure is a meta meta-model represented as a UML class-diagram).

It should be noted that criteria that should be evaluated may vary over time because, for instance, of new user needs or more detailed analysis of existing needs.

For organisational reasons in INTEROP the evaluation of criteria (while this is not mandatory) has to stop. However, as better explained in Annex A8, the way of organising the work in INTEROP provides a good understanding of the activities beyond the end of the project.
The meaning underlying the figure can be illustrated by the following example. The criterion number 18 (Annex A6) is described by Table III.2.3 below. Table III.2.3 essentially represents the grey part of the Figure III.2.1 above.

Table III.2.3: Example showing how a criterion is defined.

<table>
<thead>
<tr>
<th></th>
<th>Programmable infrastructures</th>
<th>Technical appropriateness</th>
<th>actor</th>
<th>Can the language be used as a code generation skeleton for some programmable infrastructures? Is the language interpreted by any programmable infrastructure?</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The criterion number 18 is then related to several requirements (Table III.2.4 below). This criterion has been defined independently of currently collected requirements and has been related to some them. From that, it can be easily noted that if a language can be used for programming infrastructures, it supports the associated requirements. More precisely, the criterion 18 groups together several requirements and synthesises the main issue addressed by these requirements. Further discussion about relationships between requirements and quality criteria can be found in Annex A7.

Table III.2.4: Example showing a link between a criterion and corresponding requirements.

<table>
<thead>
<tr>
<th>Modelling the complexity</th>
<th>Programmable infrastructures (describe also the type of envisioned infrastructure)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robustness (reactivity);</td>
<td></td>
</tr>
<tr>
<td>Usability (interactive modification of models; reconfigurability; scalability)</td>
<td></td>
</tr>
<tr>
<td>Dynamic modelling (resource/system/process monitoring and feedback overtime; dynamic rescheduling);</td>
<td></td>
</tr>
<tr>
<td>Simulation capabilities (linking to the discrete event simulators)</td>
<td></td>
</tr>
<tr>
<td>Easy to use (enable multiple enterprise user)</td>
<td></td>
</tr>
<tr>
<td>User friendly interfaces (including interactive training facilities; visual representation capabilities)</td>
<td></td>
</tr>
<tr>
<td>Availability of enterprise engineering methodology (tools availability for managing models)</td>
<td></td>
</tr>
<tr>
<td>Usability (specification libraries)</td>
<td></td>
</tr>
</tbody>
</table>

The previous example can also be used to illustrate what is the meaning of the statement “criteria should be associated to explicit or implicit methods: these methods are required to evaluate the criteria”. For instance, in the case of “programmable infrastructures” criterion, there is just the need to collect the relevant information about the language. However, in other cases, as briefly described in the remainder, the methods should be explicitly defined. Therefore, there is the need to distinguish carefully between ways for collecting information about languages and ways to evaluate (i.e. the method) the various quality criteria.

To the aim of defining these methods, there are two possible ways for collecting information about languages to further be used in those methods (Table III.2.5).
Table III.2.5: The two ways for collecting information about languages to be further used by methods for evaluating quality criteria.

<table>
<thead>
<tr>
<th>Way to collect information on languages</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using a &quot;language template&quot; (one per language)</td>
<td>To collect once &quot;factual&quot; information about each language to be elaborated by methods</td>
</tr>
<tr>
<td>Using criteria-driven questionnaires (one per user type of the language)</td>
<td>To perform a statistical evaluation of the quality criteria</td>
</tr>
</tbody>
</table>

The “language template” has been defined for collecting “factual” information about languages (for instance, “number of constructs”) to be used by the methods for evaluating “factual” criteria. The current “language template” can be found in Annex A5. This “language template” should be filled-in with the relevant information by spending a limited effort since these criteria are usually evaluated to eliminate those languages that are not useable for UEML. Therefore, to design the “language template”, the main question is: which “factual” information about languages can and should be collected?

According to Table III.2.5 above, criteria-driven questionnaires can be related to criteria that cannot be evaluated by using “factual” information about languages: for instance (not currently included in the criteria list in Annex 6), a criterion about the “easiness of use of the language” cannot be evaluated with the “language template”; instead, questionnaires and statistical methods have to be used.

However, criteria-driven questionnaires can also be used as a mechanism for distributing the work of evaluating “factual” criteria. In this case, questionnaires are the methods for evaluating the “factual” criteria. For instance, instead of asking “factual” information (i.e. a language meta-model in the “language template”), for further evaluating the “number of constructs” criterion, it is possible to directly ask the “number of constructs”. In this sense, criteria driven questionnaires seem to be an effective alternative to the “language template” especially for “factual” criteria with simple evaluation methods (i.e. there is not need to carefully separate collection of information required for evaluating methods and the definition of methods themselves).

Criteria-driven questionnaires are discussed in the Section IV.1 below because they are part of the future plans (criteria-driven questionnaires have not been discussed in the first phase of INTEROP).

The current list of languages that may be evaluated (Annex A4) has been completed using the available states of the art (UEML SoTA, 2002), (ATHENA SoTA, 2004) and INTEROP Partners’ experience. Requirements can also be used to expand this list of languages over time.

For organisational reasons in INTEROP the list of languages (while this is not mandatory) has to stop. However, as better explained in Annex A8, the way of organising the work in INTEROP provides a good understanding of the activities beyond the end of the project.

The last point to be addressed is the selection of languages for UEML. This is not an easy task because of two main problems (facets of the same one):

- How to treat languages that fulfill some criteria very well and do not fulfill others? (in other words, should criteria be clustered in some way?);
- How to treat contradicting criteria, for instance, a language that is used in practice quite well (criteria 8, 9 in Annex 6) but it is scores poorly in other criteria? (in other words, should criteria be prioritised?).

The consequence is that the language selection can probably not be done at once and that goals for using the UEML template approach have to be included also in the selection process. In other
words, criteria and their evaluation make transparent the selection base, but the selection itself is effectively done within specific situations and according to specific goals.

An example of how criteria and goals are used for language selection is described in Section IV.1 about the experimental application of the UEML template approach. There, the goal is corresponding to <Objective O2> (Section II.3.1). This goal is used to select the criteria (not all criteria are interesting); then, to evaluate the languages in the language list according to the selected criteria; finally, to select the languages accordingly.

Another example about the usage of goals can be the definition of the core concepts for enterprise modelling. Even in this case, the quality criteria have to be used carefully in the selection method. Finally, it is also possible to use the proposed way of working to help in selection processes in specific enterprises facing different user needs and specific goals: As has been pointed out several time in this document (Sections II.3.1, III.1.1), the quality of languages is a decisional tool, which helps in choosing between languages that should be further analysed according to the UEML template approach.

Figure III.2.2 below depicts as tasks and their dependencies in the whole ways of working that have been proposed and discussed in this section.

![Diagram](image)

*Figure III.2.2: Tasks and their dependencies to perform the selection of languages for UEML.*

**III.2.3 Summary**

Sections III.2.1 and III.2.2 present the proposed way of working taking both the requirements and the quality of languages into account. Requirements and language quality have been formally
related by introducing a meta-meta-model (Figure III.2.1). Accordingly, a set of quality criteria has been introduced and the related requirements have been established. To evaluate these quality criteria, a way based on collecting “factual” information on several languages, has been introduced. This way is essentially a “language template” that has to be filled with the relevant information. However, additional ways to evaluate quality criteria can be envisioned: the most relevant one concerns the definition of criteria-driven questionnaire.

However, once several criteria have been evaluated, the selection of languages is not an easy task. Indeed, it is quite obvious to have contradicting criteria; i.e. criteria for which some languages score good and others score poorly. Therefore, in the selection task, selected goals providing additional information about why the UEML template approach is used, should be introduced. These goals can also be used to limit the number of criteria to be evaluated as well.

### III.3 Experimental application

According to the criteria shown in Annex 6 (Table II.A6.1), the appropriateness and related criteria applied to the current list of languages (Annex A4) has been the “organisational appropriateness” (criteria 8,9,10,11 in Annex 6) and especially the “availability of skills” (criterion 11 in Annex 6). In Table III.3.1 below, there is the list of the languages that have been analysed using the “language template”. This list of languages seems interesting because it allows to understand the effect of combining, through the UEML template approach, languages originating from distinct application domains (but related to enterprise modelling). This is in line with the <Objective O2> (in Section II.3.1), which requires to evaluate to what extent the UEML template approach is suitable for representing enterprise modelling languages.

<table>
<thead>
<tr>
<th>Language(s)</th>
<th>Information from the “language template” used to evaluate the criteria 8,9,10,11</th>
</tr>
</thead>
<tbody>
<tr>
<td>PetriNets</td>
<td>Wide community, skill is available</td>
</tr>
<tr>
<td>IDEF (0,1,3) (many sub-languages; only 0 and 3 have been described in the “language template”)</td>
<td>Wide community, skill is available</td>
</tr>
<tr>
<td>WPDL/XPDL</td>
<td>Wide community</td>
</tr>
<tr>
<td>ISO DIS 19440</td>
<td>Community of enterprise modelling, skill is available</td>
</tr>
<tr>
<td>BPMN/BPML</td>
<td>Wide community, skill is available</td>
</tr>
<tr>
<td>ebXML/UBL</td>
<td>There are no significant information on this language, skill is not longer available, however, based on the corresponding filled “language template”, the various meta models of UMM seem to be appropriate for UEML</td>
</tr>
<tr>
<td>I*/GRL</td>
<td>Wide community, skill is available</td>
</tr>
<tr>
<td>UML 2.0</td>
<td>Wide community, skill is available</td>
</tr>
<tr>
<td>AUML</td>
<td>It seems no longer alive (the language AML seems to replace AUML), skill is available</td>
</tr>
</tbody>
</table>

Among the languages analysed using the “language template” (Annex A5), two have been eliminated (that is, not considered when filling-in the “UEML template”). The reasons have been:

- **AUML** seems to be replaced by **AML** (and it seems not longer alive, criterion 13 in Annex 6),
- **ebXML/UBL** skill is no longer available in the work-package (criterion 11 in Annex 6).
The filled-in “UEML templates” are contained in Annex A2. The experimental application of the UEML template approach revealed the underestimation of the difficulties of participants. Using the UEML template to define the modelling constructs provided by the languages listed above has turned out to be a challenging task for two main reasons. Firstly, the UEML template itself is hard to use, in particular because it requires a practical understanding of the sometimes complex ontological ideas behind the template, many of them inherited from the BWW-model and Bunge’s ontological theory. Secondly, some of the languages listed above themselves have weakly and unclear defined construct (often the construct definition is not fully expressed and it just represents the underlying thinking of the community that has built the language).

The critical aspects that have been pointed out are:

- What a construct is really comprising? (for instance, how is the construct composed and what are the properties of the construct. Often these two aspects are mixed, especially in the language meta-models; for instance, in the ISO/DIS 19440, the EnterpriseActivity is the construct while the Objective is a property; in Coloured Petri Nets, the Arrow is the construct but being incoming in a Transition or outgoing from a Transition are properties of Transition and not properties of Place).

- The representational mapping of classes, properties, states and events are difficult to establish and understand because they depend on an existing ontological model (or upper ontology). For example, we have applied the BWW/Bunge ontological foundation, using concepts such as “mutual properties”, “(state/transition) law properties” and “whole-part relationship”. Some knowledge of the BWW model and Bunge’s ontological model is therefore required, although other ontological foundations might in principle can be used instead.

- The behavioural aspects in the UEML template are based on two simple concepts i.e. state and event (this requires to precisely describe some behavioural aspects of the constructs with very basic concepts as events and states; on one hand, the precise definition of these events and states may become complex; on the other hand, the precise definition of these events and states may become difficult because these events and states are poorly defined in the original description of the language constructs; for instance, this point has been approached by using the ISO/DIS 19440 and it was difficult to characterise how events are related to states. In fact, states are not explicit in this standard).

Each of these critical aspects should be carefully addressed by further actions. To this aim, one possible action is to develop a tutorial on the UEML template approach. The other possible action is to develop software tools to support the UEML template approach. These two points are better addressed in Section IV.1 about the future plans and the definition of refined research objectives.
Part IV Future Plans and conclusions

IV.1 Future plans

This section provides a further definition of research objectives. These objectives refine the previously defined ones (Section II.3). All the new research objectives have been based on the experimental application of the UEML template approach and on the analysis about the status of quality criteria. As described in Section II.3.1, these research objectives can be addressed with various degrees in the context of INTEROP.

IV.1.1 The UEML template approach

Concerning the UEML template approach, starting from the experimental application and the initial research objectives, the following objectives have been defined:

- **Language breadth:** the UEML core language should be broadened by incorporating more languages, frameworks and standards for representing enterprises and related domains. In particular, more industrial modelling languages, frameworks and standards should be considered. Although the main focus of UEML is enterprise modelling, it can also incorporate modelling languages for related domains, such as information systems modelling and ontology languages.

- **Ontological depth:** Initial experience with UEML versions 2.0 and 2.1 shows that the common ontology can be described at different levels of precision. For example, many modelling constructs that have been encountered represent active things (things that transform themselves from certain states to certain other states). But these constructs do not all represent the same subtype of active things. Some of them represent active things with only one input and one output; others allow many inputs and outputs. Some of them represent active things that consume all their inputs and produce all their outputs synchronously, other behave asynchronously. Some have typed inputs and outputs, others not etc.

  In the common ontology, it is possible to represent all these variants and their combinations separately (an ambitious, fine-grained approach), or it is possible to represent only the class of active things (a simple, coarse approach). A fine-grained ontology is harder to use and manage, but allows to support a tighter integration and is preferable when precise modelling is required. A less detailed ontology is smaller, easier to manage and user-friendlier, and is needed when describing less ontologically precise languages. It should be investigated whether the common ontology should offer both coarse and fine-grained concepts (of course arranged hierarchically).

- **Ontological clarity:** The ontological grounding of UEML should be elaborated. Whereas the previous point (ontological depth) dealt with the number and precision of phenomena in the common ontology, this point deals with the (“upper”) ontological concepts used to describe the common ontology and how well they are defined. Because the common ontology is based on Bunge’s ontological model and the Bunge-Wand-Weber (BWW) representation model, it is already more elaborately described than most competing ontologies, but its description can always be further elaborated. For example, UEML may increasingly come to incorporate languages that represent social constructs and mental
concepts in addition to material objects. For these purposes, the Bunge/BWW-model ontological concepts are not equally clear. Also, the meta-meta model on which the UEML template is based should be evolved correspondingly. (Mathematical clarity, or mathematical formality, is a related issue, which we discuss in a separate point below.)

- **Presentation:** the UEML-template should manage how models are presented visually not only what the models represent. The representation part (previously called the “semantics part”) is the most innovative and challenging part of the UEML approach. The presentation part (“syntax part”) has therefore been less focused on so far. Future versions of UEML template should support both presentation and representation and should support verification of their mutual consistency. For example, presentational (or syntactical) relationship between two modelling constructs matched by a corresponding representational relationship (i.e., by a mutual property in the common ontology) and vice versa? So far, the presentation part of UEML template has focused on conventional boxes-and-arrows diagrams. It should be extended to account for the widest possible variety of external presentations, because the future of enterprise modelling lies beyond static boxes-and-arrows diagrams. Today’s dynamic simulation and animation of behavioural models are only a first step towards immersive simulation/animation of 3-dimensional worlds generated from models.

- **Mathematical formality:** the UEML-template should be defined formally. The formal definition should include at least: Definitions of the “upper” ontology concepts from Bunge/the BWW-model that are used to define the common ontology. Provisions should be made for defining the phenomena in the common ontology by refining the definitions of the “upper” ontology concepts. The representation and presentation parts of the template should also be formalised, and provisions should be made for describing modelling constructs formally using the template. For example, in the current version of the UEML template (version 1.2), state and transformation laws are described informally using natural language or an arbitrary formal notation. Future versions of the template should offer a standardised formal notation for describing laws. Provisions should be made for ensuring that the representation part of a construct definition is self-consistent, e.g., that the formal law descriptions have as unbound parameters only other ontology properties of the thing that possesses the law. It should also be possible to ensure that the presentation part of a construct description is consistent with itself and with the representation part. Further provisions should be made to ensure presentational and representational consistency between all the constructs in a language and between languages. A particularly challenging task is ensuring that the various laws implied by different modelling constructs are consistent with one another, in particular when those constructs belong to the same language. Although this problem cannot be fully solved formally, automated reasoning may help. Further work should also address the feasibility and importance of supporting other formal properties than consistency for UEML.

- **Tool support:** Tools should be developed that make the UEML template approach easier to manage and use. Currently, the representation part of UEML is supported by a prototype repository implemented in Protégé-OWL. A first priority is to provide a more suitable user interface to this tool. A second priority is to provide simple analysis features for validating construct descriptions, e.g., checking that the descriptions adhere to the cardinality constraints expressed in the meta-meta model and enforcing other constraints presented in (Opdahl & Henderson-Sellers, 2005). Generating Prolog code is one viable implementation path. Further work can take several directions. One direction is supporting the presentation
part in addition to the representation part, along with the corresponding validation mechanisms. Another direction is to support formal construct descriptions, i.e., with standard notations for defining state and transformation laws. Expressing and validating the consistency of fully formal construct descriptions will likely require stronger formal support than offered by Prolog. The current plan is envisioning to use Alloy (http://alloy.mit.edu/) to express and reason about such descriptions.

- **Model management:** tools supporting the UEML approach should support management and use of enterprise models in addition to the management of languages. According to the vision, without supporting the model level, UEML will remain a preparation for functions such as cross-language consistency checking, automatic update reflection, cross-language model translation etc. A UEML that supports the model level can potentially actually support these and related functions. The focus of model-management support should be utilising the language coordination and integration facilities provided by UEML, in order to support better coordination and integration of the models expressed in those languages: a first step in integration and integrated use of models expressed in different languages is necessarily integrating their languages.

- **Validation:** the UEML approach should be validated empirically using a variety of research methods. Current language analysis and description work indicates that the UEML template approach is indeed sufficiently powerful to incorporate a wide range of industrial, standardised or academic enterprise modelling languages. This work must be continued and extended. As soon as UEML has been extended to support model management in addition to language management, it will be necessary to show that the approach is indeed sufficiently powerful to support functions such as cross-language consistency checking, automatic update reflection, cross-language model translation etc. A possible starting point is to manually replicate existing results provided by other approaches. When a tool support for model management becomes available, more elaborate case studies could be undertaken using real industrial models, preparing for eventual action research studies of UEML core language and model management in real organisations.

- **Standardisation:** the UEML approach should become the international standard for exchanging enterprise models and other representations of enterprises and related domains. The appropriate standard organisations for achieving this must be addressed. The appropriate positions relative to existing and emerging standards must also be defined. For example, there is clearly a need for a more specific and powerful exchange standard than what is provided by the OMG's XML Model Interchange (MXI) format.

- **Dissemination:** the UEML approach should be disseminated widely to both industry and academia. Dissemination tasks include developing written, oral and interactive (internet) tutorials and providing high-quality examples Technical documentation must be produced. Research results must be presented in both academic and professional journals and conferences.

Not all these research objectives can be accomplished inside INTEROP. Therefore, the current proposal for the work inside INTEROP concerns the following points:
1. Simple tool support, including a user interface and simple consistency checking. This is the first priority, because it is becoming hard to continue UEML development without tool support.

2. A second priority is dissemination throughout tutorials and presentations: this kind of activities is already started and an initial tutorial on the BWW-model has been delivered to the involved participants.\(^8\)

3. A third priority is formalisation of the meta-meta model of the UEML template approach. This formalisation should be performed to a certain extent before it can support model management and be realised by a tool.

Additionally, at least embryonic activities about the following points should be performed:

- Model management as well as language management, supported by a tool. Model management must be in place to a certain extent before UEML can be meaningfully validated empirically.
- Empirical validation of tool-supported language and model management. Empirical validation is obviously necessary to ensure industrial uptake of UEML. Empirical validation is also necessary of UEML work shall contribute to theory.

IV.1.2 The quality of languages and its use

According to the tasks depicted in Figure III.2.2 (Section III.2.2), the following research objectives have been defined.

- **To analyse (additional) languages with the language template.** (Additional) Languages from the (current) list of enterprise modelling languages (Annex A4) should be analysed.

- **To refine criteria and align with requirements.** Though there is already an alignment between requirements and criteria (Annex A7), refinements of criteria should be performed and the required relationships with requirements should be defined.

- **To update the language template for criteria evaluation.** Some criteria can be evaluated according to information gathered with the language templates. According to Section III.2.2 whenever criteria are refined and improved, the language template should be extended for capturing the relevant information.

- **To define/to refine a criteria-driven questionnaire for language users.** According to Section III.2.2, some criteria should be evaluated by using criteria-driven questionnaires. These questionnaires are usually submitted to end-users, i.e. users using languages for modelling (in real contexts). To this aim, the questionnaire should be defined by using a vocabulary easy to understand and to reduce the risk of misunderstanding.

- **To evaluate languages according to the criteria** (by using filled-in language templates and questionnaires).

\(^8\) The INTEROP WP10 has the required skill to support the work.
• To define selection criteria and select high quality languages for a UEML core language. There is the need to establish which criteria are mandatory to accomplish the UEML core language goal, and to select the relevant enterprise modelling languages according to the languages that got best marks when evaluated.

• To analyse how languages for which a filled-in language template exists are mutually complementary (i.e. to what extent languages might fit-together in an holistic architecture). Enterprise modelling language can be used for achieving specific targets. Specifically, they can be used to analyse, to enact and/or to simulate models. If several languages are used together, it is interesting to look at existing enterprise (reference) architectures to assess the coverage of the languages for which a filled-in language template exists.

• To perform case studies with the languages for evaluating criteria related to the domain appropriateness. Theoretical studies though interesting are not complete and remain subjective. To deeply evaluating languages, it can also be interesting to define empirical methods based on real cases. Therefore, for complementing the information gathered in the languages template and the questionnaire, it is interesting to perform complex case studies with the different enterprise modelling languages.

Not all these research objectives can be accomplished inside INTEROP. Therefore, the current proposal for the work inside INTEROP concerns the following points:

• To refine criteria and align with requirements. This task will improve the current criteria (Annex 6) and the relationships to existing requirements (Annex 3 and Annex 7).

• To update language template for criteria evaluation. This task will explore what “factual” information about languages can be additionally collected for further evaluation of quality criteria. The result will improve the current language template (Annex 4).

• To define/to refine a criteria driven questionnaire for language users. This task will develop a criteria-driven questionnaire concerning both “factual” criteria and “subjective” criteria.

• To define selection criteria for a UEML core language. This task will suggest the criteria that should be taken into account for the development of a UEML core language.

Finally, at least embryonic activities about the following points should be performed:

• To analyse (additional) languages with the language template. This task will record the possible additions of new languages to the current list of languages (Annex 4). The task could possibly update the currently existing filled-in language templates (Annex 5) according to the new version of the language template (point 2 above) or add new filled-in language templates for additional languages.

• To evaluate languages according to the criteria (by using filled-in language templates and questionnaires). This task could be performed to evaluate the validity and the easiness of usage of the language template.

• To select high quality languages for a UEML core language. This task could be performed to suggest languages to be taken into account in the development of a UEML core language. However, this task is dependent on the previous one.
IV.1.3 Correspondences between languages

The basic correspondences (Section II.3.1) between constructs in distinct languages are the suitable mechanism to support the (conceptual) enterprise model management recommended in Section IV.1.1. Model management is broader than the problem of correspondences but it seems that the correspondences are at the base for implementing a sound and advanced model management. As said in Section II.3.1, to characterise the basic correspondences between languages, there is the need to formalise the common ontology. Therefore, it is proposed to concentrate the effort on this formalisation and to start, at the same times, studies about these basic correspondences.
IV.2 Conclusions

This document presents the vision, the challenges, the research objectives and the adopted solutions about UEML, currently named UEML 2.1. As said in the Introduction, UEML 2.1 is not just a “language for enterprise modelling”. Though, what is currently named UEML 2.1 comprises theories and mechanisms for “modelling enterprise modelling languages”, (embryonic) theories and mechanisms for characterising and finding correspondences between constructs in distinct languages and, finally, politics for selecting/classifying modelling languages driven by users’ needs.

As it has been clearly pointed out, research objectives address the challenges listed in Section II.2.3 and therefore the adopted solutions need to be validated from at least, scientifically and practically points of view. These solutions need also to be continuously improved. However, the principles on which the solutions are based are stable and are the key for understanding UEML.

In INTEROP and beyond INTEROP, the current participants will continue the work towards the scientific validation. Nevertheless, there is the need to continue the practical validation which can take place in enterprises. For this second kind of validation, the participants should be engaged in very actively in dissemination. Indeed, practical validation should take place in real contexts like enterprises, groups of enterprises and standardisation bodies.

The performed work and the achieved results are anyway an important advance on UEML that provide a sound base to continue the work towards the vision.
List of annexes

The list of annexes is provided below. The content of each annex is explained and backward referenced to sections and annexes in which that content is relevant.

A1: The “UEML template”. It describes the several sections of the “UEML template” that has been defined on a meta-meta model and used for deep analysis of single constructs of enterprise modelling languages. Section III.1.3

A2: The filled-in “UEML templates”. It contains all the filled-in “UEML templates” for a set of relevant languages. Section III.1, Section III.3.

A3: The filled-in “requirement templates” and their categorisation. It contains all the filled-in “requirement templates” and the categorisation of the users’ needs. Section III.2.1.

A4: The initial list of languages. It contains the list of languages to be potentially taken into account for UEML. Section III.2.2.

A5: The “language template” and the filled-in “language templates”. It contains the structure of “language template” used to collect “objective” information about the languages (in the initial list of languages). It also contains the filled-in “language templates” for the subset of languages in Table III.3.1 above. Section III.2.2, Annex A 4, Section III.3.

A6: The extended quality framework and the identified quality criteria. It contains the extended quality framework relating requirements to qualities. It also contains the defined quality criteria. Section III.2.2

A7: The links between quality criteria and requirements. It contains the full definition of the relationships between the current requirements (Annex 3) and quality criteria (Annex 6). Section III.2.2, Annex 3, Annex 6.

A8: The organisation model. It contains the description of the organisation model used to perform the work in the first phase on INTEROP. Sections III.2.1 and III.2.2.
Technical glossary

This glossary of the work package is intended to support the clear and consistent definition of terms used all documents and presentations related to UEML. The glossary is based on definitions provided by the contributing experts or introduced during the collaborative work. This specific glossary has to be added to the whole INTEROP glossary.

**Language or modelling language:** The basic set of constructs used for modelling. A language is often conceived to contain (at least) a syntax and semantics:

- Syntax can be related both to the internal structure of languages and models, i.e., abstract syntax; Syntax can also be related to external model presentation aspects such as visual icons, edge types and layout, i.e., concrete syntax.
- Semantics is related to model content, i.e., to what languages and models represent, refer to, are about, etc. Because some researchers find the term semantics problematic, we will instead talk about representation in this report.

Syntax and representation can be formal or informal. The syntactical and representational aspects of one language can be described by other languages. A formal language is typically described by another, formal language (a “mathematical language”).

**Enterprise modelling language (EML):** A modelling language used for representing enterprises, usually supporting some analysis (formal or informal) of these enterprises. Some languages are fully qualified as enterprise modelling languages because they are historically related to the field of enterprise modelling. In addition, various other languages are used for modelling enterprises even if these languages were not created specifically for enterprise modelling.

**Construct:** An component of a language that can be used to express a model element. Typically, each modelling construct in a language can be identified by its having a specific visual presentation.

**Model element:** A self-consistent element which is an element of a model. This element is naturally related to syntax, concrete syntax and abstract syntax.

**Model:** A self-consistent and complete set of elements built by using the constructs of a language.

**Meta model:** A meta model is a description (or model) of a modelling language. Any language can, in principle, be used for meta modelling other languages (for instance, parsers and compilers). The meta model is also a way of describing the constructs of a modelling language.

**Meta-meta model:** A meta model that describes a language used for meta modelling.

**Syntax:** A general concept to indicate the set of possible model elements that can be built according to, for instance, a meta model or other kind of language definition.

**Abstract syntax:** A way to formally describe the internal (or underlying) structure of languages and models. Abstract syntax is often a starting point for describing a language formally (“formal semantics”). The abstract syntax of a language can be described in a meta model.

**Concrete syntax:** A way to formally describe the external presentation of (diagrammatic) models, including, e.g., their icons, nodes, lines and arrows, their sizes, textures and shapes, as well as rules for which icons, nodes, lines, arrows etc. that can connect to which others. Conventionally, however, concrete syntax does not deal with less formal aspects of external model appearance, such as layout conventions.

**Domain:** A part of the world represented by a model or intended to be represented by a language. For example the enterprise domain, for models of and languages for enterprises or, more generally, the concrete domain, comprised by all material/physical as opposed to conceptual/mental domains. Every language or model that is intended to represent something has a domain, even if that domain is not explicitly described.

**Ontology:** A model that describes the basic elements of a domain. In particular, an ontology can be used to describe the implicit domain of a language. (Many approaches to “formal semantics” do not
distinguish between the domain and the ontology, referring to both as the “semantic domain”, a term we therefore avoid.)

**Representation mapping:** A set of relationships between the abstract syntax of a modelling language and an ontology describing the (implicit) domain of that language. The set of relationships implicitly assigns meaning to models by relating all their elements to the domain. A representation mapping can thereby cover an important aspect of the “semantics” of many languages.

**Formal semantics:** A way to describe mathematically the meaning of constructs belonging to a language. Usually there are several formal semantics of a language that are related by, usually, equivalence (but it is not limited to); these several formal semantics are usually related to specific objectives. Formal semantics thereby account for additional aspects of “semantics” which representation mappings may not cover.

**Domain language, ontology language:** A language used for describing a domain, i.e., for describing an ontology.

**Syntax language:** A language used for describing syntactical aspects of a language, such as concrete syntax, abstract syntax, layout conventions etc.

**Layout:** Specifically in a visual language, i.e., in a language equipped with a diagrammatic (or graph) syntax, how the various nodes and arrows in the diagram are depicted.

**BWW model:** The Bunge-Weber-Wand ontological representation model; an adaptation of Mario Bunge's ontological model.

**Integration and integrated use of (enterprise) models:** This definition indicates any activity using models in a coordinated way to perform the activity itself (and maybe to provide models as activity output) and activity aiming to integrate (in general sense) models.
References


