Specifications, not Meta-Models

James Skene
Department of Computer Science
University College London
Malet Place
London, UK, WC1E 6BT
j.skene@cs.ucl.ac.uk

Wolfgang Emmerich
Department of Computer Science
University College London
Malet Place
London, UK, WC1E 6BT
w.emmerich@cs.ucl.ac.uk

ABSTRACT

In a model-driven development, software engineers will have to manage multiple artifacts expressed in several languages. Current meta-modelling and concrete syntax standards fail to adequately preserve a link between artifacts and the languages in which they are expressed, potentially leading to inconsistencies and misunderstandings both in the production and reuse of artifacts. Standards such as XMI and JMI permit the meta-model of an artifact to be accessed. However, such meta-models primarily define the syntax of a language. A full semantic definition requires a supporting document, the language specification, which is typically not explicitly referenced. In this paper we argue that the role of meta-models and specifications should be combined to eliminate this ambiguity. We describe the possible impact on OMG standards and standardisation processes that this would have. We present an open-source project that implements this philosophy, and a case-study in which a domain-specific language is used to express service-level agreements, the legalistic nature of which imposes strong requirements for semantic accessibility.

Categories and Subject Descriptors

F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages

General Terms

Languages

Keywords

MDA, MOF, UML, OCL, meta-modelling

1. INTRODUCTION

It is frequently important that the meaning of software development artifacts, such as requirements specifications, models, and even program code, is preserved across time and space. Several people may work in a software development endeavor, and they will communicate not only verbally but through shared artifacts. It is reasonable to expect that any reader or modifier of a software development artifact should be able to understand it in the same terms as its original author. Where ambiguity exists in an artifact, disambiguation may be possible by contacting the author of the artifact, but this is not always true. The author may be unavailable for some reason, or the author and recipient may be the same person, separated only by an interval of time during which the vital information has been forgotten. Development artifacts should be as unambiguous as possible in their meaning, in isolation from the original author and the conditions under which they were authored.

A lack of semantic ambiguity is particularly important in the context of a model-driven development. Although no strong definition of what constitutes such a development exists, it seems likely that the primary type of artifact in such developments will be models, expressed in languages defined using OMG meta-modelling technologies. Furthermore, given the dichotomy that exists in most formulations of the MDA approach between concepts that are in some sense generic, or 'platform-independent', and those that are specific to particular concrete implementations of a system, it seems likely that several languages, or language variants will be used in any given development [9]. Because the meaning of software development artifacts is partly determined by the meaning of constructs in the languages in which they are defined, this proliferation of languages introduces several pitfalls for development. First, the true intent of a developer may not be captured by an artifact because a developer has failed to correctly understand the meaning of a language construct that they have employed, a possibility that increases in likelihood with the number of languages used. Second, it is the claim of this paper that in current MDA standards nothing guarantees that a link between artifact and a full syntactic and semantic definition of the language in which it is written can be followed, potentially leading to ambiguity in later interpretations of the artifact. The latter issue is of acute importance in MDA development as it is a principal claim of the approach that systems can be redeployed as hardware and middleware standards change, depending on the reuse of models over an extended period of time.

In this paper we argue for some relatively minor revisions to existing OMG standards [1] to address these issues. OMG standards represent a strong force for consensus in the MDA community, and we believe this approach lends greater rel-

GAMMA'06, May 22, 2006, Shanghai, China.

evance to our work than the introduction of radical new modelling approaches. In this paper OMG standards are referred to without additional citation.

In the next section we look at how modelling languages are defined in existing standards. We highlight the role that natural language plays in all approaches, regardless of their apparent degree of 'formality', as the ultimate method of semantic denotation. We also consider various schemes by which definitions of syntactic elements are associated with instances of that syntax.

Based on these analyses we propose that all meta-models should embed some definitive documentation that is at least expressed in natural language, or some form that is ultimately documented in natural language. We call such documented meta-models *specifications* because in practice they resemble the specification documents published by the OMG. We describe the contribution of this prescription to addressing the twin problems of capturing and retrieving developer intention. Capturing developer intention is assisted because specifications embed language documentation that is accessible whenever the specification is in use. Any tool that utilises the specification, for example as the meta-model accessible via JMI reflection, will also be able to access this documentation and present it to the user. Recovering developer intention is rendered straightforward by the prescription that any artifact should reference its specification, and hence its semantic documentation, and any relevant concrete syntax standards. We discuss the possible impact of these prescriptions on OMG standardisation processes.

Next we describe the UCL MDA tools, a set of command-line interface tools for model-driven development built to conform to the OMG EMOF, OCL, and XMI standards, and JCP JMI standard[7], but incorporating extensions to enable the approach to domain-specific language development and documentation advocated in this paper.

Finally we describe the use of the UCL MDA tools in the development of a novel domain-specific language for Service-Level Agreements (SLAs) in the electronic-service domain, called SLAng. SLAs have a particularly strong need to preserve traceability from the concrete expression of the SLA to the semantics of the language. As potential components of legal agreements the meaning of an SLA must be as completely and precisely defined as possible. Clearly an SLA containing no reference to its semantics is not a sound basis for an agreement as the meaning of the concrete document can be disputed after an agreement has been made. SLAs potentially have a high commercial value, so the consequences of such disagreements can be dire. This example highlights the importance of the association of software engineering artifacts with full definitions of the languages in which they are expressed, as in many cases variations in the interpretation of an artifact can have serious financial repercussions.

The remainder of this paper is structured as follows: In the next section we review approaches taken to defining languages in OMG standards, and the extent to which concrete syntax standards allow reference to be made to the language in which an artifact is written. In Section 3 we outline our proposals for improvements to these standards. In Section 4 we describe the UCL MDA tools, an open source project implementing tool support for these proposals. In Section 5 we describe the SLAng SLA language, a case study for these proposals. Finally, in Section 6 we summarise.

2. RELATED WORK

2.1 Defining modelling languages

It is perhaps a surprising observation that English is currently the state-of-the-art language for defining full modelling languages, such as UML and MOF, in OMG standards. These languages have an abstract syntax, several concrete syntaxes, and semantics. The abstract syntaxes are structured as MOF models. However, they are not generally defined by a concrete artifact expressed in the MOF language, such as a MOF XMI file. Instead they are described in a specification PDF. In UML 1.5, the structure of the meta-model was definitely established using a combination of class diagrams and supporting natural language descriptions, in English. The English descriptions are necessary to disambiguate the diagrams, which occasionally suppress details and which systematically omit any definition of their context. In UML 2.0 the use of diagrams is deprecated to purely informal. The structure of the meta-model is definitively established in 'formal concept definitions' associated with elements in the abstract syntax, and structured according to the features of the MOF type of the element being defined. Having defined the syntax of the language in English, albeit a systematic English description of a MOF model, these specifications also use English to define the semantics of the language.

Concrete syntaxes are also defined in standards documents that are primarily human readable, either within a language specification document, as is the case for UML notation, or in separate standards that define generic concrete syntaxes for languages benefitting from an abstract syntax defined using MOF, such as XMI and HUTN.

In some cases the OMG publishes the meta-models of their languages in a machine readable form, although not necessarily as MOF XMI. These meta-models are not definitive of the syntax of the language, the language specification is. Neither are they definitive of the semantics of the languages. MOF, in its current state, is not naturally expressive of semantics, and in any case, the semantics are definitively established by the natural language statements included in the specification document.

Other approaches to defining MDA languages have also been applied. A number of light-weight extensions have been proposed for UML 1.5 and documented as standard profiles. These profiles extend the syntax of UML 1.5 using tables of stereotypes and tagged values. The incapacity of such extensions to modify the meta-model of the UML to reflect their own domain of interest has led to the common practice of providing a domain model, with reference to which the semantics of the profile are defined. Examples following this approach are the Profile for Schedulability, Performance and Time Specification, and the Enterprise Distributed-Object Computing Profile. In this case, natural language definitions attach to elements in the domain model. Descriptions attached to profile elements establish an equivalence between these elements and instances of the domain model, thereby establishing the semantics of the profile element.

The model-denotational style of semantic definition for meta-models pioneered by the Precise UML group [5], and employed in its submissions to the UML 2 standardisation effort, extends this notion by formalising the relationship between syntactic elements and domain model elements using standard meta-model relationships and constraints. In

practice the syntactic model and the domain model form a joint meta-model for the language, in which the notion that all meta-model elements are elements of the abstract syntax of a language is dropped.

The model denotational approach has found some adoption in standards. The OCL 2 specification provides such a semantic.The semantics of OCL 2 are defined in terms of expression evaluation events which are somewhat different in structure to the underlying expressions (loop evaluations are rolled out, for example). Also, the style is used to define an 'abstract semantics' for CMOF in the MOF 2 specification.

A number of attempts have been made to introduce more traditional styles of semantic definition of languages with MOF-defined abstract syntaxes. A popular option has been to employ a traditional mathematical approach based on logic and set theory. A 'formal' semantics has been proposed for OCL, for example, although it is not definitive, nor is equivalence with the definitive model denotational semantics proven [10].

These approaches all appear to conform to three observations. First, the definitive reference for a language is a specification document, not a technical artifact such as an XMI document of a meta-model. Second, several approaches introduce intermediate models in order to add structure to the semantics description not otherwise afforded by the definition of the language. And third, the semantics of the languages are always ultimately defined, directly or indirectly, by natural language statements. Remove these statements from any formal description of a language, and the description is likely to become impenetrable and useless, regardless of the sophistication of any intermediate models employed.

2.2 Referencing languages from models

Arguably, only three standards for concrete syntax for MOF defined languages are published by the OMG. These are the XML Metadata Interchange (XMI) standard, the Human-Usable Textual Notation (HUTN) standard, and the diagrammatic concrete notation recommendations included in the UML superstructure specification. Also of note is the UML Diagram interchange standard, which allows the encoding of UML diagrammatic notation as XMI or SVG [12]. However, it does not introduce any new concrete syntax.

In this section we consider the degree to which concrete artifacts expressed in these syntaxes reference the syntax and semantics of the language in which they are written.

The XMI standard has undergone a number of revisions, and implementations are in use according to several of these. XMI standards map MOF models to XML grammars. In XMI 1.2 a mapping to a an XML DTD is defined. According to standard XML syntax, this DTD may be referenced in the header of any document, thereby identifying the syntax of the language in which the document is written. Furthermore, the XMI 1.2 standard acknowledges that DTD syntax is not as expressive of syntactic constraints as MOF models. Therefore an XMI specific header element may be included that includes an optional reference to an XMI file containing the MOF model for the language, providing a better reference for the expected syntax.

This scheme is inadequate in several respects. No unambiguous reference is made to the XMI standard, so an interpreter is not guaranteed to be able to identify a sound basis for interpreting the document. This might potentially hinder the identification and interpretation of the meta-model XMI as a reference for the syntax of the document. Interpretation of the link to the specific syntax of the language essentially relies on the syntax and semantics of the document already being understood to some extent.

Another important inadequacy in the XMI 1.2 language referencing scheme is that the semantics of the meta-model are not referred to, either from the instance document, or from the referenced XMI for the meta-model. In fact, in the case of UML and MOF the referenced meta-model cannot even be considered to be a definitive statement of the abstract syntax of the language being used, as this is finally established by natural language statements in the specifications.

XMI versions 2.0 and 2.1 revise the standard to generate XML schema, rather than DTDs. Schema specifications of the grammar of an XML file are more expressive than DTDs, and can therefore capture more of the constraints inherent in the source meta-models. Perhaps as a result, the ability to reference the meta-model XMI has been dropped from the standard. This is deeply to be regretted, as no link is preserved to the true syntax of the language. Also, a clue as to the possible applicable language specification has been removed.

XMI version 1.2 and 2.0 depend on version 1.4 of the MOF standard. Version 2.1 of the XMI standard depends on version 2.0 of the MOF standard. Both versions of the MOF standard allow the embedding of syntactic constraints in an arbitrary constraint language. A common constraint language employed is OCL, which is more expressive than XML schema. In no version of XMI are these constraints mapped into either DTD or XMI schema. Therefore XMI version 2.0 and 2.1 are seriously negligent in not referencing the meta-model definition, as some syntactic constraints will be entirely inaccessible.

Similar to the XMI standard the HUTN maps MOF models to a grammar, in this case specified using a form of Backus-Naur notation, and intended to be easy for humans to use. The mapping from meta-model to grammar is customisable using a 'configuration', an instance of a configuration meta-model. Details of the configuration used can be included in a discriminated comment at the start of a document in a HUTN. Although configurations can reference elements in a meta-model using their fully-qualified MOF names, nothing in the configuration or elsewhere in a HUTN document references the location of a concrete artifact defining the meta-model for the language used. Moreover, no reference need be made to the HUTN standard, and the inclusion of configuration information is optional, so confusion can potentially arise regarding the syntax to which the document is intended to conform. Finally, no reference need be made to any semantic specification of the language in the document.

UML diagrams are an extremely well-known notation. However, they are potentially highly ambiguous artifacts. No reference need be made to the UML standard in diagrams conforming to any revision of the concrete syntax standard. Moreover, since diagrams only ever display projections of a model, a diagram in isolation is frequently not enough information to determine the true intent of the author. For example, attributes, associations and classes may all be suppressed in class diagrams. To eliminate this ambiguity, diagrams should at minimum refer to a concrete representation of the model that they represent. This concrete

instance could then identify information about the language being employed, such as the revision of UML being used, and any profiles being employed.

3. SPECIFICATIONS

3.1 Approach

In the preceding sections we have identified two major features of existing MDA standards that contribute towards ambiguity in concrete artifacts:

1. Current revisions of concrete standards do not require unambiguous references to themselves in concrete artifacts. If they are generic, they do not require reference to the language specification being employed. If they permit such a reference to be made, it is to a concrete artifact, such as a meta-model, and not to the actual language specification.

2. Although a language, MOF, exists for defining the abstract syntaxes of MDA languages, languages are typically not defined using a concrete MOF model, but using a description of one in English. English is also required to define the semantics of the languages, possibly with the assistance of intermediate semantic models.

To interpret a concrete artifact, it is necessary to understand its concrete and abstract syntaxes, and its semantics. Faced with a concrete artifact, a human interpreter should not have to guess what concrete syntax is being employed, in order to establish a basis for an initial interpretation of the artifact. Therefore all concrete artifacts should include a human readable comment referencing the specification of their concrete syntax.

Generic concrete syntax standards, applicable to multiple abstract syntaxes, can be adequately defined in current human-readable standards documents. Specialised concrete syntaxes may be documented with other aspects of the language to which they apply. In either case, having established the concrete syntax of an artifact, the abstract syntax and semantics of the artifact must then be identified in order for a complete interpretation. The concrete syntax standard should permit the artifact to be interpreted sufficiently that unambiguous links to specifications of syntax and semantics can be followed if they are separate from the concrete syntax standard.

Clearly there are a number of different possible schemes whereby abstract syntax and semantics can be referenced. The meta-model of the language and the specification document for the language could both be referenced. The meta-model could be referenced, and contain a reference to the specification, or vice-versa. Alternatively, because the specification documents both syntax and semantics, the specification alone could be referenced.

We prefer the final approach because a language specification provides the definitive documentation for a language. Hence we argue that specifications, not meta-models should be regarded as first-class entities in MDA developments.

Abandoning the use of meta-models as a language definition is potentially problematic, because existing language specifications cannot be machine interpreted in a useful way. For example, the descriptions of the abstract syntax include in the UML specification cannot be used to populate the meta-layer of a JMI repository. We address this issue by proposing that the use of MOF and natural language be inverted. Instead of describing a MOF model using natural language in a specification, a specification should consist of a concrete representation of a MOF model with natural language descriptions embedded, to provide an informal commentary on the aspects of the language that are defined technically, and to define the semantics where a technical language is inadequate. In this respect we propose that the MOF specification be revised to define a language which is somewhat similar to a programming language used for literate programming, in which technical and human readable aspects provide mutual support [8].

3.2 Revisions to standards

Our proposals are captured by the following particular revisions to existing OMG standards:

1. The MOF 2 standard already allows the inclusion of comments in meta-models. The Comment class should be extended with an attribute that indicates whether the comment is intended to be definitive of the semantics of the associated model element, or is merely an informal remark. A constraint should be added to the specification that all types, associations and references should be associated with a non-empty definitive comment that defines the semantics of the element in a manner that can ultimately be understood by a human interpreter.

2. Classes in the MOF standard should be extended with a `repositoryType` boolean attribute. This attribute supports the inclusion of semantic types in specifications by indicating whether the class specifies a datatype normally intended to be included in a repository and concrete syntaxes for the language, or whether it is a real-world type intended to disambiguate the language.

3. The MOF standard should be redistributed as an XMI concrete specification, with documentation included.

4. Future revisions of all OMG standards that rely on MOF meta-models should have a definitive form published as a concrete specification, for example in XMI form, rather than the definitive version being a PDF.

5. The XMI standard should be revised to incorporate a reference to the specification of the language used. All XMI files should be required to include an XML comment in natural language identifying the location of the XMI specification so that this link can be interpreted.

6. The HUTN standard should be revised, making it mandatory to include syntax configuration information in instance documents if a syntax configuration is being used, and also making it mandatory to refer to the specification of the language being used. All HUTN documents should be required to include a comment in natural language identifying the location of the HUTN standard so that this link can be interpreted.

7. Future concrete-syntax standards should respect the principle that instances should refer both to the concrete-syntax standard, and to the concrete specification of

the language being used (if separate from the concrete-syntax standard as is the case with XMI and HUTN). The reference to the concrete syntax standard must be human readable.

8. Diagrams should unambiguously reference a concrete representation of the complete model that they depict.

3.3 Discussion

The contents of definitive comments associated with elements should permit the interpretation of the element by a user. It may do this by introducing some intermediate formalism, encoded in the comment text in some manner. It may also unambiguously reference external documentation. However, the semantics should ultimately be interpretable by a human.

We do not prescribe any particular scheme by which unambiguous references should be made between concrete artifacts. However, URIs would be an appropriate choice [6].

The need to document MOF 2 according to its own standards assumes a new significance in this scheme. For a language specification to be understood the meta-language used to document it must also be understood. In this context we assume that that language will be a revision of MOF. The revised MOF specification will be machine readable, and recursively defined, which might seem to make it harder to understand. However, with reference to the XMI specification, which will still be human readable, and the embedded comments, the specification will nevertheless ultimately be decipherable by a human.

Our recommendation to add a `repositoryType` attribute is intended to support semantic definition styles that benefit from refinement of the syntactic meta-model of the language. The model-denotational style of semantic definition is the prime example of such an approach.

In practice these modifications would be relatively painless to implement. Existing specifications could be revised into conforming concrete specifications by editing the existing specifications into comments in an XMI file of the meta-model, simultaneously identifying those parts of the documentation that are definitive, and those which are informal. XMI concrete artifacts can be commented using XML comments to identify the XMI concrete syntax specification.

Existing language specifications, such as UML and MOF, include OCL and diagrams in addition to natural language statements and descriptions of meta-models. It is desirable to include this information in our machine-readable specifications. OCL statements form part of the definitive specification of the abstract syntax. They also have a semantic character, in that they rule out instantiations of the meta-model that would be illogical given the language semantics. If a model-denotational approach is taken, OCL constraints may also contribute to the semantic definition. Diagrams are typically informal and assist with the explanation of the language and its semantics.

Three approaches can be taken to including this information in concrete specifications. The OCL and diagrams could be included in documentation elements in an unstructured manner. Existing mechanisms for including this information could be employed: CMOF supports the inclusion of constraints with a constraint meta-element; XMI 2.1 supports the inclusion of diagram information, or other extension elements using an extension element, or XML namespaces.

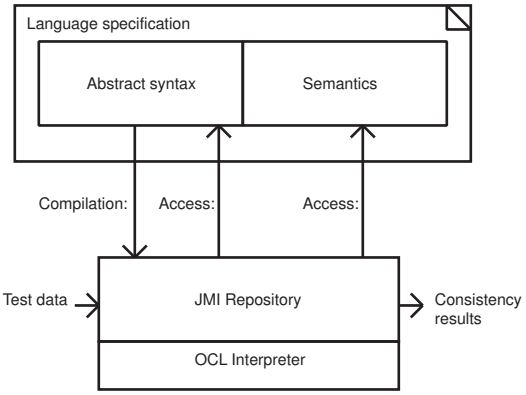

Figure 1: A specification used as input to a JMI generator. Abstract syntax and semantic documentation are available to the repository user via reflection

Finally, the information could be incorporated by extending the MOF specification with the OCL and diagram interchange specifications. Future work will consider trade-offs between these approaches.

The proposals we have made address the issue of capturing developer intent by ensuring that in any context in which the meta-model would otherwise have been employed, for example to populate the meta-layer model in a JMI repository, the semantic definition of the language is available. Additionally the availability of OCL and diagram types ensures that the semantic documentation is available in as machine-readable a form as possible, maximizing the potential for using this information intelligently in tools. As a first measure, presenting this information to developers in the same context in which they are using a novel language should assist in ensuring that they use the language correctly. The use of a specification in this manner is illustrated in Figure 1.

The proposals address the issue of recovering developer intent from an artifact by ensuring that artifacts always refer to both any relevant concrete syntax standard employed, and to the concrete specification of the language being used. The concrete syntax is referenced in as unambiguous a manner possible, using a natural language statement included in the artifact. Understanding the concrete syntax permits the recovery of the concrete specification of the language, which is totally definitive of the language. The concrete specification is constructed on similar lines to the original artifact, so can ultimately be understood in the context of a higher-level language, or in the case of MOF, recursively in terms of itself, and whatever concrete syntax standard is being used to describe it. Figure 2 illustrates the process of interpreting a concrete artifact. References in the concrete artifact to concrete and abstract syntax documentation provide a basis for interpretation of the document, traversing up meta-layers until a well-known standard such as MOF is encountered.

Our proposals, if implemented, would possibly have positive consequences for the quality of OMG standards. According to our proposals, specifications may contain both definitive and non-definitive documentation. They may also contain both repository types that are part of the abstract syntax of the language being specified, and other types for

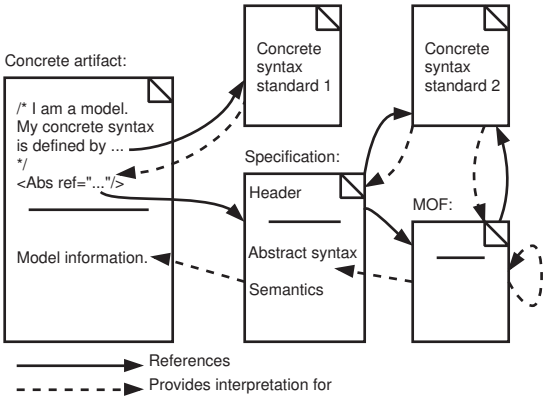

Figure 2: Recovering the meaning of an artefact by navigating links to concrete syntax standards and language specifications

the purpose of semantic exposition.

Non-definitive comments included in a specification remark on aspects of the specification already inherent in some definitive part. For example, a human-readable explanation is often given for OCL constraints included in a specification. However, the meaning of the OCL constraint is completely defined by the OCL standard, so this explanation does not refine the specification in any way.

However, non-definitive elements should not be misleading. Ideally the non-definitive parts of a specification should be entirely consistent with the definitive parts. Meta-model constraints can be checked to guarantee this to some extend, for example checking that a diagram is consistent with the underlying model. Checking of this kind has not been scrupulously performed in prior OMG standards. In [4] for example, a significant number of syntactic and type errors are discovered in the OCL constraints included in the latest version of UML. In some cases testing will also be necessary. The machine-readable nature of specifications would facilitate this to a great extent. For example, the specification could be used to implement a JMI repository. The consistency of a test suite of instance documents could then be checked to ensure that OCL constraints act consistently with the semantic intent documented in their accompanying non-definitive descriptions. Thus far, this kind of testing has never been performed for an OMG standard.

Testing of semantic elements in specifications is also possible. Although these would not normally be included in a repository, this constraint can be relaxed to allow testing of these elements in the same manner as the syntactic elements.

4. THE UCL MDA TOOLS

The UCL MDA tools are an open-source project intended to provide a set of command-line tools for model-driven development [3].

The UCL MDA tools project currently provides:

- JMI interfaces and implementation for the EMOF metamodel, including XMI serialisation and deserialisation.
- A JMI generator, proceeding from EMOF XMI models.
- A parser for a textual concrete syntax of EMOF, in which definitive comments may be embedded in a literate programming style.
- A parser and type checker for OCL2 expressions in the context of EMOF models.
- An OCL2 interpreter, including an implementation of the OCL2 standard library, integrated with the generated JMI repository, for the evaluation of consistency constraints embedded in EMOF meta-models.
- A converter from EMOF/OCL pages to HTML documentation, HTMLEMOFDoc.

These tools implement the recommendations provided in the preceding section. Atextual concrete syntax of EMOF embeds definitive comments in a special syntax similar to JavaDoc comments for Java, which leads them to be associated with elements in the resulting model. We have also explicitly combined the EMOF and OCL meta-models in the meta-modelling tool. OCL constraints can hence be embedded in the specification also. The whole specification can be parsed to an XMI representation (including XMI for the OCL constraints, and the embedded comments). Syntax checking is performed according to the OCL2 specification, and our own syntax for textual EMOF specifications. Type checking of all elements is performed according to the semantics of EMOF and OCL2.

One application of the specification is to reformat it into a format more suitable for human comprehension. We have hence developed a tool, similar to the JavaDoc tool that generates a webpage from our EMOF/OCL/English specification of SLAng. The structure of the page resembles the structure of an OMG language specification, with each element in the meta-model presented along with its documentation. The page also benefits from hyper-links that cross-reference related elements.

5. CASE STUDY

An SLA, as a concrete artifact, is the record that an agreement exists with respect to service levels. If at some time after the agreement has been made it is impossible to determine the meaning of a concrete SLA, then it is possible for a party to assert that no agreement existed, and hence abdicate their responsibilities with respect to the agreement. Concrete SLAs should therefore either unambiguously define their own meaning (through the use of carefully written natural language statements, for example) or should unambiguously reference a document allowing their meaning to be interpreted.

We chose to adopt EMOF, OCL and English as the languages of definition for SLAng, a language for electronic service SLAs, and we structured the specification using a model denotational approach. A previous discussion of this approach applied to an SLA language is provided in [11]. The full EMOF/OCL definition of SLAng is currently available online as EMOFHTMLDoc documentation [2]. Detailed design rationale for the language will be the subject of a future publication. An example fragment of the definition of SLAng is shown below. The fragment defines a percentage datatype used in various constraints in the language. Note the inclusion of definitive semantic documentation in

a special comment syntax, which are retained in all representations of the specification, including an XMI version that may be compiled from the EMOF/OCL notation.

```
/[
Definitive: In the syntactic model, indicates a percentage
written in an SLA.  In the semantic model, this is the type
of features of an object that can be interpreted as a degree
of completeness of some totality.
]/
class Percentage {

    /[
    Definitive:  The percentage is this value multiplied by 100.
    ]/
    value : Real

    /[
    Wellformedness:  Percentages are expressed as a value greater
    than 0.
    ]/
    invariant {

        value >= 0
    }
}
```

A large portion of the meaning of a SLAng SLA is established in the definition of the SLAng language. It is therefore necessary for an SLA in a concrete syntax for SLAng to refer to its specification document to retain its significance as the record of an agreement. XMI, with the extensions described in the previous section is currently the only concrete syntax for SLAng. An example fragment of XMI for a SLAng SLA is shown below. The XMI includes an instance of the percentage datatype defined above. Also visible is the XMI header, which identifies the concrete syntax of the document and references the SLAng specification XMI. Other contents have been omitted.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--This document is in XMI format according to the OMG
XML Metadata Interchange (XMI) Specification v.1.2, OMG
Document formal/02-01-01 (http://www.omg.org/)-->
<XMI
  xmlns:SLAng=
    "http://www.cs.ucl.ac.uk/staff/j.skene/slang/SLAngV1.xmi"
  xmi.version="1.2">
<XMI.header>
<XMI.metamodel
  href=
    "http://www.cs.ucl.ac.uk/staff/j.skene/slang/SLAngV1.xmi"/>
</XMI.header>
<XMI.content>
  <SLAng:Percentage value="0.5" xmi.id="mofid:25717"/>
  ...
</XMI.content>
</XMI>
```

6. SUMMARY

In this paper we have outlined a set of minor modifications to OMG standards, particularly language specifications and concrete syntax standards, that help to address two problems in model driven development, namely ensuring that a developer's intentions are accurately captured when working with several possibly unfamiliar languages, and ensuring that those intentions can be faithfully recovered from the resulting artifact by other developers. The modifications ensure that a link is preserved between artifacts and the definition of both the syntax and semantics of the language in which they are defined.

A consequence of these changes, if they are adopted, will be to transform the specifications upon which the MDA relies into technical artifacts. More than was previously the case, there will be the opportunity for the OMG to test and thereby guarantee the quality of their published specifications. In particular it will be possible and desirable to test coherence between informal commentary concerning the rationale behind any specification and technical aspects of the specification, such as OCL constraints.

We have also introduced an open-source project, the UCL MDA tools, which implements the tooling required to support the changes that we have proposed. We described a novel domain-specific language developed using the tools. The language is for SLAs, the legalistic nature of which provided further justification for the common requirement to maintain an explicit link between artifacts and the language in which they are defined.

7. REFERENCES

[1] The Object Management Group (OMG). `http://www.omg.org/`.

[2] The SLAng language specification v. 0.1, 2005. `http://www.cs.ucl.ac.uk/staff/j.skene/slang/spec.html`.

[3] The UCL MDA Tools, 2006. `http://uclmda.sourceforge.net/`.

[4] H. Bauerdick, M. Gogolla, and F. Gutsche. Detecting OCL traps in the UML 2.0 superstructure: An experience report. In *UML 2004*, number 3273 in Lecture Notes in Computer Science (LNCS), pages 188–196. Springer-Verlag, 2004.

[5] A. S. Evans and S. Kent. Meta-modelling semantics of UML: the pUML approach. In *2nd International Conference on the Unified Modeling Language*, volume 1723 of *Lecture Notes in Computer Science (LNCS)*, pages 140 – 155, Colorado, USA, 1999. Springer-Verlag.

[6] The Internet Society. *Uniform Resource Identifier (URI): Generic Syntax*, rfc: 3986 edition, 2005. `http://www.ietf.org/rfc/rfc3986.txt`.

[7] Java Community Process. *Java(TM) Metadata Interface API Specification 1.0 Final Release*, June 2002.

[8] D. E. Knuth. Literate programming. In *The Computer Journal*, volume 2, pages 97 – 111. 27 edition, May 1984.

[9] The Object Management Group (OMG). *MDA Guide Version 1.0.1*, omg/2003-06-01 edition, June 2003.

[10] M. Richters and M. Gogolla. On formalizing the UML Object Constraint Language (OCL). In *17th International Conference on Conceptual Modelling (ER'98)*, volume 1507 of *Lecture Notes in Computer Science (LNCS)*, pages 449 – 464. Springer-Verlag, 1998.

[11] J. Skene and W. Emmerich. Precise service level agreements. In *26th International Conference on Software Engineering (ICSE)*, Edinburgh, UK, May 2004. IEEE Press.

[12] The World Wide Web Consortium (W3C). *Scalable Vector Graphics (SVG) 1.1 Specification*, January 2003. `http://www.w3.org/TR/SVG/`.