Specification of Modelling Languages in a Flexible Meta-model Architecture

Liping Mu  
Faculty of Engineering and Science  
University of Agder  
Serviceboks 509, NO-4898  
Grimstad, Norway  
liping.mu@uia.no

Terje Gjøsåter  
Faculty of Engineering and Science  
University of Agder  
Serviceboks 509, NO-4898  
Grimstad, Norway  
terje.gjosater@uia.no

Andreas Prinz  
Faculty of Engineering and Science  
University of Agder  
Serviceboks 509, NO-4898  
Grimstad, Norway  
andreas.prinz@uia.no

Merete Skjelten Tveit  
Faculty of Engineering and Science  
University of Agder  
Serviceboks 509, NO-4898  
Grimstad, Norway  
merete.s.tveit@uia.no

ABSTRACT

Originally, meta-models were used to specify the structure (abstract syntax) of modelling languages. This is reflected both in meta-languages like MOF and Ecore, and the four-layer meta-model architecture. Presently, meta-modelling is used for specification of complete languages. In this situation, it turns out that the traditional meta-languages are not always expressive enough to capture all language aspects. This usually implies the use of more than one meta-language in the meta-model architecture to cover the different language aspects. There are many approaches to address this challenge. In this paper, we analyze these approaches, and based on this analysis, we re-think the meta-model architecture focusing on complete language specifications. In our meta-model architecture, each aspect of a language conforms to an aspect-specific meta-language at the level above, and models can reside at different levels depending on their context and use. This meta-model architecture is easier to understand, more flexible and more extensible; therefore it may be useful in the design of meta-model-based language specification platforms, as well as for promoting the understanding of the principles of meta-modelling.

Categories and Subject Descriptors
D.3.1 [Programming Languages]: Formal Definitions and Theory

General Terms
Theory, Languages, Design

Keywords
Meta-model architecture, MDA

1. INTRODUCTION

As information and applications with high complexity spread into all areas of daily life, traditional methods of software production and data handling can not cope with this increasing complexity. New ways of complexity handling use higher levels of abstraction and describe systems with models. In particular, the Object Management Group (OMG) puts forward their idea of a model-driven architecture (MDA) [15] which focuses on software development by means of high-level models. To take full advantage of a model-driven development process, it is necessary to have appropriate modelling languages and tool support for these languages. There are basically two ways to build appropriate tool support, either by writing a language standard and manually building tools, or by having higher-level meta-tools that allow to automatically generate tools from a description of a language. In any case it is necessary to have a description of the language first. A complete language description consists mainly of structure, presentation and behaviour [13].

Specification of the structure of a language by means of meta-models is well-known, and there are standard meta-languages like MOF [14] for this purpose. MOF is at the upper-most level in the OMG four-layer architecture, where a meta-model at one level is used to define models at the level below. Or said in another way, a model at one level conforms to a meta-model at the level above [2].

In the OMG four-layer architecture, every model element on each layer is strictly an instance of a model element of the layer above. This architecture and MOF are mainly focused on the structure or abstract syntax of a language. However, a complicating factor is that languages have different...
aspects. Therefore, a question arises: How can a complete meta-model-based description of a language, including the presentation and behaviour, be supported by this traditional four-layer architecture? In this paper, we present some different meta-model-based approaches, and place them and their meta-languages in the four-layer architecture. We see that if we consider the meta-levels more relative to each other, it will give more possibilities to specify all the aspects of a language in a uniform and consistent way. This idea of seeing the levels as relative to each other is also emphasised by Favre in [6].

This paper is organised as follows. Section 2 introduces the term meta-model architecture. Section 3 starts with a presentation of the different aspects of a language specification and describes how a language specification is conceptually placed within a meta-model architecture. Section 4 gives an overview of existing meta-model-based approaches and how they use the meta-model architecture to specify languages. In Section 5 we present an evaluation based on the different solutions. The main aim of this section is to show how different views on the meta-model architecture have consequences for how a language is specified. In Section 6, we describe our approach at the University of Agder modelling lab. Finally, Section 7 summarises and concludes the paper.

2. THE META-MODEL ARCHITECTURE

Generally speaking, a model is a simplified abstract view of the complex reality. A meta-model is a model of models, or a model that defines the language for expressing a model. Meta-modelling refers to the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for modelling a predefined class of problems, and mostly deals with the construction of better ways to specify modelling languages. It is conventionally used for three major purposes: generic language definition, domain-specific modelling, and model interchange [10]. A meta-model architecture or a meta-model hierarchy is a tree of models that are connected by relationships, like ‘instance-of’ or ‘conforms-to’ relationships, and the most well-known example is the OMG four-layer architecture.

This traditional OMG architecture is mainly focusing on language definition, which means the instance-of relationships that connect models are linguistic instantiations. It has been argued that domain definition should be also concerned within the meta-model architecture, in other words, ontological instantiation relationships are needed so as to fully satisfy the requirements of a model-driven development infrastructure [1]. However, in this paper, we are more interested in language definition and we use the OMG four-layer architecture, which is focusing on linguistic relationships, as our starting point. One might also argue that the term ‘meta-model’ is primarily referring to the structure or abstract syntax of a language. However, in the future, the term ‘meta-model’ might cover a complete language specification, such that it defines abstract syntax, concrete syntax and semantics, which is in line with the statement that meta-modelling should be able to describe complete languages. Hence, in our case, we do not consider a meta-model only as the definition of abstract syntax of a language but the complete specification of a language. In the next section we will introduce what is a complete specification of a language and how it is placed in the meta-model architecture.

3. LANGUAGE DESCRIPTIONS IN THE META-MODEL ARCHITECTURE

A description of a modelling language, whether it is a domain specific language (DSL) or a general purpose language, usually involves several different technologies and meta-languages. Traditionally, we are familiar with the distinction between the syntax and the semantics of a language. The syntax specifies the structure of sentences in the language, while the semantics assign a meaning to the sentences. A complete description of a modelling language can be seen as consisting of structure, presentation and behaviour as illustrated in Figure 1.

![Figure 1: The aspects of a modelling language.](image)

**Structure** defines the constructs of a language and how they are related. This is also known as abstract syntax. Constraints (roughly corresponding to static semantics) describe additional restrictions on the structure of the language, beyond what is feasible to express in the structure itself.

**Presentation** defines how instances of the language are presented. This can be the description of a graphical, textual, form-based, matrix-based, table-based, map-based and other types of concrete language syntax. Here, we take graphical and textual presentation as examples.

**Behaviour** explains the dynamic semantics of the language. This can be a mapping or a transformation into another language (translational semantics, denotational semantics), or it defines the execution of language instances (operational semantics).

As Figure 1 shows, the structure, including constraints, is the central aspect in a language description, and all the language aspects are defined completely independently with clear boundaries between them.

It is well-known to use meta-models to specify the structure of a language, like the existing meta-languages MOF and Ecore [21], but they are not expressive enough to handle language aspects like presentation and behaviour.

If we take a look at the meta-model architecture presented in Figure 2, we can see a state machine diagram written in the UML language on the left-hand side. The state machine diagram resides at the M1 level in the architecture, while the UML language itself resides one level above at the M2 level. The right-hand side of the architecture illustrates which aspects the description of the diagram and the definition of the language consists of. All the meta-models defined at the M2 level are written in a special language, residing...
Figure 2: A state machine diagram written in the UML language and their descriptions in the meta-model architecture.

at M3, that is tailored to describe a particular language aspect. This kind of language is called a meta-language, and in a meta-model-based approach, a meta-language is described in terms of a meta-meta-model. When we see the meta-model architecture like this to define the complete languages, it is not necessarily enough with a meta-language for structure only. One possible solution would be to have a set of meta-languages that together covers all the language aspects and is self-defining. That means each aspect of a meta-language in the set can be defined either by itself or by one of the other meta-languages in the set.

The following section starts by giving more details on some relevant OMG standards for language specification, and in addition presents the approaches in the frameworks XMF, KM3, Kermeta and EMF/GMF, related to the meta-level architecture.

4. APPROACHES TO THE META-MODEL ARCHITECTURE

4.1 The OMG approach

MOF is the central part of the OMG set of technologies. It is defined by re-using and extending the UML Infrastructure [19]. The OMG is adding technologies to fill in the gaps where MOF is insufficient to cover all aspects of a language. They have defined OCL for constraints, QVT for transformation behaviour, and there is also a Diagram Definition Request For Proposal for graphical presentation.

The Object Constraint Language (OCL [17]) is a formal language for describing logical constraints on models and meta-models. It is part of the UML 2.0 standard, but OCL constraints can also be applied to any MOF-based model. Since OCL is part of the UML language, it resides at the M2 level.

OMG has developed the MOF Query/View/Transform (QVT) standard [16] to define transformations between MOF-based models. We note that not only is the structure of QVT defined using MOF, but OMG also uses QVT itself for defining transformation behaviour. However, execution behaviour is defined by a combination of English and first order predicate logic. OMG does not in the specification explicitly place QVT at a certain level in the meta-model architecture. Usually in OMG specifications, MOF is alone on M3. However, being named MOF Query/View/Transform, may be a clue that QVT can be seen as residing on M3 together with MOF.

Currently, OMG has announced a Request for Proposals for Diagram Definition [20]. The Diagram Definition will be a language that will supply means to formally define and exchange the presentation of diagrams of a language, as well as to define the mapping between the presentation and the structure. There already exists an OMG standard for exchanging diagram presentation, the UML Diagram Interchange (UML DI) [18], but this standard is not precise enough to define the presentation of a diagram. The UML DI resides at the M2 level as it extends the UML meta-model. It is a requirement from OMG that the new Diagram Definition language is defined using MOF and that it is making use of UML-DI. This indicates that the Diagram Definition language also resides at the M2 level.

Even if it is unclear which level the supplementary languages like OCL, QVT and Diagram Definition reside at, we see that OMG brings us closer to a complete standard toolkit for defining all aspects of a language.

4.2 The XMF approach

XMF-Mosaic from Ceteva [4] is a stand-alone platform, recently connected to Eclipse, for building tailored tools that should provide high-level automation, modelling and programming support for specific development processes, languages and application domains.

One of the most important features of XMF is that it is capable of describing itself. That is, it provides several meta-languages that together describe themselves, and thereby also XMF. The central language in XMF is XCORE. XCORE is inspired by MOF and is reflexive, and the structure of a language is specified using the concepts defined in XCORE itself. To be able to describe all the aspects of a language, XCORE is extended by the meta-languages XOCL and XBNF, which are used to specify actions and textual presentation, respectively. XOCL is in turn extended by XMap and XSync. These two meta-languages are used to describe mapping relationships between, and synchronizations of, models. Figure 3 illustrates the inheritance relationship between the meta-languages.

Figure 3: The architecture of XMF.
This approach is close to supplying a complete set of mutually self-defining meta-languages at the M3 level in the architecture. The only aspect that is not covered at M3, is the graphical presentation. For this aspect, XMF provides the language XTools which defines the diagram meta-model. Instead of specifying the graphical presentation for a given language, the diagram meta-model is inspired by the UML DI and gives a general abstraction of all diagrams. This language-independent diagram meta-model is used to extend a structure meta-model that conforms to XCORE at the M2 level.

4.3 The KM3 approach

KM3, the Kernel MetaMetaModel, has been defined by the ATLAS INRIA team [11]. KM3 is a textual meta-language for defining meta-models, and can be used to define the structure of DSLs. The structure of KM3 itself is defined in KM3, and it also provides semantics for meta-model descriptions.

The KM3 approach uses a meta-architecture with absolute levels, and KM3 is the only meta-language at the M3 level. The other aspects of a language, like presentation and behaviour, are covered by languages that are defined at the M2 level. The structure of the transformation language ATL is one example of a language that is specified by KM3 [3]. As Figure 4 illustrates, ATL is at the level below KM3, which means a transformation \( T_{AB} \) between structure meta-models \( (MM_A, MM_B) \) will be at a level below the meta-models themselves. TCS (Textual Concrete Syntax, [12]) is a DSL for specification of textual presentation that conforms to KM3. We see in Figure 4 that using this approach, KM3 and TCS are not compatible with the conceptual architecture presented in Section 3. Instead of having a specification of all possible textual sentences that are well-formed according to the language at level M2, we here get a DSL that gives a language independent description of textual presentation. This leads to a situation where we have the structure \( (MM_A) \) and textual presentation \( (CS_A) \) of a language at different levels in the meta-model architecture.

4.4 The Kermeta approach

Kermeta [5] is a modeling and aspect-oriented programming language, defined by the Triskell team of IRISA. Its underlying meta-model conforms to the EMOF standard. It is designed to write programs which are also models, to write transformations of models, to write constraints on these models, and to execute them. Kermeta has been designed to be fully compatible with EMOF. The meta-model of Kermeta is divided into two packages: EMOF for defining structure, and Actions for defining behaviour.

In the Kermeta approach, the structure of the Kermeta language, Executable EMOF, is defined by EMOF, and the execution behaviour (operational semantics) corresponding to MOF concepts is defined in Kermeta Actions.

Figure 5 shows how Executable MOF is defined by MOF and then promoted from M2 to M3. Thereby, we have Executable EMOF on M3, which can define both the structure and execution behaviour of a language. This promotion is a good illustration of flexibility in a meta-model architecture.

![Figure 5: Kermeta in the meta-model architecture (from [5]).](image-url)
5. EVALUATION

In a language description that is based on the thought that all aspects of the language are covered by one meta-model, then only one meta-language is necessary. This meta-language will provide all the necessary features to specify all the aspects of a language. There are several ways this can be handled in practice; one way is to define the other language aspects, like presentation, as general languages at the M2 level. The structure is then specialising the concepts in the general language for graphical presentation. Figure 7 gives an illustration of how this would look like in the meta-model architecture.

With this conceptual perspective, the graphical presentation is not specified for one specific language at the M2 level like the structure description, but only gives a language(structure) independent abstraction of what a diagram is. The language can be seen as a language for describing diagrams, not only for specific language like the meta-model for structure, but all possible diagrams. The metatool XMF-Mosaic and the OMG standard UML DI are examples that are based on this perspective, see Sections 4.1 and 4.2. For textual presentations, the KM3-based language TCS, presented in Section 4.3 plays a similar role in the architecture as it resides at the M2 level.

For behaviour, there seems to be a common approach of using some language independent action language (e.g. ASM) for execution behaviour, or transformation language (QVT, ATL) to define transformation behaviour. These languages usually reside at the M2 level, conforming to MOF, KM3 or Ecore on M3; a notable exception being Kermeta Actions that are promoted to M3 and reside there together with EMOF. With generic action- or transformation languages on M2, the transformations and execution specifications will be located on M1 but will apply to meta-models on M2.

In our opinion, the structure, the behaviour, and the presentation of a language should be specified as separate language specific aspects at the M2 level. The structure will then specify all possible internal models in the language, the presentation may specify all possible diagrams in the language, and the behaviour may specify transformations of all possible language instances to a given language, or all possible runtime states and state transitions. A mapping description will keep the structure synchronised with presentation and behaviour. Then we need meta-languages to provide all the necessary features to specify all the aspects of a language at the M3 level. Figure 8 illustrates how this approach resides in the meta-model architecture. The Eclipse-based meta-tools Kermeta and GMF are approaches that adapt this architecture, see Sections 4.4 and 4.5. We list some advantages of this architecture:

1. Clear:
   - Different meta-models represent different aspects of a language instead of having one mixed meta-model.

2. Flexible:
   - One structure could have two or more presentations. For example, The SDL language has both a graphical and a textual concrete syntax, that both map to the same structure, as shown in Figure 9.
   - One presentation could be mapped to different (but similar) structures. For example, it would be possible to map a UML state machine presentation to either UML structure or SDL structure depending on the need, as shown in Figure 9.
   - Separating presentation and structure allows for easier handling of the situation where there is not a one-to-one relation between elements of structure and presentation. For example, in a SDL state machine, state A is one element in the structure, and the same state can be represented by two or more different diagram elements in different locations.

3. Extensible:
   - It is easier to change a language definition or define new languages.
6. OUR APPROACH

The approach within the modelling lab at the University of Agder, is based on our opinion stated in the previous section that all aspects of a language should be defined specifically by using suitable meta-languages on the level above, and at the same time, the need to use commonly available tools such as Eclipse with relevant plug-ins.

In our approach, we start from the M2 level, focusing on what is necessary for specifying a language. The following example is describing the approach for a language with a graphical presentation. The M2 level is where the description of the language resides, that is a description of all possible structures and a description of all possible diagrams in that particular language. The description of all the possible diagrams in the language must conform to a meta-language at M3. If meta-languages for the presentation and behaviour reside at the M3 level, then MOF could be the meta-language for these languages at the M4 level, giving the following meta-model architecture:

- **M4**: MOF or similar language.
- **M3**: The meta-languages for structure, presentation and behaviour.
- **M2**: Complete definition of all the aspects of a language.
- **M1**: Descriptions of all the aspects of a given diagram (or textual language instance).

Having meta-languages that cover all aspects of languages at the above level in the architecture brings us also closer to the idea of having a set of mutually self-descriptive meta-languages. We see meta-languages as offering interfaces that languages on the level below can use. For example, the meta-language OCL offers constraints, that can be used by itself, by other meta-languages, languages, and language instances (see Figure 10).

However, it should be noted that definition of a language is not the same as usage of a language; for example, if we follow the common notion that MOF is used at M3, and languages are defined at M2, we have to also consider MOF to be defined at M2. The same is true for other meta-languages like OCL and QVT. This separation between definition and usage will remove the need for having more than three levels.

When that is said, the main issue in this discussion is not which level a model resides at. What is more important is to know what resides at the level above, or more precisely, to know what meta-languages are used at the level above when defining a complete language below. Hence, a more relative architecture could be defined like this:

- **N-1**: The meta-languages for structure, presentation and behaviour. We may either add another level, N, on top of this, or close the top of the hierarchy by having the languages on this level to be defined in terms of themselves; i.e. level N is the same as N-1.
- **N-2**: Complete definition of all the aspects of a language.
- **N-3**: Descriptions of all the aspects of a given diagram.

Thereby, we apply the notion that models can freely be promoted or denoted between levels depending on the intended use, and models in the meta-model architecture are relative to each other based on the relationship between them.

7. SUMMARY

In this paper we have given an introduction to the meta-model architecture, and examined various approaches to using the meta-model architecture for language description,
with a focus on how the different approaches handle the different language aspects such as structure, presentation and behaviour.

Based on an evaluation of some approaches for defining complete languages in the traditional meta-model architecture, we have proposed our own approach. In this approach, M3 contains meta-languages for structure, presentation and behaviour, M2 contains complete definitions of all aspects of a language, and M1 contains a complete description of the diagram or language instance. However, we suggest to use a relative numbering scheme instead of M1-M3 to emphasise that the absolute location of a model is not important, but rather the connection to the levels directly above (meta-model of the model) and below (the instances of the model). Each aspect of a language conforms to an aspect-specific meta-language at the level above, and models can reside at different levels depending on their context. Thus, models can freely be promoted or demoted between levels depending on the intended use. On the top level, instead of just MOF or similar language, we suggest to have a set of mutually self-defining meta-languages covering all aspects of the languages below.

The clarity, flexibility and extensibility of this approach should make it useful in the design of meta-model-based language specification platforms, as well as for promoting the understanding of the principles of meta-modelling.

8. REFERENCES