A Taxonomy of Metamodel Hierarchies

Ralf Gitzel, Tobias Hildenbrand
Department of Information Systems
University of Mannheim, Schloss
D-68131 Mannheim, Germany
Tel.: +49 621 181 1642
email: gitzel@wifo3.uni-mannheim.de, hildenbrand@uni.mannheim.de

A Taxonomy of Metamodel Hierarchies

Ralf Gitzel
University of Mannheim
Department of Information Systems III
L5,5 D-68131 Mannheim
+49(0)621 181 1645
gitzel@wifo3.uni-mannheim.de

Tobias Hildenbrand
University of Mannheim
Department of Information Systems I
L5,6 D-68131 Mannheim
+49(0)621 181 1670
hildenbrand@uni-mannheim.de

ABSTRACT
In the context of software engineering and model-driven development in particular, metamodeling gains more and more importance. So far, no classifying study of theoretical metamodeling concepts and hierarchy design options has been conducted in order to establish a comprehensive set of interrelated design variables, i.e. a coherent design space. A well-designed metamodeling hierarchy is essential to avoid problems not easily noticeable, like ambiguous classification and the replication of concepts. This study aims at exploring the theoretical foundation and providing a taxonomy or a design space for constructing tailor-made metamodel hierarchies for specific problems areas and domains.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *computer-aided software engineering, object-oriented design methods, software libraries.*

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

Keywords
METAMODELING, METAMODEL HIERARCHIES, MODEL-DRIVEN DEVELOPMENT, SOFTWARE ENGINEERING, CASE, DOMAIN-DRIVEN DEVELOPMENT, ONTOLOGIES.

1. INTRODUCTION
Briefly speaking, metamodels are models which describe other models. Since this paper analyses variants of metamodeling, alternative definition techniques, such as graph grammars (cf. [13]), are not considered and therefore, the claim that every model has a metamodel, even if it is in many cases implicit (cf. Albin [1]), can be accepted as true within the scope of our analysis. Metamodeling today is conventionally used for three major purposes: generic language definition –as in the case of MOF and the UML, domain-specific modeling, and model interchange. In contrast to mere language definition, domain-specific modeling languages (DSL) can be provided in order to facilitate the software engineering process for domain-specific applications (cf. [20]). Model interchange between different tools on the other hand, is a classical application of metamodeling in the field of computer-aided software engineering. In this context metamodels serve as "data exchange formats" for models describing their syntax. Examples for metamodel interchange standards comprise the CASE Data Interchange Format (CDIF, cf. [10]) and XML Metadata Interchange (XMI, see [26]) which is developed in accordance with the MOF by the OMG.

A well-designed metamodeling hierarchy is essential to avoid many typical problems, such as ambiguous classification and the replication of concepts. This study aims at exploring the theoretical foundation in terms of terminology and providing a taxonomy of options for designing tailor-made metamodel hierarchies for specific problems areas and domains. These options and their impact on the quality of the hierarchy are measured against a group of carefully selected criteria, creating a design space for metamodel hierarchies. A **design space** in the context of software engineering is defined by Herbleb and Mockus (2003) as "the set of all possible assignments of the set of variables representing the engineering decisions that have to be made" [20]. So far, no such design space exists for metamodel hierarchies and hopefully this paper will facilitate the design of good metamodel hierarchies in the future.

To achieve all this, we will initially shed light on some major criteria for evaluating different metamodel architectures in terms of design choices (see section 2). In section 3 we will apply these criteria on a set of design options in order to build a taxonomy of metamodel hierarchies consisting of various design parameters and classification schemes. Hierarchy designs will be evaluated according to various interrelated design choices and problem domains. The conclusion in section 3.5 recapitulates our findings and outlines the context of our study in terms of ongoing and future research activities and projects.

Due to its limited scope, this paper assumes a basic knowledge about metamodelling such as described in various papers and tutorials (cf. Atkinson [8], Atkinson and Kühne [7], or Völter [31]) and does not further elaborate on these concepts.

2. CRITERIA FOR EVALUATING METAMODELING HIERARCHIES
Following the OMG's basic ideas of multilevel metamodeling [27] we use the following definition of a metamodel hierarchy as foundation for our research:

*A **metamodel hierarchy** is a tree of models connected by instance-of relationships. The term model layer or model level describes all (meta-)models with the same distance to the root metamodel in a metamodel hierarchy. Each level is given a unique name, often containing a number.*

In order to evaluate design choices and thus possible architectures for metamodeling hierarchies according to certain quality criteria typical goals of metamodel hierarchy designers need to be identified. In this respect we mainly focus on the **general**

applicability of a hierarchy and its **ease of use** at all levels of abstraction due to complexity and maintainability considerations.

After having established the relevant goals it is now possible to derive **quality criteria** for evaluating different design choices and "vectors" (architectures) in the design space examined. The list of criteria in this study includes:

1. Complexity,
2. Consistency,
3. Expressional Strength,
4. Extensibility, and
5. Robustness to Change.

Metamodel hierarchy **complexity** and **consistency** are largely affected by one core problem of metamodel hierarchies, namely **replication of concepts**, an issue first identified by Atkinson and Kühne who provide the following definition:

If a modeling element from any layer in a metamodel hierarchy is reproduced on a lower layer, the hierarchy contains a ***replication of concepts*** [7].

This should be avoided for two reasons. First of all, with additional elements, the model size is increased unnecessarily, making it harder to understand. Second, inconsistencies might be introduced, if the replicated concepts are accidentally implemented slightly different in each case (cf. Atkinson and Kühne [7]).

In the same context **ambiguous classification** was spotted as another critical challenge of multilevel metamodeling which also has a negative impact on the consistency criterion. Atkinson and Kühne provide the following definition:

A metamodel hierarchy is called ***ambiguously classified*** *or as suffering from* ***ambiguous classification****, if there exists an instance which has more than one type.* [7]

Given that Seidewitz states that "a single modeling language might have more than one metamodel, with each expressed in a different modeling language" [29], it seems that all hierarchies automatically suffer from this problem Obviously, each instance in the model must have a type in each of the metamodels and, therefore, it is ambiguously defined according to the definition provided above. On the other hand, the definition of ambiguous classification might be interpreted as referring to several types per metamodel, seeing the types in other metamodels as part of alternative model hierarchies. For the discussion in this paper, the latter point of view is adopted.

A third property indicating the quality of a metamodel hierarchy is its **expressional strength**. Although it appears to be at odds with model complexity, this is not necessarily the case, as we will see. Great expressional strength is required in order to allow the modeling of a wide range of domains and the presentation of information at different levels of abstraction.

Eventually, the degree of **extensibility** of a metamodel hierarchy also influences its applicability to a wide range of problem domains, i.e. the ability to add new modeling elements for unforeseen circumstances. A criterion closely related to extensibility is **robustness to change**, which reflects how much impact a genuine change as opposed to a mere extension has on the existing instances of the hierarchy. However, since extensibility is generally easier to achieve than robustness to arbitrary changes, the two criteria are examined separately.

3. DESIGN OPTIONS FOR OBJECT-ORIENTED METAMODEL HIERARCHIES

Based on the quality criteria established in the section 2, the following sections try to establish a comprehensive and coherent system of variables and design options for constructing suitable metamodel hierarchies for diverse domains.

3.1 Linear vs. Non-Linear Hierarchies

One of the fundamental decisions in metamodeling is whether to use a traditional linear metamodel hierarchy or a more advanced non-linear metamodel hierarchy. Many of the other decisions such as the number of layers or the nature of the top level depend on this choice.

According to Atkinson and Kühne ([3], [4]), two different kinds of *instance-of* relationships can be distinguished – linguistic and ontological ones. ***Linguistic instance-of relationships*** describe language definition type constructs, e.g., facts like "*Person is an instance of Class*" or "*Painter is an instance of Object*". ***Ontological instance-of relationships*** on the other hand describe domain specific facts, e.g., *Painter* is a logical instance of *Artist*. While Atkinson and Kühne have coined another pair of terms, i.e., physical and logical, which they use alternatively. The linguistic/ontological terminology will be used exclusively in this section, for reasons which will be discussed later. [5]. The idea to distinguish between different types of instance-of relationships has also been proposed by other authors, e.g. Bézivin and Lemesle [9] or Geisler et al. [14].

After having identified this dichotomy, the question arises how two different *instance-of* relationships can be combined in one coherent metamodel hierarchy. In a ***naïve approach*** (as shown in Figure 1), the elements in the highest layers of a model hierarchy would connect to their instances via linguistic relationships, defining a modeling language while the lower layers define the ontological relationships within the domain. This is in effect a ***linear hierarchy*** [5].

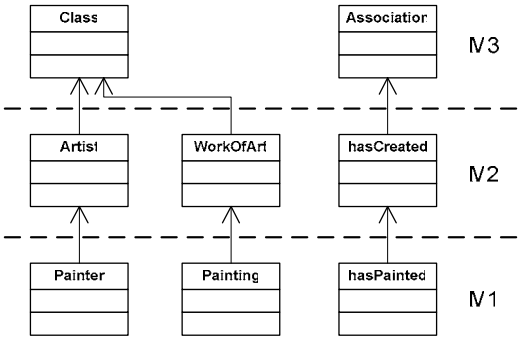

Figure 1 - Naïve Linear Hierarchy

To allow such an intuitively correct hierarchy, the instantiation semantics have to be carefully chosen. For example, the standard MOF instantiation semantics (cf. [27]) will not allow any further instantiation of *Painter*, as its metaclass is not *Class* or *Association* and there are no rules provided on how to instantiate model elements whose type is *Artist*.

Besides the solution already outlined in Gitzel and Schader [15], a preferable approach is to introduce orthogonality into the model layers, effectively creating a ***nonlinear framework*** [5]. Figure 2 recalls the example given above, this time using a nonlinear framework.

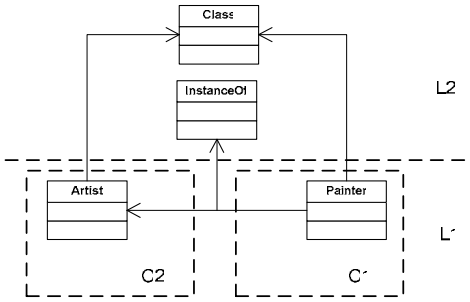

Figure 2 - Orthogonal Metamodel Hierarchy

The hierarchy is now non-linear or ***orthogonal***, respectively, since the layers are no longer arranged vertically but instead the ***ontological layers*** (i.e. layers connected by ontological instantiation) are nested horizontally within the ***linguistic layers*** L1 and L2 which are connected by linguistic instantiation.

The most important difference to the linear approach in Figure 1 is that the instance-of relationships between ontological and those between linguistic layers now differ significantly. Whereas linguistic instance-of relationships are a concept well-known from MOF, an ontological instance-of relationship is established by an explicit association with that name. The instantiation semantics associated with instance-of can be defined suitably by constraints in L2. One potential problem with this approach is ambiguous classification, first identified by Atkinson and Kühne [7].

This problem is a rather subtle one and depends largely on the interpretation of the model hierarchy. If a clean separation of the physical and logical models exists, the linguistic classes can simply be omitted, adopting a purely ontological view on a model. With the linguistic metamodel adopting the role of a description language for the ontological hierarchy, there is no longer any ambiguity as can be easily seen in the figure.

However, in our opinion, unlike in the solutions presented in the literature so far (cp. Atkinson and Kühne [5] [4], as well as Riehle et al. [28] and Álvarez et al. [2]), there is more than one dichtonomy to consider in this context. For this reason, we precisely distinguish between the two concepts of physical vs. logical modeling and the division between linguistic and ontological modeling.

A **physical metamodel** is a metamodel which is described in the form of program code and data structures. Its instances are described by a program state but can be used to generate code for a physical metamodel corresponding to the instance or to serialize it in some form. On the other hand, a **logical metamodel** is an instance of a physical metamodel which describes a metamodel in the form of a program state. Its instances are also described by a program state using elements of the same physical metamodel (e.g. in the context of MOF and the Java Metadata Interface (JMI) implementation [12]).

The other important aspect is to distinguish between different purposes for instantiation, i.e. either to define the syntax of a language or to describe ontological relationships, although it might be argued that ontological metamodeling is a subset of linguistic metamodeling, defining a DSL. Therefore the following definition is applied in the context of our studies: **linguistic metamodeling** uses a metamodel to describe a language syntax without a concrete real-world mapping, whereas **ontological metamodeling** uses metamodels to describe domain specific hierarchies. This differentiation enables us to better describe the relationship between the different axes of metamodeling.

The decision between a linear or non-linear metamodel hierarchy depends on the situation. In terms of model **complexity**, two cases need to be distinguished: small hierarchies with few layers and larger hierarchies with several, equally frequented layers. As an example of this distinction, the UML would be considered a small hierarchy, since only 2 of its layers are really relevant for typical usage profiles. Due to a certain overhead from the linguistic metamodel non-linear hierarchies will account for a significantly higher model complexity than linear ones for small hierarchies. Larger hierarchies, on the other hand, tend to become inflated by replicated concepts over several layers. Regarding the **expressional strength** of those two basic architectures, it is hard as well to provide a general proposition. Since non-linear frameworks are more suitable for ontological hierarchies spanning multiple layers, it might be argued that the expressional strength of the non-linear approaches is higher when disregarding the impact other design choices. Generally speaking, the **consistency** of non-linear hierarchies will be higher, because each ontological layer will use the same linguistic elements. As with UML and its profiling mechanism, linear approaches offer a great degree of **extensibility**, which also applies for non-linear ones allowing easy editing at any level.

3.2 Layer Design Options

After having scrutinized the fundamental design option of linear and non-linear architectures, the following sections examine metamodeling variants dealing with more specialized questions concerning the design of modeling layers.

3.2.1 Number of Metamodel Layers

The choice regarding the number of layers in a metamodel hierarchy is, according to Bézivin and Lemesle, “a classical problem [..] in meta-modeling” [9]. Most approaches use four layers in a fashion similar to MOF and CDIF. This classical 4-layered hierarchy is usually sufficient for the conventional applications of metamodeling and for this reason is often accepted without question.

Using fewer layers is in many cases possible, whereas more layers require a non-linear metamodel hierarchy, as the conventional infrastructures such as MOF do not support more than 4 layers despite claims to the contrary (cf. [27]). Since the suitability of a specific number of model layers depend largely on the application it is used for, especially in the context of software engineering and MDD, a detailed analysis of different numbers of model layers is an obligatory first step.

To a certain degree, the number of modeling layers influences the **complexity** of a metamodel hierarchy with each layer added, especially in the case of linear hierarchies. A replication of concept might thus generate a negative impact on the hierarchy’s **consistency**. Considering ontological hierarchies, a higher number of layers can increase the expressional strength, and if the number

of layers can be kept flexible, an improved degree of **extensibility** will result.

3.2.2 Explicit or Implicit Real World Level

According to the definition provided in section 2, models are a representation of systems in the real world. Therefore a mapping between model elements and elements in the real world exists at least implicitly (cf. [20]).

Opinions on how this real world mapping should be represented in the context metamodeling are divergent. Atkinson and Kühne advocate the real world elements to be located on the M0 level (cf. [4]). The benefits of such an **explicit real world level**, however, are not immediately obvious, especially as it excludes a direct mapping of the metaelements to real world concepts. Therefore, if a solid definition of the language semantics exists, there is little need for any additional mappings to real world elements which results in an **implicit real world level**.

On the other hand, it might be argued that such a mapping is part of the semantics definition and therefore has no role in a metamodel. Harel and Rumpe criticize the fact that many researchers are unaware of the fact that a metamodel is a pure syntax definition (cf. [19], [20]) and the explicit real world level might be one manifestation of this misconception.

With regard to the quality criteria, this design choice has little influence on the overall architecture. No definite proposition can be given, whether the **consistency** of the hierarchy is improved by an explicit real world level, whereas **complexity** is definitively increased. Since the real world mapping occurs at a single layer only, the overall **expressional strength** of the metalayers remains unchanged. Being also effectively irrelevant for **extensibility** and **robustness to change** in general, it is our impression that the inclusion of a real world level is of little value added in most metamodel hierarchies, even though it is discussed in literature.

3.2.3 Axiomatic or Recursive Top Level

In a strict metamodel hierarchy all layers are defined by the metamodel situated in the next higher level with the exception of the highest layer which of course has no metamodel above it. The top-most layer can either be ***recursively defined***, i.e. self-describing, or modeled as an ***axiomatic layer***. MOF [27] and CDIF [10] are examples for recursive top-levels. Also, Riehle et al [28] give an example for a logical recursive metamodel. Some researchers though, such as Seidewitz [29] or Geisler et al. [14], are opposed to the recursive top-level concept.

The main advantage of a **recursive top level** is that the model hierarchy is self-defining (cf. MOF [27]). Harel and Rumpe consider this solution to be "elegant" and useful "from a pragmatic point of view" ([19], pg. 17), as users are probably already familiar with the language by the time they look at the metamodel. However, they also point out that the recursive metamodel is not self-sufficient and must be supplemented with alternative definitions.

The potential problems of a recursive top level described in the literature are somewhat hard to grasp. While recursive definitions also exist in other areas such as mathematics and are well-accepted there, they are difficult to understand as information is "coming from thin air" ([8], also see [19]). Figure 3 gives a small MOF-derived example for this problem. All elements in the MOF metamodel (i.e. the middle layer in the figure) which are instantiated are themselves instances of `Class`. Thus they differ only in their attributes and the associations and it is unclear how the different types of elements can be distinguished based on the recursive definition alone. This impression of incompleteness is caused by the fact that the instantiation semantics, which are not semantics at all but rather part of the syntax (similarly to the "context conditions" mentioned by Harel and Rumpe [20]), are not visible in the model. Nevertheless, these definitions exist and are part of the metamodel and therefore the "thin air" theory is not true and one of the main perceived drawbacks of recursive top level layers is void.

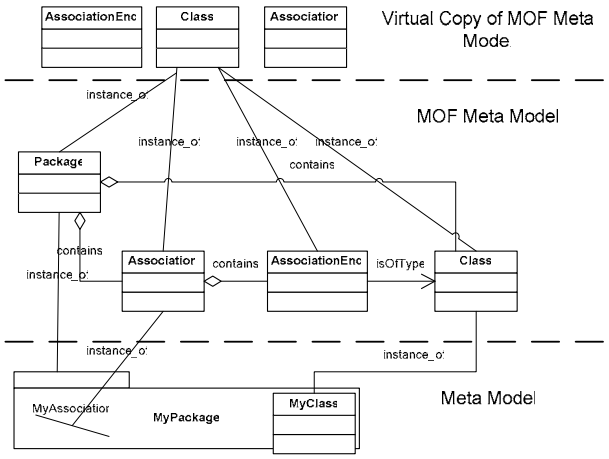

Figure 3 - Insufficient Recursive Definition

Seidewitz also criticizes recursive (or in his terms "reflexive") metamodels (cf. [29], [30]) which is mostly due to the fact that the recursion has to be resolved by duplicating the metamodel. In our opinion, this view sums up to the statement that without a proven valid base metamodel, the validity of all other models with regard to their immediate higher level metamodel is of questionable value. However, to us it is unclear why such a resolution of the recursion should be required.

An **axiomatic top level metamodel** uses another language to describe the top level metamodel. The most simple example is specifying the metamodel in natural language or some formal language. For example, the simplified metamodel in the previous example would require about a page of natural language explanations.

With all these problems of recursive definitions, an **axiomatic top level definition** appears to be an attractive alternative. Therefore, Geisler et al. propose the use of a formal metalanguage to avoid "self-referencing problems" ([14]). In the context of non-linear metamodel hierarchies (see section 3.1) the highest ontological layer can easily be seen as axiomatic with the linguistic metamodel taking the part of the metalanguage. On the other hand, Atkinson [8] finds this top level design problematic because its elements cannot be treated as objects. Albin even indirectly denies the possible existence of axiomatic metamodels by saying that "every model has a metamodel that describes it, although the metamodel may be implicit." ([1], chapter 11). Indeed it is questionable, why scenarios where the metalanguage (e.g. natural language) used to avoid the recursive definition was recursively defined itself should be preferable to a recursive top-level model.

The vast number and variety of statements made on this fundamental design option in literature is complicating the evaluation in terms of discrete quality criteria. For example, while some see a recursive definition as a tool for reducing **complexity**, there is also a strong opposition to this thesis. This dichotomy might result from the fact that a recursive definition can only be considered self-sufficient if implemented correctly, which is a complex task in its own right – as it is the case for **consistency**. Expressional strength, extensibility, and robustness to change, in our opinion, are fundamentally unaffected by the top level design.

Despite good arguments for recursive definitions the most prominent real world examples tend to have a recursive top level (cp. MOF and CDIF). In our opinion, in a linear hierarchy, a recursive metamodel, augmented with natural language clarifications, is preferable to a purely axiomatic approach. However, it is even better to have a recursive metamodel and its explanations offered as an alternative to an existing formal definition, as is proposed by Harel and Rumpe [19]. In a non-linear hierarchy, the highest ontological layer can be considered axiomatic as it is defined in terms of the linguistic metamodel which is beyond the scope of the ontological aspects.

3.3 Variants of Instantiation Semantics

Having studied the many possible layering concepts, the rules interconnecting those layers, i.e., the semantics for instantiating an element on a lower level, need to be reconsidered in more detail.

3.3.1 Strictness Definition

Strictness is a concept that provides order to the model layers in a metamodel hierarchy. It is often used implicitly in architectures such as MOF where it is an integral aspect. Atkinson and Kühne state in their definition of strictness that model elements should generally only have relationships within their own layers and not between layers, with the exception of the (possibly implicit) *instance-of* relationship, which represents the connection between the different layers. Another restriction introduced by their definition is that an element in a model layer can only instantiate elements of its immediate parent layer. They provide the following formal definition:

In an n-level modeling architecture, M_0, M_1, ..., M_{n-1}, every element of an M_m-level model must be an instance-of *exactly one element of an M_{m+1}-level model, for all $0 \leq m < n-1$, and any relationship other than the* instance-of *relationship between two elements X and Y implies that level(X) = level(Y)* [5]

On the other hand, a relaxation of strictness can be used to avoid **replication of concepts**. Figure 4 shows an example metamodel hierarchy with 3 layers (cp. [15] p. 65). The top layer, called ontology layer, describes a business ontology which is based on a "Convergent Architecture" architectural model proposed by Hubert [23].

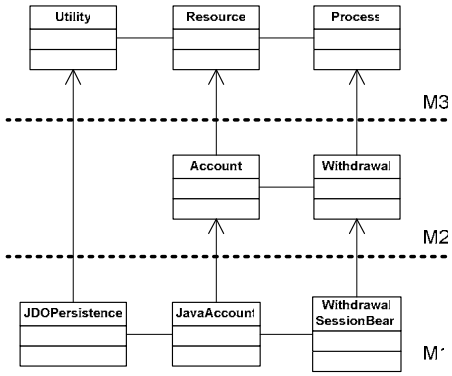

Figure 4 - A Scenario With Relaxed Strictness

In the figure a replication is avoided by allowing the instance-of relationship to skip a layer (relaxation of strictness). For an in-depth examination of this specific problem please refer to Gitzel and Merz [15].

Since a rigid strictness definition can lead to a replication of concepts, it can thus negatively affect both the **complexity** and the **consistency** of a model hierarchy (see above, cf. also [15]). Even though relaxed strictness stems replication, it might increase the degree of "perceived complexity". However, no strictness will have a negative impact on hierarchies due to interpretation intricacy and circular dependencies. A relaxation of strictness is deemed to improve **extensibility** by decoupling the metalayer to be extended from its directly adjacent layers. Besides merely theoretical advantages this might be particularly useful in the context of ontological hierarchies (cf. Figure 1). In many cases, **robustness to change** will be unaffected by the degree of strictness, since most changes to an upper layer diffuse transiently down the hierarchy anyway, a conclussion which was also substantiated by our personal experiences.

3.3.2 Shallow vs. Deep Instantiation

The possibility of choosing different instantiation semantics has already been hinted at in the previous discussion. Before this subject is addressed in the following section though, the concepts of deep and shallow instantiation should be introduced as a decision in this regard which impacts the design of the instantiation semantics as a whole.

A metamodel hierarchy supports **deep instantiation**, if it is possible for a class to make statements about its instances and their instances in turn transitively. If a class, on the other hand, can only affect its direct subclass, the system only supports **shallow instantiation**. The boundaries between the two design variants are sometimes blurred, though.

In a **linear** metamodel hierarchy, **shallow instantiation** can lead to several problems. All core modeling concepts, for instance, have to be defined in the top layer m leading to a replication of concepts if layers other than m-1 are using those elements. In a **non-linear** setup, layer-spanning linguistic concepts can be defined as part of the physical metamodel, negating the disadvantages described (cp. section 3.1).

Deep instantiation offers the advantage that concepts can be defined at a relatively high level to hold true for all sub layers without having to replicate this information on every layer. Atkinson and Kühne [5] [7] propose the concept of **potency** for

attributes as an implementation of deep instantiation which is illustrated in Figure 5. The potency P of an attribute A is an integer value which denotes the influence on instances of the class C which contains the attribute. This value is decremented after the instantiation of C and transferred to the instance.

If an attribute (or field as it is called in [7]) is defined as being **dual**, it can also be assigned values if the potency is still more than zero, e.g. "Name" in the very right column of Figure 5.

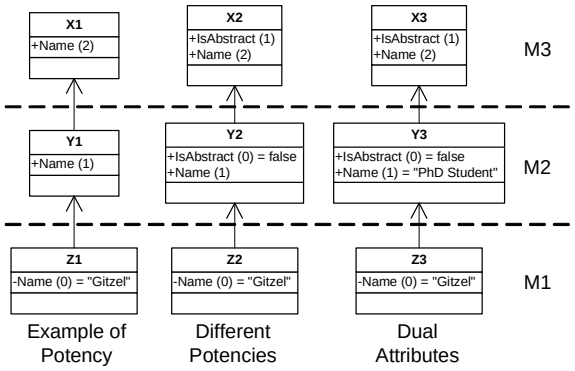

Figure 5 - Potency of Attributes

An interesting scenario occurs when combining potency with a **relaxation of strictness** (see section 3.3.1). The question arising is whether potency should be reduced by one for each instantiation or by one for each layer boundary crossed. Another question to answer when implementing a potency-based hierarchy is whether **type, multiplicity, and name** of the attribute should be immutably tied to those defined by its topmost definition or not.

While deep instantiation in the form of potency works and has a certain elegance, it implicitly restricts the number of layers and is not necessarily needed for a metamodel hierarchy. In combination with a relaxed strictness definition, the same modeling power is achieved without the need to implement a new mechanism which can hardly be realized using existing metamodeling APIs anyhow. On the other hand, if potency is an existing concept in a modeling language, it is easier to use than this alternative.

With regard to the quality criteria applied in this study, the instantiation depth design decision seems to generally point in favor of the deep option, which is a useful tool to reduce **complexity** and improve **consistency** by providing a single point of definition. Thus, the metalayers' ability to influence other layers beyond their immediate instances also increases the expressional strength of the hierarchy.

3.3.3 Other Aspects of Instantiation Semantics

Besides the aspects of strictness and shallowness, **instantiation semantics**, covers all the rules by which a type influences the make-up of its instances and which can take all kinds of form, letting them elude classification. One central question is whether uniform instantiation semantics should apply between all layers or whether each instantiation step should be able to have its own instantiation semantics (layer-dependent).

The **uniform** approach seems to make more sense, since the model hierarchy is easier to understand keeping the set of rules to a minimum. This approach enables the arbitrary addition of new model layers without the need to specify new instantiation semantics and thus potentially reduces **replication of concepts**. However, such an instantiation semantics is not implemented in the existing **linear** metamodel hierarchies which use **layer-dependent instantiation semantics** instead, **non-linear** solutions, on the other hand, can easily apply the same instantiation rules to all layers. Atkinson provides an example of a layer-independent instantiation semantics, where the instance of an association can also be an association with all the implied semantics and syntax, and the instances of (meta)classes can also be (meta)classes (cp. [8]).

Considering the evaluation of different metamodel hierarchies, this design option has a great impact on the quality of the overall architecture. Since type-instance relationships form an integral part of any hierarchy, a non-uniform instantiation semantics as well as a poorly designed uniform one, can negatively influence both **complexity** and **consistency** due to replication issues and possible confusion. A decision for a non-uniform instantiation semantics offers more potential for **extensibility**, but from our experience, there are rarely cases where an extension of the instantiation semantics is desirable and the cost of the increased complexity is too high to be offset by the meager benefits.

3.4 Linguistic Model Element Definitions

The linguistic model elements defined for a metamodel hierarchy heavily influence its nature. In a **linear** hierarchy, these elements are normally defined in the topmost layer, e.g. using one of the established standards such as MOF, whereas a **non-linear** framework generally has to provide more linguistic elements, such as `Class`**,** `Object`**,** or `MetaClass` allowing the modeling of the ontological hierarchy.

Among the proposed solutions presented here, is a naïve approach described and criticized by Atkinson and Kühne [5], which has one class for each layer. The drawback of this approach is that the number of layers will be fixed by the number of model elements defined as in the case of MOF (cp. section 3.2.1). On the other end of the scale is a solution also offered by those authors which has only a single class, **ModelElement** containing a level attribute that identifies the corresponding level the element is found on, effectively making it a `Class`, `Object`, or `MetaClass` [5]. A third variant, again proposed by Atkinson, consists of the model elements `(Meta)Class`, **`Clabject`**, and `Object`. A `Clabject` here is both an instance and a type. Thus it has both links and associations, attributes and attribute values [8]. A model hierarchy based on these elements also implies an axiomatic top level (see section 3.2.3).

The solution we propose is located somewhere in between the first two. There are three classes, `Class`, `MetaClass`, and **`ModelLayer`** where each element is associated with a `ModelLayer` by containment. This approach allows the relatively easy introduction of new model layers at the bottom or in between, without having to change all elements' layer attribute and the inclusion of information about the layers which can be used to check the compliance to domain-specific constraints. Effectively, a `MetaClass` has all the properties of a `Clabject` but does not need to be the instance of some type, merging the two concepts. Our simplification reduces the number of model elements without reducing the power or quality of the model.

Further issues concerning linguistic model elements include, for instance, whether to model **methods** or **operations** (cp. [8] and

[14]) as well as static and dynamic relationships interconnecting the other model elements. According to Atkinson [8], an Association is called a **dynamic relationship**, because it can be instantiated. A Link, which cannot be instantiated, is called a **static relationship**, even though this terminology is usually only applied to those links which do not occur in the lowest model layer, e.g. generalization and aggregation (containment) relationships.

Overall, the number of possible elements is probably infinite; however, it is feasible to identify several elements such as *Classes*, *Metaclasses*, or *Associations* which form a solid backbone of a linguistic metamodel. Due to the many options, it is important to provide a clear definition of the elements semantics and behavioral constraints as well as choose them wisely with regard to the intended application, e.g. code generation in software engineering.

Due to the multitude of options available in this context, it is again difficult to make a general statement concerning the evaluation by criteria. The **complexity** is influenced both by the number of elements and their particular design. As an example, the introduction of links can help reduce the complexity of a hierarchy for the user (e.g. generalizations), whereas the complexity of the underlying implementation increases. Conversely, a limited number of elements might reduce the **expressional strength** of the metalayers by subsuming several instances in shared metaclasses which in turn convey less information about the instances. The **consistency** of a hierarchy is also strongly interrelated with the choice of linguistic model elements, e.g. overlapping responsibilities can easily lead to inconsitencies. In general, the most expensive changes to a metamodel hierarchy are those made to its linguistic elements. This conclusion can be ascribed to the fact that those elements mostly reside on the higher level or, in case of a non-linear architecture, directly affect all ontological layers. If linguistic metaelements covering a wide range of ontological instances can be identified, they account for a certain degree of **robustness to change**, since changes to the ontological model layers are unlikely to require a modification at the linguistic level. Similarly, an intelligent choice of elements positively influences the extensibility.

3.5 Summary of Findings

As we have mentioned in the beginning, we have analyzed the available options for metamodel hierarchies from the viewpoint of the goals of general applicability and ease of use at all levels of abstraction. Based on this goal, we identified the criteria of complexity, consistency, expressional strength, extensibility, and robustness to change. Since the emphasis between these criteria will probably vary for individual projects and since not all of the design option described can be limited to a finite number of choices, it is not possible to make a general recommendation here. However, *tendencies* can be identified, especially when looking at those design options which are limited to a finite number of choices or whose possible values can at least be grouped into categories. Table 1 shows the tangible design options for metamodel hierarchies in an overview, assigning each of the possible choices a rating where applicable.

There are several findings that can be noted. First of all, the real world level discussion seems to be largely pointless, at least in the light of our criteria and thus, an explicit real-world level is not worth the effort. The discussion whether an axiomatic or recursive top-level model is the better choice, on the other hand, cannot be solved, because the interpretation on its impact depends largely on one's standpoint. Thus, we refrain from a recommendation in this context.

Another interesting fact is that there are quite a few values where the contribution towards a specific criterion depends on the right circumstances. For example, a wide range of linguistic model elements can lead to less complex models, however, without effort, a very complex but wide-ranged metamodel can be designed, for example, by adding useless elements.

However, there are some choices where a clear statement can be made as they are suitable for most situations. For example, a non-linear hierarchy is required to allow choices such as relaxed strictness or deep instantiation to be realized. Besides, non-linearity, there are some other choices, where a general (but not universal) recommendation can be offered. For instance, relaxed strictness is a preferable choice when possible, avoiding many of the problems caused by traditional strictness. Also, the linguistic model, if carefully designed, can provide a number of improvements in nearly all categories. On the other hand, a layer-dependent instantiation semantics should be avoided at all costs, increasing the complexity to an unacceptable degree, unless there are but a few layers.

4. CONCLUSION

The preceding examination of fundamental metamodel hierarchy design options, their advantages and drawbacks, as well as their interrelatedness outlined the complex decision process when constructing those hierarchies for different problem areas within the domain of software engineering in particular.

A robust set of definitions has been created from the myriad of conflicting opinions on the subject matter. Options available to the designer of a metamodel hierarchy have been analyzed and evaluated by a set of criteria described in section 2. Using these definitions and the taxonomy for metamodels, the benefits of metamodeling for MDD can be exploited more deliberately in future research endeavors.

In the context of an upcoming dissertation on at the University of Mannheim, metamodeling as an enabling technology for domain-specific code generation is evaluated using the example of web applications for e-business purposes. Part of this work is the *Ontological Metamodel Extension for Generative Architectures* (OMEGA, cf. working paper [16]), a metamodel hierarchy based on an early version of the design space described here. As future work, it is planned to use the design space for a metamodel hierarchy in the context of an ongoing research project, called *CollaBaWue*, which deals with collaborative, component-based software development within the semantic domain of financial service providers (see [11] and [20]).

Impact of Design Options on the Quality of a Metamodel Hierarchy	Complexity	Consistency & Precision	Expressional Strength	Extensibility	Robustness to Change
Linearity					
Linear	-	**0**	**0**	+	**0**
Non-Linear	+	++	+	+	**0**
Number of Layers					
Less than 4 Layers	**0**	**0**	**0**	**0**	**0**
4 Layers (cf. MOF)	**(-)**	**(-)**	**0**	**0**	**0**
More than 4 Layers	**(-)**	**(-)**	+	**(+)**	-
Real World Level Explicitness					
Explicit	-	**(+)**	**0**	**0**	**0**
Implicit	**0**	**0**	**0**	**0**	**0**
Top Level Definition					
Axiomatic	+/-	+/-	**0**	**0**	**0**
Recursive	+/-	+/-	**0**	**0**	**0**
Strictness Definition					
Strict	-	-	**0**	**0**	**0**
Relaxed	**(-)**	**0**	**0**	+	**0**
None	+	--	--	+	**0**
Instantiation Depth					
Deep	++	++	**0**	**0**	**0**
Shallow	-	-	**0**	**0**	**0**
Instantiation Semantics					
Uniform	**(+)**	**(+)**	**0**	-	-
Layer-dependent	--	-	**0**	+	+
Scope of Linguistic Model					
Wide	**(+)**	**(-)**	**0**	**(+)**	**(+)**
Medium	**(+)**	**0**	**0**	**(+)**	**(+)**
Narrow	**(-)**	**0**	**(-)**	**(+)**	**(+)**
Legend: -- to ++ (more or less pos./neg. influence, with -- being unacceptable), **0** (no influence on this criterion) +/- (depending on standpoint), **(_)** (under certain circumstances)					

Table 1 - The Design Options and Their Impact on a Hierarchy's quality

5. REFERENCES

[1] Albin, S. (2003): The Art of Software Architecture: Design Methods and Techniques, John Wiley & Sons, March 2003.

[2] Álvarez, J., Evans, A., and Sammut, P. (2001): Mapping between Levels in the Metamodel Architecture. In: Gogolla, M. and Kobryn, C. (Editors): *UML 2001, Lecture Notes in Computer Science 2185*, Springer, New York, 34-46, 2001.

[3] Atkinson, C. and Kühne, T. (2003): Calling a Spade a Spade in the MDA Infrastructure, In: *Proceedings of the Metamodeling for MDA First International Workshop*, York, UK, November 2003, 9-12.

[4] Atkinson, C. and Kühne, T. (2003): Model-Driven Development: A Metamodeling Foundation. In: *IEEE Software*, September/October 2003 (Vol. 20, No. 5), IEEE, 36-41.

[5] Atkinson, C. and Kühne, T. (2002): Rearchitecting the UML Infrastructure. In: *ACM Transactions on Modeling and Computer Simulation*, Vol. 12, No. 4, October 2002, 290-321.

[6] Atkinson, C. and Kühne, T. (2002): The Role of Metamodeling in MDA. *International Workshop in Software Model Engineering (in conjunction with UML '02)*, Dresden, Germany, October 2002. http://www.metamodel.com/wisme-2002/

[7] Atkinson, C. and Kühne, T. (2001): The Essence of Multilevel Metamodeling. In: Gogolla, M. and Kobryn, C. (Editors): *UML 2001, Lecture Notes in Computer Science 2185*, Springer, New York, 19-33.

[8] Atkinson, C. (1997): Meta-Modeling for Distributed Object Environments, In: *Proceedings of the 1st International Enterprise Distributed Object Computing Conference (EDOC'97)*, IEEE Computer Society 1997, 90-101.

[9] Bézivin, J. and Lemesle, R. (1998): *Ontology-Based Layered Semantics for Precise OA&D Modeling*. In: Lecture Notes in Computer Science 1357, Springer 1998, 151-154.

[10] CDIF Technical Committee (1994): *CDIF – CASE Data Interchange Format*, Extract of Interim Standard, EIA/IS-107, Electronic Industries Association, January 1994.

[11] COLLABAWUE project homepage. http://www.collabawue.de/

[12] Dirckze , R (Spec. Lead) (2002): *Java™ Metadata Interface(JMI) Specification - JSR 040, Version 1.0 Final Specification*. Java Community Process, June 2002. http://jcp.org/aboutJava/communityprocess/final/jsr040/

[13] Ehrig, H. (1979): *Introduction to the Algebraic Theory of Graph Grammars* Proc. Int'l Workshop Graph-Grammars and Their Application to Computer Science and Biology, V. Claus, H. Ehrig, and G. Rozenberg, eds., LNCS 73, Springer-Verlag, 1979.

[14] Geisler, R., Klar, M., and Pons, C. (1998): Dimensions and Dichotomy in Metamodeling, In: *3rd BCS-FACS Northern Formal Methods Workshop*, Ilkley, UK. 14th-15th September 1998.

[15] Gitzel, R. and Merz, M. (2004)**,** How a Relaxation of the Strictness Definition Can Benefit MDD Approaches With Meta Model Hierarchies, In: *Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2004)* , 19-21 July, 2004, Orlando, USA, International Institute of Informatics and Systemics (IIIS), 62-67. http://www.bwl.uni-mannheim.de/Schader/_files/ gitzel-Strictness.pdf

[16] Gitzel, R. and Korthaus, A. (2004): The Role of Metamodeling in Model-Driven Development, In: *Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2004)* , 19-21 July, 2004, Orlando, USA, International Institute of Informatics and Systemics (IIIS). http://www.bwl.uni-mannheim.de/Schader/_files/ gitzel-MetaMDD.pdf

[17] Gitzel, R., Ott, I., and Schader, M. (2004), *Ontological Metamodel Extension for Generative Architectures (OMEGA)*, Working Paper, University of Mannheim, Department of Information Systems III, June, 2004. http://www.bwl.uni-mannheim.de/Schader/_files/ gitzel-omega.pdf

[18] Gitzel, R. and Schader, M. (2003): Generation of XML-based Web Applications Using Metamodels. In: *Proceedings of the 7th IASTED International Conference on Internet And Multimedia Systems And Applications (IMSA'03)*, 13.-15. August, 2003, Honolulu, USA. http://www.bwl.uni-mannheim.de/Schader/_files/ gitzel-XMLPaper.pdf

[19] Harel, D. and Rumpe, B. (2000): *Modeling Languages: Syntax, Semantics, and All That Stuff – Part I: The Basic Stuff.* The Weizmann Institute of Science, Rehovot, Israel, MCS00-16. http://www4.in.tum.de/~rumpe/ps/Modeling-Languages.pdf

[20] Harel, D. and Rumpe, B. (2004): Meaningful Modeling; What's the Semantics of "Semantics"?, In: IEEE Computer, October 2004 (Vol. 37, No. 10), IEEE, 64-72.

[21] Herbsleb, J.D. and Mockus, A.: Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering. In: Proceedings of the 9th Eurooean Software Engineering Conference held jointly with the 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM Press, 2003, 138-147

[22] Hildenbrand, T. and Korthaus, A. (2004): A Model-Driven Approach to Business Software Engineering, In: *Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2004), Volume IV Information Systems, Technologies and Applications*, Orlando, Florida, USA, July 18-21, 2004, 74-79. ftp://ftp.wifo.uni-mannheim.de/pub/PEOPLE/korthaus/ HildenbrandKorthaus-SCI2004.pdf

[23] Hubert, R. (2002): *Convergent Architecture – Building Model-Driven J2EE Systems with UML*, John Wiley & Sons, 2002.

[24] Muller, P.-A., Studer, P., and Bézivin, J. (2003): Platform Independent Web Application Modeling. In: P. Stevens et al. (Eds.): *UML 2003*, LNCS 2863, Springer, 220–233.

[25] Odell, J. (1994): Power Types. In: *Journal of Object Oriented Programming*, May, 1994

[26] OMG (2003): *XML Metadata Interchange (XMI) Specification Version 2.0 formal/03-05-02*. OMG, May 2003, http://www.omg.org/docs/formal/03-05-02.pdf

[27] OMG (2002): *Meta Object Facility (MOF) Specification, Version 1.4*. OMG, April 2002. http://www.omg.org/cgi-bin/doc?formal/2002-04-03

[28] Riehle, D., Fraleigh, S., Bucka-Lassen, D., and Omorogbe, N.: The Architecture of a UML Virtual Machine. In: *Proceedings of the 2001 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'01)*, ACM Press, 2001.

[29] Seidewitz, E. (2003): What Models Mean. In: *IEEE Software*, September/October 2003 (Vol. 20, No. 5), IEEE, 26-32.

[30] Seidewitz, E. (2003): *What Do Models Mean?* OMG Document ad/03-03-31, OMG, March 2003, http://www.omg.org/docs/ad/03-03-31.pdf

[31] Völter, M. (2000): Metamodellierung, http://www.voelter.de/services/mdsd.html