A meta model transformation approach towards harmonisation in information system modelling

J.L.H. Oei
Centre for Telematics and Information Technology
University of Twente
P.O.Box 217, NL-7500 AE Enschede, The Netherlands
E-mail:oei@cs.utwente.nl

Abstract

The Meta Model Transformation (MMT-) approach provides a method to relate meta models of languages in an open ordering and transformation scheme by means of a set of basic meta model transformations. Each of the basic transformations is motivated by its influence on certain, application-independent language characteristics. In this paper, the MMT-approach, which may serve for comparison, integration, and evolution of languages, is formalised, and a suitable language for this approach is presented. Finally, it is applied for constructing an order on meta models, which harmonises languages for modelling information systems from different perspectives.

Keywords

information systems modelling; method engineering; meta modelling; meta model transformations; harmonisation; comparison; integration; evolution

1 THE NEED FOR HARMONISATION IN IS MODELLING

Due to the wild growth of modelling languages or formalisms, information systems practitioners, and method and language engineers are facing an enormous Methodology Jungle (a term introduced in [AF88]) in which one easily gets lost. To provide a standardised language or set of languages would certainly simplify some of the problems in the information system field. However, because of suitability reasons individual or co-operating organisations may want to select different languages for different aspects or perspectives of the system. Furthermore, the set of languages being used may evolve in time. A threat to this situation is that if one selects and applies various unrelated and incompatible languages, communication will be hampered and the overall consistency will not be guaranteed. In this paper, we aim at harmonisation of a wide variety of languages for modelling information systems rather than standardisation of single languages.

One approach to achieve harmonisation of languages could be to provide an all-embracing reference language, which covers the notions and constructs of all the various existing languages, and to provide suitable translation mechanisms between the reference language and the others. Attempts of this sort have been made (e.g. [Ama87], [OHM$^+$88]). The major problem of such an approach is that it is easy to create yet another, even more comprehensive language whose notions and constructs are (partially) not covered by the reference language, and thus not (entirely) compatible with it.

An approach which seems to be more successful, is to give up the idea of a reference language for good, and to provide a suitable, open, ordering and transformation scheme for languages. In such a scheme, the various languages are specified, as well as the transformational relationships between them. It is called open, because any existing or future language can be incorporated in this scheme.

In the present paper, we present and formalise an approach to such an ordering and transformation scheme, the Meta Model Transformation (MMT) approach. This approach* has been introduced in [OHFB92], [OF94], and [FO94].

The organisation of this paper is as follows. First we discuss the rationale behind our approach in section 2. In section 3 we present and formalise the MMT-approach, before presenting a suitable and more specific language for the MMT-approach in section 4. The MMT-approach and its language are illustrated in section 5 by constructing a partial order of meta models, which harmonises languages modelling information systems from different perspectives. Finally, this paper is concluded by summarising the benefits of our approach and identifying further research in section 6.

2 RATIONALE OF THE MMT-APPROACH

In this section we present the way of thinking that is underlying our MMT-approach. This includes a presentation of the main characteristics of this approach. Furthermore, some characteristics of languages and models are discussed that are of importance.

The way of thinking underlying our approach can be characterised as follows:

- *Rather than 'pre-cooked solutions' for problems being caused by the Methodology Jungle, we provide a 'powerful means' for solving these problems.*
- *We do not wish to choose sides in a sometimes 'religous war of competing approaches', but we prefer to provide a 'peaceful landscape' for an oecumenical information systems community.*
- *Our approach does not aim at standardisation of one or a few modelling languages, but at harmonisation of all sorts of languages.*

How can we accomplish this harmonisation of modelling languages? In order to answer this question it is obvious that we need means and criteria to compare modelling languages. Several approaches can be, and have been adopted. First of all, several comparisons on the basis of a list or matrices of criteria have already been performed (e.g. [OSV82], [SZ92]). These comparisons are often qualitative and/or situation-dependent of kind. A second approach is to compare on the basis of experiences of practitioners. Such an empirical

*In earlier papers it was referred to as Meta Model Hierarchy-approach. As it appeared that this naming caused confusion because of different connotations we decided to rename our approach.

approach is to a large extent subjective. Finally, a formal, objective and application-independent comparison is possible on the basis of meta models of languages.

Any of these approaches may have its benefits for different situations and purposes. However, we claim that a formal comparison on the basis of meta models is the best approach, if one aims at a 'true' harmonisation of modelling languages. This is even more the case if we aim at automated support in the end. The reason is that the meta model of a language contains the (formal) description of the essentials of a language, i.e. the structure of modelling constructs provided for modelling concepts in an application domain.

The MMT-approach that is proposed has the following characteristics:

- *Formalisation of the comparison process by identifying a set of basic relations among meta models of languages.*
  Meta-modelling (e.g. [Bri90]) is the process of formalising languages. In the MMT-approach, a set of basic relations on meta models is defined. The MMT-approach aims at a comparison on a formal basis, because of several reasons. First of all, formalisation prevents the chance of ambiguity which distorts the process of communication. Furthermore, formalisation is a prerequisite for automatic support.
- *A set of application-independent language characteristics is identified. Each of the MMT-relations on meta models is related to one of these language characteristics.*
  We distinguish three characteristics of languages which are independent of the application and situation in which they are used: genericity, liberality, and expressiveness. It is noted that we realise that the ultimate success of an IS-method for a specific IS-project to a large extent also depends on situational factors. An approach which take these situational factors into account can be found in [HBO94].
- *Provision of a set of basic operations on meta models.*
  To facilitate evolution of languages and transformations between different languages, a set of basic operations on meta models is defined. Each of these basic operations corresponds to one of the basic relations defined between meta models. From these basic operations more complex operations can be constructed. Application of these operations on meta models results in meta model transformations.
- *The sets of basic relations and operations are in principle independent on the choice of a specific meta language.*
  In section 3 a presentation and formalisation of the MMT-approach is given which is as independent as possible of any modelling decisions. Of course, in order to apply the MMT-approach the meta model operations have to be 'operationalised', i.e. defined in a specific meta language with its semantic domain. In this paper the MMT-approach is operationalised in a meta language being designed from an object-role modelling perspective in section 4.
- *The basic MMT-relations form partial orders on meta models.*
  Each MMT-relation defines a partial order on meta models. Furthermore, the set of partial orders being defined can be merged into one partial order which is referred to as the overall MMT-order. Such a MMT-order is "upward compatible" with respect to the set of application-independent language characteristics. That is, a language based on meta model x which is "higher" in the MMT-order than another meta model y, is at least as generic, liberal, and expressive as a language based on meta model y.
- *A MMT-order is 'open' in the sense that all meta models of existing and not (yet) existing languages can be positioned in this partial order.*

We claim that given the existence of a highly expressive meta model as the minimal element of the MMT-order, the set of meta model operations is complete. That is, all meta models can be derived from this meta model by means of sequences of the provided meta model operations. The minimal element is not necessarily a very complex meta model, but preferably a small but highly expressive meta model. As existing languages usually differ in more than one aspect, meta models of these existing languages are usually related via intermediate meta models, which result from the application of a sequence of meta model operations. These intermediate meta models may or may not correpond to existing languages. The process of identifying these intermediate meta models may lead to the definition of a new language on the basis of such an intermediate meta model. Also an adjustment of an existing language to the needs of a specific situation can be facilitated by adjusting meta models by the provided meta model operations.

- *The validity of an MMT-order depends on the (shared) ontology of the meta modellers.* We do not claim that we can construct an ultimate and unique order on meta models. The reason for this is that any attempt made to modelling, including meta modelling, is dependent on the *Weltanschauung* of the modeller. In particular, an *ontology* of the world has to be assumed. An ontological theory is a theory that contains and interrelates ontological categories, or generic concepts representing components or features of the world ([Bun77],[Bun79]). Consequently, for each specific situation we will construct a MMT-order which is suitable for that situation, and is based on an ontology shared by the involved (meta-)modellers.

2.1 Characteristics of languages

Suitability of languages for a specific situation depends on the purpose of use. It can be compared on the basis of many different criteria (e.g. [Law88], [OHM$^+$88], [HO92]). Quality of models specified in languages can also be measured by means of many different properties. [BCN92] distinguishes expressiveness, simplicity, minimality, formality as well as readability, self-explanatory, extensibility, and normality of ER-schema's (being intensional models specified in the Entity Relationship modelling language). In this section, characteristics of languages are discussed which are important for the MMT-approach. These characteristics are situation-aspecific and application-independent.

Definition 2.1

Genericity is a characteristic of a language stating to what degree a given language is generally applicable, or specifically tailored for the particular task of modelling some domain.

In the area of programming languages third-generation languages like PASCAL or C are often more generic than fourth-generation languages like SQL, which offer more specialised language constructs for a specific type of domain (for SQL: the domain of databases). This is also the case in the information systems modelling area. Specialised languages for modelling domains in e.g. decision support systems, CAD/CAM, office automation, and industrial (real-time) applications have been developed. Even within one family of languages more and less specialised languages exist. For example, in the family of data modelling languages, several extended ER-languages offering modelling constructs for special relationships like specialisation, aggregation, and grouping have been developed. These extended ER-languages are more specialised than the original, more generic, ER-language proposed by [Che76].

Definition 2.2

Liberality is a characteristic of a language stating the degrees of freedom one has when modelling one and the same domain.

The more liberal a language is, the more semantically equivalent models of one and the same domain can be formulated. If the language allows one and only one model for each domain to be modelled, the liberality is minimal. We call such a language deterministic. In the programming area it is well-known that, using even one programming language, there are often many equivalent programs for solving the same problem. For example, a repeating loop can be programmed by both a 'do-while' and a 'repeat-until' statement. However, some languages are more restricted than others. A minimal language, for example, does not allow two different constructs for making the same statement; thus excluding either the 'do-while' or the 'repeat-until' construct. In the information system modelling area, we also have more liberal and more restrictive languages. For example, in the family of ER-languages and object-role modelling (ORM-)languages (eg. NIAM in [NH89], or PSM in [HW93]) both binary dialects and n-ary dialects exist. The binary dialect allowing only binary relationships is more restrictive and thus less liberal than the n-ary dialects. However, it should be noted that for each n-ary model a semantically equivalent binary variant exists. This implies that the same range of domains can be modelled.

Definition 2.3

Expressiveness is a characteristic of a language stating to what degree a given language is capable of describing any number of domains.

Note that in our definition of expressiveness it is not referred to the number of models that can be produced in that language, but to the number of domains that can be modelled. There may be several models modelling the same domain.

In the programming world we know the Turing-test, which determines whether a language is able to simulate a Turing-machine. Evidently, languages which are able to simulate a Turing-machine are more expressive than those which are not. In our area we also know about modelling languages which are more expressive than others. For example, modelling language for temporal ([McK86]) and evolving information systems ([OPF94], [Pro94]) have to be more expressive than languages for modelling snapshot systems. Another example are process- or action-oriented languages which do allow the specification of domains from an actor-perspective (e.g. [Agh86]), and proces-oriented languages which do not (e.g. A-schemas in ISAC [LGN81], and DFD [DeM78]). The latter languages can now be considered as degenerations of the former ones with respect to actor specification.

Note that it is difficult to assign absolute measures to these characteristics of languages. However, as it will be shown in the rest of this paper, it is possible to compare languages on the basis of these characteristics, resulting in relative measures. Furthermore, it is neither always possible nor desirable, to provide as much as possible of each of these measures. There will be often a balance or a trade-off.

For example, there may be a special-purpose language or language which is highly suited for that special purpose and for a specialist in that special field, but which has a low overall expressiveness. There may be a general-purpose language with a high expressiveness, but which is only moderately well-suited for many or most purposes. Very often, a trade-off between expressiveness and genericity is required. A modeller who likes his artistic freedom will probably prefer a highly liberal language. If we are aiming however at a discipline of modelling, artistic freedom and a liberal language will be undesirable. Rather we would prefer an entirely deterministic language([Fal93]).

A thing or domain (in the 'real world') can, in general, be described as various different models, using the same (non-deterministic, liberal) language or different languages. In order to be able to determine whether or not two models describe exactly the same thing, we have to establish the notion of equivalence of models.

This statement applies regardless of whether the models are described in the same or in different languages, and regardless of the metalevel of the models. Whether equivalence of models can be established with confidence in practice, depends both on the syntactical correctness and the semantical validity of the transformation rules.

Finally, it is important to notice that in general, genericity, liberality, and expressiveness, do not imply eachother. For example, a binary dialect of an ER-language is more restrictive (thus less liberal) than its n-ary variant. However, both languages have the same expressiveness, because for each n-ary ER model, there exists an equivalent binary model. On the other hand, the characteristics are not orthogonal. A more restrictive language cannot be more expressive than the more liberal language.

2.2 Meta models and application models

Modelling languages are intended for specification of particular aspects of the application domain in consideration, also called the universe of discourse ([Gri82]). A common property of all languages, is that they are all based upon a structure of modelling constructs, which reflects their particular view on the phenomena of the world. The description of the structure of modelling constructs on which a language is based, and all constraints thereupon is given in the meta model of that language. A description of a specific application domain in that language is called an application model. A type/instance or intensional/extensional relationship exists between meta model and application model. That is, the meta model determines all application models that can be produced in a language. We will base our comparison of languages on the basis of comparison of their meta models as this allows for a formal comparison on the essence, i.e. the underlying modelling constructs, of a language.

Note that the type vs. instance dichotomy is in principle different from the metamodel vs. application model dichotomy. The first is based upon whether a description is given of what is possible or what actually exists. The latter dichotomy distinguishes between what is application-independent and what is dependent on a specific application domain. To illustrate this point: the type-instance dichotomy may also be present within an application model. The traditional distinction between database schema and database instances is an example of the type-instance dichotomy within an application model. Not only the database schema is now an instantation of the meta model, but the complete application model is, including database schema, database instances, and their interrelationship ([BF91], [OPF94]).

A meta model is application-independent, and consists of a structure of modelling constructs and a set of constraints thereupon. An instantiation of a meta model, ie. a particular application model, should be in accordance with the structure of constructs, and should satisfy the set of constraints. Note that the modelling constructs (of a language) are the representational means for describing concepts which are conceived in the application domain by the modeller.

How these modelling constructs are actually denoted in the language is specified in the concrete syntax of the language. Together with the specification of the abstract syntax which is included in the meta model, a complete description of the language is obtained, which is called the language model. The semantics of a concrete application model in the language is determined by the mapping of its denotation on the mathematical structure underlying its meta model.

3 META MODEL TRANSFORMATION-APPROACH

In this section we present and formalise the Meta Model Transformation (MMT-) approach. The main characteristic of this approach is that it identifies relations and operations on meta models. As we wish to present this approach on a formal basis, first a mathematical structure for meta models is given.

3.1 Structure for meta models

For specifying meta models the following mathematical structure is defined (see Appendix 1 for the mathematical conventions being used):

Definition 3.1

Let $\mathcal{MM}$ be the set of all possible meta models, and let $\mathcal{O}_{All}$ be the set of all possible modelling constructs.

A meta model $m \in \mathcal{MM}$ is a 2-tuple $m = \langle \mathcal{OS}, \mathcal{CS} \rangle$ where $\mathcal{OS}$ is a structure of modelling constructs and $\mathcal{CS}$ is a set of constraints upon that structure. Here is $\mathcal{CS} \subseteq \boldsymbol{\Gamma}(\mathcal{OS})$, where $\boldsymbol{\Gamma}(\mathcal{OS})$ denotes the set of all possible laws or rules on $\mathcal{OS}$.

A $\mathcal{OS}$ is a tuple consisting of the following components:

1. *A non-empty set $\mathcal{O} \subseteq \mathcal{O}_{All}$ of modelling constructs*[†]*.*
2. *A partial function $\mathsf{DefRule} : \mathcal{O} \rightarrowtail \boldsymbol{\Gamma}(\mathcal{OS})$. The function yields the defining (or derivation) rule of a Construct in terms of other constructs.*

Note that in this definition we have not chosen for a particular classification of modelling constructs yet. Such a classification depends on the ontology chosen by the language designer. The only choice we have made is that a meta model consists of a structure of

[†]We prefer the term 'modelling constructs' rather than 'objects', because of the overload of connotations to the term 'object'. Also the term 'concept' is avoided in the meta model of a language, as we feel that a language construct is rather *representing or modelling* a concept than *being* a concept.

modelling constructs together with a set of constraints defined on this structure. Furthermore, some modelling constructs can be defined in, or can be derived from, others. Examples of a possible DefRule are the subtype defining rules and derivation rules in NIAM. If we apply the meta model structure to predicate logic, the set of constructs is formed by the predicates, definitions represent the defining rules, and axioms are included in the constraint set. In section 4 the proposed structure for meta models is refined to facilitate a more suitable specification of meta models.

A meta model determines the set of possible application models in a language. In other words, an application model is said to be an instantiation or population of its meta model.

Definition 3.2

Let Ω be the universe of instances of modelling constructs,then
let POP $\subseteq$ $(\mathcal{O}_{All} \rightarrow \wp(\Omega))$ *be the set of all possible population functions from modelling constructs on instances.*
A specific instantiation of a meta model is now given by a population function Pop $\in$ POP *from its set of modelling constructs to a set of instances for each of these constructs .*

A meta model provides the abstract syntax of application models. Several concrete syntaxes can be chosen for one and the same meta model. Different choices result in different concrete denotations of the same application model.

3.2 Meta model transformations

The Meta Model Transformation (MMT-) approach provides an open transformation scheme between meta models. A basic set of transformations is provided, which allows the positioning of any meta model relative to any other meta model. The transformation scheme is called open, because any meta model of existing or future languages can be positioned in this transformation scheme.

Our basic assumption is that for each finite set of languages, there exists a meta model which is at least as expressive, generic, and liberal as the meta models of the languages being considered. Note that this 'upper-bound' meta model is not necessarily the meta model of one of the considered language. It is preferably a *small and simple* meta model from which all modelling constructs of the considered languages can be derived by definitions, rather than a large and complex meta model comprising all modelling constructs. The main reason for this is that we do not believe in the feasability of a *universal* meta model which is *large and complete*, and which embraces the constructs provided by all existing or future languages.

Before introducing a set of basic relations or transformations between meta models which allow us to construct partial orders on meta models being 'upward compatible' with respect to genericity, restriction, and expressiveness, we first define the environment of a meta model.

The semantics of languages is determined by the (possible) instantiation or population of its meta model, i.e. the possible application models. Such an application model usually, but not necessarily includes a type-instance dichotomy itself. In order to define MMT-transformations which not only influence syntax of languages, but also their semantics, it is important to take all possible population functions into consideration. For this purpose, the so-called *environment* of a meta model is defined as all populations being allowed for that meta model. The set of all possible environments can now be defined as follows:

Definition 3.3
$\mathsf{ENV} \subseteq \wp(\mathsf{POP})$. *An environment* $\mathsf{env} \in \mathsf{ENV}$ *of a meta model is the set of all population functions being allowed for that meta model.*

In the following we define three relations between meta models. Each of these relations corresponds to a pair of meta model operations: specialisation/generalisation, restriction/liberalisation, and degeneration/generation respectively. These operations on meta models themselves will be defined in a more specific meta language in section 4. Illustrative examples of the MMT-relations and operations can be found in section 5 and in [OF94].

Specialisation

The first basic transformation between meta models is motivated by providing the possibility to influence genericity of languages. A language can be made more specific tailored for modelling a particular (type of) application domain by introducing more specialised constructs. In order to preserve upward compatibility with respect to genericity, these specialised constructs have to be defined in terms of already existing constructs. Note that it is the task of the meta-modeller or method engineer to identify these defining rules between modelling constructs. For the meta model of such a special-purpose language, this means that new constructs have been added to the structure of constructs. The resulting meta model is called a specialisation of the meta model of the more generic language. Note that a specialisation alone does not affect the expressiveness of a language, it only facilitates the modeller in describing application domains in more compact application models by using more specialised constructs which are, however, derivable, i.e. they are defined in terms of already existing constructs. The inverse of specialisation is called generalisation.

Definition 3.4 (*specialisation-relation*)
A meta model y is called a specialisation of meta model x iff the modelling constructs in x are a subset of those of y, and if all other modelling constructs of y are defined in terms of the structure of constructs in x. Furthermore, the populations being allowed for their shared constructs are identical.

Let $\preceq_{\mathsf{Spec}} \subseteq (\mathcal{MM} \times \mathsf{ENV}) \times (\mathcal{MM} \times \mathsf{ENV})$ *be a relation between metamodels with their environments, where*

$$
\begin{aligned}
&\langle x, \mathsf{POP}_x\rangle \preceq_{\mathsf{Spec}} \langle y, \mathsf{POP}_y\rangle \equiv \\
&\quad \mathcal{O}_x \subseteq \mathcal{O}_y \wedge \\
&\quad \forall_{o \in \mathcal{O}_y \setminus \mathcal{O}_x} [\mathsf{DefRule}_y(o) \in \Gamma(\mathcal{OS}_x)] \wedge \\
&\quad \forall_{\mathsf{Pop}_y \in \mathsf{POP}_y} \exists_{\mathsf{Pop}_x \in \mathsf{POP}_x} [(o \in \mathcal{O}_x \cap \mathcal{O}_y \Rightarrow \mathsf{Pop}_y(o) = \mathsf{Pop}_x(o)) \wedge \\
&\quad (o \in \mathcal{O}_y \setminus \mathcal{O}_x \Rightarrow \mathsf{Pop}_y(o) \models \mathsf{DefRule}_y(o))]
\end{aligned}
$$

Note that in this definition the symbol $\preceq_{\mathsf{Spec}}$ is used both as a relation symbol and an operator symbol. That is, $x \preceq_{\mathsf{Spec}} y \equiv \langle x, y\rangle \in \preceq_{\mathsf{Spec}}$. Furthermore, note that a shorthand notation is used for the components of the structure of meta models. For example, $\mathcal{O}_x$ is the projection on $\mathcal{O}$ in the meta model of x.

Lemma 3.1 From the definition it follows that the relation $\preceq_{\mathsf{Spec}} \subseteq (\mathcal{MM} \times \mathsf{ENV}) \times (\mathcal{MM} \times \mathsf{ENV})$ is reflexive, asymmetric, and transitive, and consequently is a partial order on meta models. The partial order is said to be upward compatible with respect to genericity. That is, $x \preceq_{\mathsf{Spec}} y$ implies that the language based on meta model x is at least as generic as a language based on meta model y.

Restriction

The introduction of new constructs, most often, comes together with some general rules concerning these new constructs. Furthermore, it may be desirable to restrict the freedom to model one and the same domain by means of different, but equivalent application models. Both situations require a restriction of the possible set of application models. In other words, one requires to decrease the liberality of a language. To arrive at the meta model of such a less-liberal, and more restrictive language, it means that constraints (which are not implied yet by the existing constraints) are added to the constraint set being imposed on the structure of constructs. The resulting meta model is called a restriction of the meta model of the language offering more liberality. The inverse of restriction is called liberalisation.

Definition 3.5 (*restriction-relation*)
A meta model y is called a restriction of meta model x iff the set of possible populations of y is a subset of the set of possible populations in x.

Let $\preceq_{\mathsf{Restr}} \subseteq (\mathcal{MM} \times \mathsf{ENV}) \times (\mathcal{MM} \times \mathsf{ENV})$ *be a relation between meta models with their environments, where*

$$\langle x, \mathsf{POP}_x \rangle \preceq_{\mathsf{Restr}} \langle y, \mathsf{POP}_y \rangle \equiv$$
$$\forall_{o \in \mathcal{O}_x} \forall_{\mathsf{Pop}_y \in \mathsf{POP}_y} \exists_{\mathsf{Pop}_x \in \mathsf{POP}_x} \left[\mathsf{Pop}_y(o) \subseteq \mathsf{Pop}_x(o) \right]$$

Lemma 3.2 The relation $\preceq_{\mathsf{Restr}} \subseteq (\mathcal{MM} \times \mathsf{ENV}) \times (\mathcal{MM} \times \mathsf{ENV})$ is a partial order on meta models. This partial order is upward compatible with respect to liberality. That is, $x \preceq_{\mathsf{Restr}} y$ implies that the language based on meta model x is at least as liberal as a language based on meta model y.

Degeneration

A decrease in expressiveness of a language means that the possible set of things that can be modelled by the structure of modelling constructs is restricted. As stated in the previous section, restriction does not guarantee this. To arrive at the meta model of such a less-expressive language, a construct has to be *degenerated* from the structure of constructs. This degeneration is accomplished by restricting the environment of the meta model. As a result of this restriction of the environment, the construct to be degenerated cannot be instantiated anymore. Furthermore, the domain that could be modelled by the removed construct, cannot be modelled in another way by constructs from the remaining structure of constructs. The resulting meta model is called a degeneration of the meta model of the language being more expressive. Note that a degeneration is different from the inverse of a specialisation, because a specialisation does not affect expressiveness of a language and a degeneration does. The inverse of degeneration is called generation.

Definition 3.6 (*degeneration-relation*)

A meta model y is called a degeneration of a meta model x iff there is a subset of the constructs of x which cannot be populated anymore in y. Moreover, the concepts being modelled by the degenerated constructs can also not be modelled by any other constructs in meta model y.

Let $\preceq_{\mathsf{Deg}} \subseteq (\mathcal{MM} \times \mathsf{ENV}) \times (\mathcal{MM} \times \mathsf{ENV})$ *be a relation between metamodels with their environments, where*

$$\langle x, \mathsf{POP}_x \rangle \preceq_{\mathsf{Deg}} \langle y, \mathsf{POP}_y \rangle \equiv$$
$$\exists_{\mathcal{O}_d \subseteq \mathcal{O}_x} \forall_{o \in \mathcal{O}_x} \forall_{\mathsf{Pop}_y \in \mathsf{POP}_y} \exists_{\mathsf{Pop}_x \in \mathsf{POP}_x} \left[\mathsf{Pop}_y(o) = \mathsf{Pop}_x(o) \backslash \left\{ \omega | (\omega \models \mathsf{DefRule}_x(o_d)) \land o_d \in \mathcal{O}_d \right\} \right]$$

Lemma 3.3 The relation $\preceq_{\mathsf{Deg}} \subseteq (\mathcal{MM} \times \mathsf{ENV}) \times (\mathcal{MM} \times \mathsf{ENV})$ is a partial order on meta models. This partial order is upward combatible with respect to expressiveness. That is, $x \preceq_{\mathsf{Deg}} y$ implies that the language based on meta model x is at least as expressive as a language based on meta model y.

As a result of degeneration no degenerated constructs can be populated anymore. Note that though the degenerated constructs still appear in the structure of constructs, they will have no denotations in the corresponding language. We are aware that it is possible to transform a degenerated meta model in an equivalent meta model in which the degenerated concepts do not appear in the structure of constructs. However, our definition does not ask for a complex restructuring of the structure of concepts, and moreover, it allows for a more simple definition of the inverse transformation of degeneration. It also prevents the meta modeller from introducing a degenerated construct later by specialisation, because specialisation cannot recover a degenerated environment. In our approach a degenerated construct can only be recovered by an inverse degeneration operation.

Finally, it almost directly follows from the definitions of restriction and degeneration that a degeneration can be seen as a very strict restriction (The setdifference in the definition of degeneration implies the subset relation in the definition of restriction!).

Lemma 3.4 $x \preceq_{\mathsf{Deg}} y \Rightarrow x \preceq_{\mathsf{Restr}} y$

The main reason that we define degeneration separately is that a degeneration necessarily decreases expressiveness, whereas a restriction which is not a degeneration restricts liberality, but may or may not affect expressiveness. Furthermore, as already stated, constructs with empty populations are usually not represented in a concrete denotation of an application model, and therefore not visible to the user.

3.3 A Meta Model Transformation-order on meta models

As most existing meta models are related by means of a sequence of basic operations on meta models (e.g. a specialisation usually comes together with some restrictions on the specialised constructs), and because we are not always interested in the details of such a composed transformation, i.e. the basic transformations that are part thereof, we now define a partial order on meta models which is based on any sequence of specialisations, restrictions, and degenerations.

Based on the three partial orders on meta models defined in the previous subsections, the following Meta Model Transformation-relation on meta models is defined.

Definition 3.7 (*MMT-relation*)
$\preceq_{\mathsf{MMT}} \subseteq (\mathcal{MM} \times \mathsf{ENV}) \times (\mathcal{MM} \times \mathsf{ENV})$, *where*

$$\begin{aligned}
&\langle x, \mathsf{POP}_x\rangle \preceq_{\mathsf{MMT}} \langle y, \mathsf{POP}_y\rangle \equiv \\
&\quad \langle x, \mathsf{POP}_x\rangle \preceq_{\mathsf{Spec}} \langle y, \mathsf{POP}_y\rangle \lor \\
&\quad \langle x, \mathsf{POP}_x\rangle \preceq_{\mathsf{Restr}} \langle y, \mathsf{POP}_y\rangle \lor \\
&\quad \langle x, \mathsf{POP}_x\rangle \preceq_{\mathsf{Deg}} \langle y, \mathsf{POP}_y\rangle
\end{aligned}$$

Theorem 3.1 The relation $\preceq_{\mathsf{MMT}} \subseteq (\mathcal{MM} \times \mathsf{ENV}) \times (\mathcal{MM} \times \mathsf{ENV})$ is a partial order on meta models. The partial order is upward compatible with respect to genericity, liberality, and expressiveness. That is, $x \preceq_{\mathsf{MMT}} y$ implies that the language based on meta model x is at least as generic, liberal, and expressive as a language based on meta model y.

Proof:
We will not proof this theorem in detail here, but the important part of the proof is that from the definitions of the three relations it follows that if a meta model $x \prec y$ with respect to one relation, $y \prec x$ cannot be true for one of the other relations.

4 A META LANGUAGE FOR THE MMT-APPROACH

Up to now in our definitions concerning the MMT-approach, we abstracted from as many as language design decisions (such as classifications of modelling constructs) to be as generic as possible. However, for specifying meta models, and transformations among them in a convenient way, we need to choose a suitable meta language.

For specifying meta models many languages can be used. In this paper we have aimed at a so-called meta-language (i.e. a language for describing languages) which is able to specify meta models of a wide range of languages, moreover formal, but also relatively simple to understand. In other words, we like to have a fair trade-off between expressiveness, formality, and understandability. In this section, we introduce such a meta language by refining the structure of modelling constructs presented in the previous section. A structure of modelling constructs is chosen based on an ontology which is characteristic to that of object-role modelling languages like NIAM([NH89]), and PSM([HW93]).

Definition 4.1

Let $\mathcal{MM}$ be the set of all possible meta models,
A meta model $m \in \mathcal{MM}$ is a 2-tuple $m = \langle \mathcal{OS}, \mathcal{CS} \rangle$ where $\mathcal{OS}$ is a structure of modelling constructs and $\mathcal{CS}$ is a set of constraints upon that structure. Here is $\mathcal{CS} \subseteq \Gamma(\mathcal{OS})$, where $\Gamma(\mathcal{OS})$ denotes the set of all possible laws or rules on $\mathcal{OS}$.

A $\mathcal{OS}$ is a tuple consisting of the following components:

1. *A non-empty set $\mathcal{O}$of modelling Constructs or Objects.*
2. *A partial function $\mathsf{DefRule} : \mathcal{O} \rightarrowtail \Gamma(\mathcal{CS})$. The function yields the defining (or derivation) rule of a Construct in terms of other constructs.*
3. *A set $\mathcal{R}$ of Roles which a Construct may play.*
4. *A set $\mathcal{E}$ of Simple Constructs. Simple Constructs are Constructs: $\mathcal{E} \subseteq \mathcal{O}$.*
5. *A partition $\mathcal{F}$ of the set $\mathcal{P}$, where $\mathcal{P} \subseteq \mathcal{O} \times \mathcal{R}$ is the set of pairs of constructs playing roles. This set is called the Predicator set. The elements of $\mathcal{F}$ are called Composite Constructs. Composite Constructs, which represent relationships between constructs, are Constructs themselves[‡]: $\mathcal{F} \subseteq \mathcal{O}$.*

Furthermore, the auxiliary function $\mathsf{Fact} : \mathcal{P} \rightarrow \mathcal{F}$ is defined by : $\mathsf{Fact}(p) = f \iff p \in f$, which yields the composite construct in which a predicator is contained.

This structure can serve as the basic mathematical structure for object-role modelling languages. Different extensions of this structure lead to different (dialects of) ORM-languages.

In order to facilitate the specification of meta models in a more convenient way, special relationships are introduced. We define specialisation, unification, grouping, and sequencing (being called specialisation, generalisation[§], powertyping, and sequence types in PSM [HW93], respectively.) The first three relationships correspond to subsets, union, and powersets in set theory, respectively.

Definition 4.2

The 5-tuple for a structure of modelling constructs $\mathcal{OS}$ of the previous definition is extended to a 10-tuple with the following components:

1. *A binary relation $\mathsf{Spec} \subseteq \mathcal{E} \times \mathcal{O}$ on Constructs, expressing specialisation. Note that the specialisation defining rule of a construct o is given by $\mathsf{DefRule}(o)$.*
2. *A binary relation $\mathsf{Unif} \subseteq \mathcal{E} \times \mathcal{O}$ on Constructs, expressing unification.*
3. *A set $\mathcal{G}$ of Set constructs. Set constructs are constructs: $\mathcal{G} \subseteq \mathcal{O}$.*
4. *A set $\mathcal{S}$ of Sequence constructs. Sequence constructs are constructs: $\mathcal{S} \subseteq \mathcal{O}$.*
5. *A function $\mathsf{Elt} : (\mathcal{G} \cup \mathcal{S}) \rightarrow \mathcal{O}$. The function Elt yields the construct being Element of a Set or Sequence construct.*

The meta language being proposed, which we call ML is based on the formalisation of

[‡]This forming of complex structures is also known as objectification.

[§]Unlike in PSM, we reserved the term generalisation for the inverse of specialisation

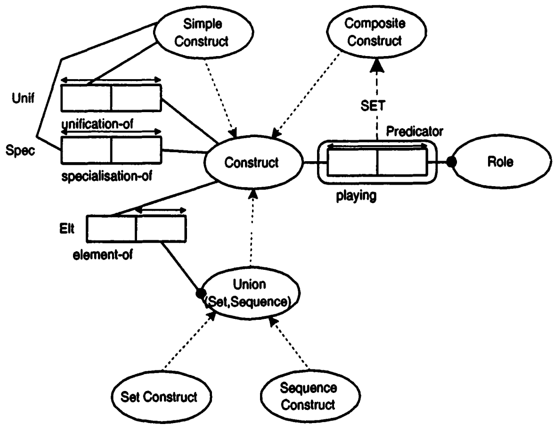

Figure 1 Structure of constructs of ML in ML

object-role modelling (ORM) languages being found in PM ([BHW91]), and its extension PSM ([HW93]). A more elaborate formalisation including all axioms on the specialisation and unification hierarchy, sets, and sequences is provided in [Hof93] in the context of PSM.

We adjusted the formalisation of object-role models to become more suitable specifically for the specification of all sorts of meta models. For this reason, our formalisation differs slightly from that of PSM, which is intended to be suitable for data-intensive domains in general.

In [HPW93] a semi-natural language called LISA-D for specifying rules and constraints on the mathematical structure for object-role models is defined. In the following we will use this semi-natural language for specifying constraints and (specialisation) defining rules. Furthermore, graphical conventions for object-role models are used for denoting (parts of) meta models in an alternative way.

In figure 1 the structure of constructs is represented in the graphical convention we will use in this paper. The graphical convention is almost identical to that of PSM or NIAM: circles denote Constructs, boxes denote Roles, lines represent predicators connecting Constructs to Roles. With respect to the special relationships: specialisation is denoted by solid arrows; unification by dotted arrows, and grouping and sequencing relationships are denoted by dotted arrows with a label SET or SEQ respectively. Finally, some graphical constraints are available: dots on predicators represent a total-role constraint, and double-headed arrows represent uniqueness constraints.

In the following, meta models are defined in the mathematical structure if we are explicitly interested in the formalisation. In other cases a graphical convention similar to that of PSM or NIAM is chosen for denoting meta models. Note, however, that each graphical representation has a corresponding formal definition.

4.1 MMT-transformations in ML

In this subsection, operations based on the three basic MMT-relations are defined in our chosen meta language. Each operation is formally defined using denotational semantics ([Sch86]). The semantics is given by a mapping of each syntactical construct (denoting a particular transformation) on its meaning in the mathematical structure for meta models given before. Note that for each of the three basic MMT-operations an inverse operation can be defined.

From the basic set of MMT-operations more complex transformations can be composed. As most existing languages differ in more than one of the characteristics mentioned, usually a sequence of basic and composed transformations are needed to relate one language to another.

The denotational semantics of each operation will be defined by the following mapping:

Definition 4.3
$\mathbb{D} : \mathit{MMTOperations} \times \mathcal{MM} \times \mathsf{ENV} \rightarrow \mathcal{MM} \times \mathsf{ENV}$
where MMTOperations is the set of syntactical constructs representing specialisation, restriction, and degeneration, respectively.

The first MMT-operation we define is the specialisation operation. In our language the modelling construct to be defined can be a simple, composite, set, or a sequence construct.

Definition 4.4 (*specialisation-operation*)
Given $x \in \mathcal{MM}$ the meta model being specialised, $o \in \mathcal{O}_{All} \backslash \mathcal{O}_x$ the new construct to be defined, and $r \in \boldsymbol{\Gamma}(\mathcal{OS}_x)$ the defining rule for that new construct, let:
$\mathbb{D} : \mathit{MMTOperations} \times \mathcal{MM} \times \mathsf{ENV} \rightarrow \mathcal{MM} \times \mathsf{ENV}$, *where*

$$\begin{aligned}
&\mathbb{D}[\![\mathsf{SPEC\text{-}BY}(o : r)]\!]\,(\langle x, \mathsf{POP}_x\rangle) = \\
&\quad \langle\langle\langle \mathcal{O}_x \cup \left\{o\right\}, \mathsf{DefRule}_x \oplus \left\{o : r\right\}, \\
&\quad \textbf{if } o \in \mathcal{F}_{All} \textbf{ then } \mathcal{R}_x \cup \left\{rl | \exists_{o_i \in \mathcal{O}_x} \left[\mathsf{Fact}(\langle o_i, rl\rangle) = o\right]\right\} \textbf{ else } \mathcal{R}_x \textbf{ fi}, \\
&\quad \textbf{if } o \in \mathcal{E}_{All} \textbf{ then } \mathcal{E}_x \cup \left\{o\right\} \textbf{ else } \mathcal{E}_x \textbf{ fi}, \\
&\quad \textbf{if } o \in \mathcal{F}_{All} \textbf{ then } \mathcal{F}_x \cup \left\{o\right\} \textbf{ else } \mathcal{F}_x \textbf{ fi}, \\
&\quad \mathsf{Spec}_x \cup \left\{\langle o, o_j\rangle \,| \mathsf{Pop}(o) \subset \mathsf{Pop}_x(o_j) \wedge \mathsf{Pop}(o) \models r \wedge o_j \in \mathcal{O}_x\right\}, \mathsf{Unif}_x, \\
&\quad \textbf{if } o \in \mathcal{G}_{All} \textbf{ then } \mathcal{G}_x \cup \left\{o\right\} \textbf{ else } \mathcal{G}_x \textbf{ fi}, \\
&\quad \textbf{if } o \in \mathcal{S}_{All} \textbf{ then } \mathcal{S}_x \cup \left\{o\right\} \textbf{ else } \mathcal{S}_x \textbf{ fi}, \\
&\quad \textbf{if } o \in (\mathcal{G}_{All} \cup \mathcal{S}_{All}) \textbf{ then } \mathsf{Elt}_x \oplus \left\{o : \mathsf{Elt}(o) | \mathsf{Elt}(o) \models r\right\} \textbf{ else } \mathsf{Elt}_x \textbf{ fi}, \rangle, \\
&\quad \mathcal{CS}_x\rangle, \\
&\quad \left\{\mathsf{Pop} \oplus \left\{o : \omega\right\} | \mathsf{Pop} \in \mathsf{POP}_x \wedge \mathsf{Pop}(o) \models r\right\}\rangle
\end{aligned}$$

A restriction of a meta model is accomplished by adding constraints to the set of constraints of a meta model. This is reflected by the following definition.

Definition 4.5 (*restriction-operation*)
Given $x \in \mathcal{MM}$ the meta model being restricted, $c \in \boldsymbol{\Gamma}(\mathcal{OS}_x)$ the constraint being added, and $\neg(\mathcal{CS}_x \models c)$ reflecting that the added constraint is not already implied by the set of constraints, let:

$\mathbb{D} : \mathit{MMTOperations} \times \mathcal{MM} \times \textsf{ENV} \rightarrow \mathcal{MM} \times \textsf{ENV}$, *where*

$$\mathbb{D}[\![\textsf{RESTR-BY}(c)]\!]\,(\langle x, \textsf{POP}_x\rangle) \;=\; \Big\langle\big\langle \mathcal{OS}_x, \mathcal{CS}_x \cup \{c\}\big\rangle, \big\{\textsf{Pop} \in \textsf{POP}_x \,|\textsf{Pop} \models \{\mathcal{CS}_x \cup \{c\}\}\big\}\Big\rangle$$

Finally, the operation corresponding to the degeneration relation between meta models is defined as follows:

Definition 4.6 (*degeneration-operation*)
Given $x \in \mathcal{MM}$ the meta model being degenerated, and $o \in \mathcal{O}_x$ the constructs being degenerated from this meta model, let:
$\mathbb{D} : \mathit{MMTOperations} \times \mathcal{MM} \times \textsf{ENV} \rightarrow \mathcal{MM} \times \textsf{ENV}$, *where*

$$\mathbb{D}[\![\textsf{DEG-OF}(o)]\!]\,(\langle x, \textsf{POP}_x\rangle) = \\ \Big\langle\langle \mathcal{OS}_x, \mathcal{CS}_y\rangle, \Big\{\textsf{Pop}_y | \exists_{\textsf{Pop}_x \in \textsf{POP}_x} \forall_{o_i \in \mathcal{O}_y} \Big[\textsf{Pop}_y(o_i) = \textsf{Pop}_x(o_i) \backslash \big\{\omega | \omega \models \textsf{DefRule}_x(o)\big\}\Big]\Big\}\Big\rangle$$

The three MMT-operations defined, and their inverse operations form the basis for more complex operations. All these operations can be used to migrate from one meta model to another. In the next section these operations are applied to construct a MMT-order of meta models integrating different perspectives in information systems modelling.

5 A MMT-APPROACH FOR HARMONISATION OF IS MODELLING PERSPECTIVES

Several attempts have been made to develop an all-embracing framework or reference language, which covers the notions and constructs of all the various existing languages, and to provide suitable transformation mechanisms between the reference framework or language and the others (e.g. [Ama87], [OHM+88], [SZ92], [Con94]). The major problem of such an approach is that it is easy to create yet another, even more comprehensive language of which the notions and constructs are (partially) not covered by the reference language, and thus not compatible with it. Furthermore, the acceptance and validation of such a reference framework or language always depends on the acceptance of the ontology of the meta model developer. For this reason, we think that it is wiser to provide an open transformation scheme of meta models such as the MMT-approach. For a specific situation, a MMT-order for that situation can be constructed which is based on an ontology which is shared by the (meta-)modellers.

In this section the MMT-approach is applied to develop the first part of a partial order of meta models relating and integrating the data-, action-, and behaviour-oriented perspectives on information systems. The ontology underlying this specific MMT-order stems from an evolutionary and object-role modelling way of thinking.

In order to construct a MMT-order which comprises the meta models of a given set of languages, different approaches can be adopted. In [OF94] we distinguished a top-down, and a bottum-up approach. In this section, we adopt a top-down approach, i.e. a meta model is selected to serve as the minimal element of the MMT-order from which other meta models can be derived by sequences of MMT-transformations. Such a minimal element must be at least as generic, liberal, and expressive as any member of the set of

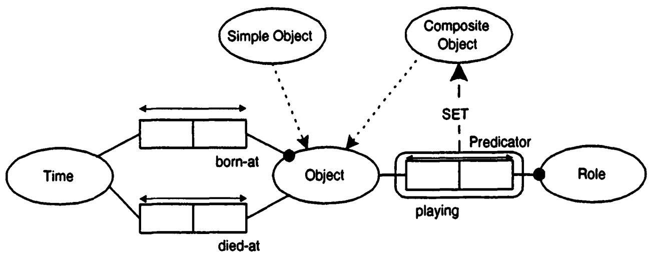

Figure 2 Meta model EIS: minimal element of the MMT-order

meta models to be ordered. For the minimal element of an MMT-order integrating the data-, action-, and behaviour-oriented perspective on information systems, a meta model is chosen which stems from an evolutionary way of thinking.

In [FOP92a], [FOP92b], and [OPF94] a conceptual framework for evolving information systems has been presented. In the EIS-framework the structure of modelling constructs has been considered as a black-box. In order to be more specific, we open this black-box, and classify the modelling constructs which are time-stamped, in accordance with the same object-role modelling perspective as adopted in our meta language.

The minimal element of the MMT-order that we will construct in this section meta model is represented in figure 2 following the graphical convention of our meta language ML. As explained, it follows from both an evolutionary and object-role modelling perspective.

Data-oriented modelling languages for snapshot systems, which abstract from the life-cycle of objects, are considered to be degenerations from this temporal meta model. As a consequence, meta models for data modelling languages like Entity Relationship Modelling, and Object Role Modelling, have degenerated the Time-construct by applying the degeneration operation to the temporal meta model.

In state-transition oriented languages it is often abstracted from the life-cycle of individual objects as can be derived from the meta model of figure 2. Rather, transitions of states are considered. In our ontology, states are considered as sets of objects (at a certain point of time). We now incorporate the modelling constructs State and Transition by applying a sequence of specialisation and restriction operations to our initial meta model.

In the following meta models and meta model transformations are specified in ML. The constraints and defining rules are represented in the semi-natural language LISA-D ([HPW93]).

Definition 5.1

Let mm_{TSTD} be the meta model resulting from applying the following sequence of meta model transformations to meta model mm_{EIS}:

$$\begin{aligned}\langle mm_{\mathsf{TSTD}}, \mathsf{POP}_{\mathsf{TSTD}}\rangle &= \mathsf{SPEC\text{-}BY}(\mathsf{Transition} : r_{\mathsf{Transition}}) \circ \\ &\quad \mathsf{SPEC\text{-}BY}(\mathsf{State} : r_{\mathsf{State}}) \circ \\ &\quad \mathsf{RESTR\text{-}BY}(c_{\mathsf{TSTD}})(\langle mm_{\mathsf{EIS}}, \mathsf{POP}_{\mathsf{EIS}}\rangle)\end{aligned}$$

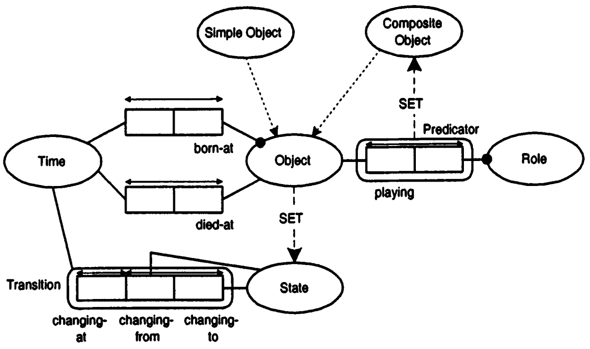

Figure 3 Meta model TSTD: integrating the data and behaviour oriented perspective

where

r_{State} ≡ LET State BE SET-OF Object
$r_{\text{Transition}}$ ≡ LET Transition BE CompObj
CONTAINING(State playing Role 'changing-from')
AND-ALSO(State playing Role 'changing-to')
AND-ALSO(Time playing Role 'changing-at')
c_{TSTD} ≡ the 'firing rules' for transitions

The resulting meta model, which we will call TSTD, is represented in figure 3. This meta model integrates the most fundamental concepts of data-oriented and behaviour-oriented languages. The meta models of behaviour-oriented languages like STD ([You89]) or Petri-nets([Pet81]), which abstract from the object-role relationships and the explicit specification of time points, can now be derived from this meta model by applying the degeneration operation to this meta model. A meta model of Petri-Nets needs a further specialisation of places and tokens.

Besides data- and behaviour-oriented languages there are also modelling languages which focus on the proces- or action-oriented perspective. These action-oriented languages generally models actions with their required inputs and outputs, which are sometimes called (input- and output) actands. Furthermore, in some languages it is possible to model the actor performing an action. In our ontology we consider actions to be sequences of transitions (with some purpose). For example, the action of writing this paper is a sequence of transitions starting from a state transition of the author (being the actor of the writing action) from being passive to being active, and ended by a state transition of the paper from being incomplete to complete, and a state transition of the author from being active to being passive again. The input- and output actands of an action are derived from the states involved in the sequence of state transitions defining the action. We now integrate the concepts of actor, action, and actand by means of a sequence of MMT-transformations on the meta model of figure 3.

Definition 5.2

Let mm_{AAA} be the meta model resulting from applying the following sequence of meta

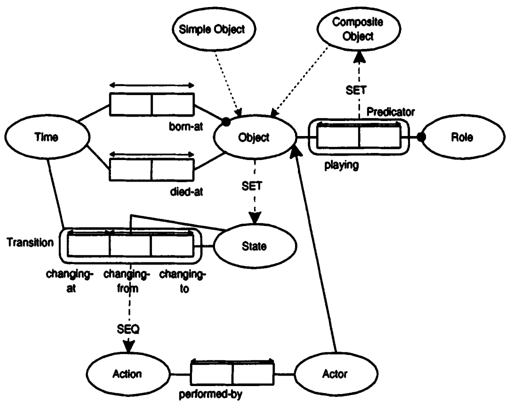

Figure 4 Meta model AAA: integrating the data-, action-, and behaviour-oriented perspective

model transformations to meta model mm_{TSTD}*:*

$$\begin{aligned}\langle mm_{\mathsf{AAA}}, \mathsf{POP}_{\mathsf{AAA}}\rangle \;=\;& \mathsf{SPEC\text{-}BY}(\mathsf{Actand} : r_{\mathsf{Actand}}) \circ \\ & \mathsf{SPEC\text{-}BY}(\mathsf{Actor} : r_{\mathsf{Actor}}) \circ \\ & \mathsf{SPEC\text{-}BY}(\mathsf{Action} : r_{\mathsf{Action}}) \circ \\ & \mathsf{RESTR\text{-}BY}(c_{\mathsf{AAA}})(\langle mm_{\mathsf{TSTD}}, \mathsf{POP}_{\mathsf{TSTD}}\rangle)\end{aligned}$$

where

$r_{\mathsf{Action}} \equiv$ LET Action BE SEQUENCE-OF Transition
$r_{\mathsf{Actor}} \equiv$ LET Actor BE Object playing Role 'is-actor'
$r_{\mathsf{Actand}} \equiv$ LET Actand BE State
$c_{\mathsf{AAA}} \equiv$ all necessary constraints on actions, actors, and actands

The resulting meta model, which we will call AAA, and which is graphically represented in figure 4, integrates the fundamental concepts behind data-, action-, and behaviour-oriented languages. Meta models of action-oriented languages like DFD, and ISAC A-schema's are considered to be degenerations of this integrated meta model.

In this section we have defined the following MMT-order on meta models. Note that languages based on the 'smaller' meta models are at least as generic, liberal, and expressive as languages based on 'larger' meta models.

$$\langle \mathcal{MM}_{\mathsf{EIS}}, \mathsf{POP}_{\mathsf{EIS}}\rangle \preceq_{\mathsf{MMT}} \langle \mathcal{MM}_{\mathsf{TSTD}}, \mathsf{POP}_{\mathsf{TSTD}}\rangle \preceq_{\mathsf{MMT}} \langle \mathcal{MM}_{\mathsf{AAA}}, \mathsf{POP}_{\mathsf{AAA}}\rangle$$

Note that by means of applying more sequences of (composed) MMT-operations the meta models of other existing or future languages can be constructed on demand.

6 CONCLUSION AND FURTHER RESEARCH

In this paper we presented the Meta Model Transformation approach, which can be characterised as:

- an approach towards *harmonisation of languages* rather than standardisation,
- providing *a method to relate their meta models*,
- by means of *a basic set of (meta-language independent) transformations: specialisation, restriction, and degeneration*,
- each influencing the *application-independent characteristics: expressiveness, liberality, and genericity.*

This MMT-approach has been formalised, a language facilitating this approach has been proposed, and has been applied to the construction of an order on meta models integrating the fundamental concepts of data-, action-, and behaviour-oriented modelling languages.

Our current research includes the improvement of the Meta Model Transformation language being proposed. Furthermore, in the context of the IFIP WG 8.1 Taskgroup FRISCO, the MMT-order is (implicitly) applied to construct a framework of the most essential information system concepts. Finally, guidelines for applying the Meta Model Transformation approach are being developed to arrive at a procedure which is as deterministic as possible. However, it should be realised that some non-determinism is inherent to the problem of modelling in general and meta modelling in particular.

The benefit of the MMT-approach is that it provides a formal basis for comparison, selection, integration, and evolution of modelling languages. Moreover in the future, a MMT-order, being a partial order on meta models with respect to genericity, liberality, and expressiveness can be useful in the process for supporting interoperability, and transparency between modelling languages.

ACKNOWLEDGEMENTS

I would like to acknowledge the contribution of Eckhard Falkenberg in this joint research. Sjaak Brinkkemper, Paul Grefen, Louis van Hemmen, and the anonymous referees are thanked for providing useful comments on an earlier version of this paper.

APPENDIX 1 MATHEMATICAL CONVENTIONS

In this appendix the mathematical notations used are described briefly.

- $\wp(A)$ denotes the powerset of set A.
- $f : A \rightarrow B$ defines a total function from A to B.
- $f : A \rightarrowtail B$ defines a partial function from A to B.
- $f \circ g(x) = f(g(x))$ denotes function composition.
- $f \oplus \{a : b\} = \{a : b\} \cup f \setminus \{x : y \in f | x = a\}$. The function $f \oplus \{a : b\}$ therefore behaves the same as function f except that it assigns a to b.

- $f[A'] = \{a : b \in f | a \in A'\}$. The function $f[A']$ is the function f restricted to a subdomain $A' \subseteq \mathsf{dom}(f)$.

REFERENCES

[AF88] D.E. Avison and G. Fitzgerald. *Information Systems Development: Methodologies, Techniques and Tools.* Blackwell Scientific Publications, Oxford, United Kingdom, 1988.

[Agh86] G.A. Agha. An overview of actor languages. *ACM Sigplan Notices*, 21(10), 1986.

[Ama87] Amadeus. The amadeus project, final report. Interprogram, Amstelveen, The Netherlands, June 1987.

[BCN92] C. Batini, S. Ceri, and S.B. Navathe. *Conceptual Database Design: an entity-relationship approach.* Benjamin/Cummings Publishing Company, Inc., 1992.

[BF91] S. Brinkkemper and E.D. Falkenberg. Three Dichotomies in the Information System Methodology. In P.W.G. Bots, H.G. Sol, and I.G. Sprinkhuizen-Kuyper, editors, *Informatiesystemen in beweging.* Kluwer, Deventer, The Netherlands, 1991.

[BHW91] P. van Bommel, A.H.M. ter Hofstede, and Th.P. van der Weide. Semantics and verification of object-role models. *Information Systems*, 16(5):471–495, October 1991.

[Bri90] S. Brinkkemper. *Formalisation of Information Systems Modelling.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1990.

[Bun77] M.A. Bunge. *Ontology I: The Furniture of the World*, volume 3 of *Treatise on Basic Philosophy.* D. Reidel Publishing Company, The Netherlands, 1977.

[Bun79] M.A. Bunge. *Ontology II: A World of Systems*, volume 4 of *Treatise on Basic Philosophy.* D. Reidel Publishing Company, The Netherlands, 1979.

[Che76] P.P. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[Con94] Euromethod Consortium. Euromethod: Work package 3, 1994.

[DeM78] T. DeMarco. *Structured Analysis and System Specification.* Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[Fal93] E.D. Falkenberg. An Approach to Deterministic Information Analysis. In *Proceedings of the International Conference on NIAM-ISDM'93, Utrecht, The Netherlands*, 1993.

[FO94] E.D. Falkenberg and J.L.H. Oei. Meta Model Hierarchies from an Object-Role Modelling Perspective. In T.A. Halpin and R. Meersman, editors, *Proceedings of the First International Conference on Object-Role Modelling, Magnetic Island, Australia*, pages 218–247, 1994.

[FOP92a] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. A Conceptual Framework for Evolving Information Systems. In H.G. Sol and R.L. Crosslin, editors, *Dynamic Modelling of Information Systems II*, pages 353–375. North-Holland, Amsterdam, The Netherlands, 1992.

[FOP92b] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. Evolving Information Systems: Beyond Temporal Information Systems. In A.M. Tjoa and I. Ramos, editors, *Proceedings of the Data Base and Expert System Applications Conference (DEXA 92)*, pages 282–287, Valencia, Spain, September 1992. Springer-Verlag.

[Gri82] J.J. van Griethuysen, editor. *Concepts and Terminology for the Conceptual Schema and the Information Base.* Publ. nr. ISO/TC97/SC5-N695, 1982.

[HBO94] F. Harmsen, S. Brinkkemper, and J.L.H. Oei. Situational Method Engineering for Information Systems Project Approaches. In A.A. Verrijn-Stuart and T.W. Olle, editors, *Methods and Associated tools for the Information Systems Life Cycle: Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle, Maastricht, The Netherlands*, pages 169–194. IFIP/North-Holland, 1994.

[HO92] T.A. Halpin and J.L.H. Oei. A Framework for Comparing Conceptual Modelling Languages. Technical Report 92-29, Department of Information Systems, University of Nijmegen, Nijmegen, The Netherlands, 1992.

[Hof93] A.H.M. ter Hofstede. *Information Modelling in Data Intensive Domains.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, 1993.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[Law88] D. Law. *Methods for comparing methods: techniques in software development.* NCC Publications, 1988.

[LGN81] M. Lundeberg, G. Goldkuhl, and A. Nilsson. *Information Systems Development - A Systematic Approach.* Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[McK86] E. McKenzie. Bibliography: Temporal Databases. *SIGMOD Record*, 15(4):40–52, December 1986.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach.* Prentice-Hall, Sydney, Australia, 1989.

[OF94] J.L.H. Oei and E.D. Falkenberg. Harmonisation of Information System Modelling and Specification Techniques. In A.A. Verrijn-Stuart and T.W. Olle, editors, *Methods and Associated tools for the Information Systems Life Cycle: Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle, Maastricht, The Netherlands*, pages 151–168. IFIP/North-Holland, 1994.

[OHFB92] J.L.H. Oei, L.J.G.T. van Hemmen, E.D. Falkenberg, and S. Brinkkemper. The Meta Model Hierarchy: A Framework for Information System Concepts and Techniques. Technical Report 92-17, Department of Information Systems, University of Nijmegen, Nijmegen, The Netherlands, 1992.

[OHM+88] T.W. Olle, J. Hagelstein, I.G. Macdonald, C. Rolland, H.G. Sol, F.J.M. van Assche, and A.A. Verrijn-Stuart. *Information Systems Methodologies: A Framework for Understanding.* Addison-Wesley, Reading, Massachusetts, 1988.

[OPF94] J.L.H. Oei, H.A. Proper, and E.D. Falkenberg. Evolving Information Systems: Meeting the ever-changing environment. *Information Systems Journal*, 4(3), 1994.

[OSV82] T.W. Olle, H.G. Sol, and A.A. Verrijn-Stuart, editors. *Information Systems Design Methodologies: A Comparative Review.* North-Holland/IFIP, Amsterdam, The Netherlands, 1982.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modelling of Systems.* Prentice-Hall, Englewoods Cliffs, New Jersey, 1981.

[Pro94] H.A. Proper. *A Theory for Conceptual Modelling of Evolving Application Domains.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1994.

[Sch86] D.A. Schmidt. *Denotational semantics.* Allyn and Bacon, Inc., 1986.

[SZ92] J.F. Sowa and J.A. Zachmann. Extending and formalizing the framework for information systems architcture. *IBM Systems Journal*, 31(3), 1992.

[You89] E. Yourdon. *Modern Structured Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

BIOGRAPHY

Han Oei is an assistant professor in the subdepartment of Information Systems at the University of Twente. From 1990 to 1993, after receiving his master degree in Computer Science (major in Business Informatics) from the University of Groningen, he has been involved in the Evolving Information systems project at the University of Nijmegen. He is a member of the IFIP WG 8.1 Task Group FRISCO investigating the conceptual foundations of information systems. His current research interests include harmonisation of information system methods (including techniques and concepts), and situational method engineering.