The Meta Model Hierarchy:
A Framework for Information Systems Concepts and Techniques

J.L.H. Oei[12]
hanoei@cs.kun.nl

L.J.G.T van Hemmen[2]
louisvh@cs.kun.nl

E.D. Falkenberg[12]
ef@cs.kun.nl

S. Brinkkemper[3]
sjbr@cs.utwente.nl

29 July 1992

Abstract

The numerous information systems design methods and specification techniques being proposed, can be compared on the basis of different criteria. In this paper a framework for ordering information systems modelling techniques based on meta models is proposed. This framework, called the Meta Model Hierarchy, positions each technique according to an analysis of the distinctions between the basic concepts, and the set of constraints defined on these concepts.

Relations between meta models are distinguished: partitioning, restriction, and degeneration. Each modelling technique is then positioned in the Meta Model Hierarchy according to these order relations. The positioning of Entity-Relationship Modelling, NIAM and Petri-nets in the Meta Model Hierarchy is discussed as an illustration of the use of the approach.

[1] The investigations were partly supported by the Foundation for Computing Science in the Netherlands (SION) with financial support from the Netherlands Organization for Scientific Research (NWO).

[2] Department of Information Systems, University of Nijmegen, Toernooiveld 1, NL-6525 ED Nijmegen, The Netherlands.

[3] Department of Computer Science, University of Twente, P.O. Box 217, NL-7500 AE Enschede, The Netherlands.

1 Introduction

In the last two decades, hundreds of information system design and development methods have been proposed (e.g. [OSV82]). Necessary and essential parts of such a method are one of more modelling techniques, which provide the possibility to model those aspects of "reality" which are relevant for the information system to be developed. Practioners as well as teachers in the information system field have been complaining for some time about the ever growing number of modelling techniques. Currently, the field appears from the outside as unsettled and even chaotic, suffering in particular from the YAMA Syndrome. This syndrome is due to the widespread attitude of researchers in the field to create Yet Another Modelling Approach, without addressing the question of its justification and its scientific added value in a sufficient way.

What are the differences between all these numerous modelling techniques? The simplest differences are terminilogical ones. For one and the same concept, different researchers might use different synonyms. A famous example is "entity" vs. "object". It is however also a tricky example, since different researchers are using the same term "entity" for different concepts. The homonym problem is even worse for the term "object".

Part of a modelling technique is a language for specifying the model developed for a specific application area, the application model, as well as for specifying operations to be performed upon that application model. Such a language is based on a system of concepts which is independent of the application case, and which will be called in this paper a meta model. A meta model is a set of basic concepts which are related to each other, the so-called concept structure, and a set of constraints determining the set of possible application models and the set of possible transitions between application models. The meta model of a language which not only allows to specify application models, but also to manipulate application models, also contains a set of specifications of types of operations which are performable upon any possible application model.

The choice of different basic concepts of the meta model is the most critical aspect. These differences are due to different classifications of, or views on the "phenomena","things" or "issues" of the "world". Should the world be classified into "entities" and "relationships", or "entities", "attributes" and "relationships", or "entities","events", "attributes" and relationships" or "states" and "transitions", or "actors", "actions", "objects of action" and "events", or "changers", "changer" and "changed items", etc.?

All the other aspects of a modelling technique, the constraining axioms of the meta model, the specifications of the types of operators, as well as the constructs of the specification languages, are dependent on those choices. For one choice of the concept structure of a meta model, differences in these dependent aspects result in different dialects of that meta model. For example, there are dialects of "entity-relationship" meta models which differ in the numbers of "entities" allowed within "relationships" (e.g. 2, 2 or more, 1 or more), or where different symbols are used to denote "entities" and "relationships" differently (e.g. boxes and diamonds vs. circles and boxes).

There are some bad reasons for all the mentioned differences. Competition among companies developing methods and tools for information system development is one of them. The attitude of too many researchers in the field leading to the already mentioned YAMA Syndrome is another one.

However, there are also some good reasons for the non-terminological differences. Different designers and users of an information system may view the "world" differently. Even more important is in this context, that for different kinds of information systems, different modelling techniques may be more or less suitable. For example, for a simple information system with a "snapshot" database, some dialect of an "entity-relationship" meta model may be convenient. In the case of a more sophisticated information system with history handling, trigger mechanisms and/or feedback mechanisms, such an "entity-relationship" meta model will be less suitable. It may be that some aspects cannot be specified at all, or only in an inconvenient and cumbersome way. For such a kind of application case, other meta models, like "time-oriented", "event-oriented" or "systems-dynamics-oriented" meta models may be much better suited. On the other hand, the use of those latter meta models for simple "snapshot" data base applications, would be an overkill.

Under these circumstances, the hope of some researchers that one day a unified and universal reference modelling technique for information systems will emerge, will probably never come true. A unified terminology might well be achievable, although it might be difficult. A universal reference meta model and specification language, however, is probably neither achievable nor desirable.

Nevertheless, the complaints of practioners and teachers in the information system field are justified. Due to the, most probably, unnecessarily large variety of modelling techniques, and due to the large numbers of unjustified differences between them, a lot of energy and brainpower is wasted when for new applications suitable modelling techniques must be selected, when for re-engineering of existing applications translations between modelling techniques must be provided, or when teachers must teach and students must learn about modelling techniques.

Various comparisons and frameworks of concepts have been proposed, but these are all of an informal nature (e.g. [Was83], [Ess86]). Comparison reports of information systems development methods were either based on laboratory experiments with emphasis on practical applicability of the methods [Flo86], or on an analysis of the method manuals ([BGKA89], [FGH91]). The reference framework of the CRIS taskgroup of IFIP WG8.1 ([OHM$^+$88]) presents an overview of the various specification techniques structured along the dimension of development phase (planning, analysis and design), versus modelling perspective (data, process and behaviour). The normative meta models provided a blue-print for a general purpose system of concepts and constructs for systems development. The notation of the meta models, however, does not allow for formal comparisons. The conferences and the work of the FRISCO taskgroup of IFIP WG8.1 ([FL89], [FRE92], [Lin90]) continue to concentrate on the need for clear and well-defined concepts in the information systems area, but the intermediate results indicate that a commonly accepted conceptual reference and terminology has not yet emerged.

Thus, there is an urgent need to get order into the chaos of modelling techniques on a formal basis. In this paper we try to tackle this challenge. We have organised it as follows. Section 2 elaborates on the criteria for the comparison of modelling languages, distinguishing expressive power and practicality. Furthermore the Meta Model Hierarchy is introduced recognizing these comparison criteria. The Meta Model Hierarchy is formally defined in section 3, distinguishing three construction operators for meta models (partitioning, restriction and degeneration). Section 4 illustrates our approach with the positioning of some well known modelling techniques in the Meta Model Hierarchy. We will conclude with some final remarks and a discussion of research options in section 5.

2 Comparison of modelling techniques

In this section it is discussed how modelling techniques can be compared, what meta models are, which role they play in relating modelling techniques, how they can be described, and how they form the basis for a framework for understanding and comparing modelling techniques, which is called the Meta Model Hierarchy.

2.1 Criteria for comparison

Information systems development methods consist of a set of modelling techniques. Modelling techniques are used for the specification of various aspects of the information system. One method can include more than one technique. The method ISAC ([LGN80]), for instance, uses among other modelling techniques A-graphs for describing activities, I-graphs for describing precedence relations between information sets, and C-graphs for describing the contents and structure of information sets. The method Information Engineering ([Mar88]) applies, among other things, Entity-Relationship Diagrams ([Che76]) for data modelling, Data Flow Diagrams ([GS79]) for process modelling, and Structure Charts ([YC79]) for the structure of programs. Each technique includes a modelling procedure. Procedures specify steps, algorithms, or heuristic guidelines for the modelling process itself. In this paper we abstract from the procedural part of a technique.

The number of modelling techniques which coexist in the information systems field has been growing to such an extent that it is easy to get lost in this jungle of modelling techniques. For this reason, the need emerges for a framework which can be the basis for understanding, and comparing these modelling techniques.

Modelling techniques can be compared on the basis of different criteria ([HO92]). One may be interested in *what* can be expressed, i.e. the expressive power or expressiveness of a technique. On the other hand, one may also be interested in *how* certain aspects of the universe of discourse are expresssed, i.e. the form or style of a technique. Even with the same technique we can formulate the same aspect of an application in different ways; with different techniques the formulations differ even more. The ‘how’ of a technique determines the practicality of a technique. Practicality involves pragmatics, and is dependent on the aspect taken in consideration.

Each technique involves a set of concepts (abstract syntax) together with a set of (sensory) symbols (concrete syntax) which represents these concepts. Examples of concepts are objects, constraints and operators. The concrete syntax for these concepts can be graphical-oriented (eg. circles, boxes, arrows etc.) or textual-oriented, or even be meant for another communication medium (e.g. auditive).

Techniques can have the same expressive power, but differ in practicality with respect to a certain aspect. The expressive power of a technique depends on the primitive concepts in the technique (and their inter-relationships); derived concepts can be defined in terms of these primitive concepts. The ‘how’ of a technique, however, depends on both concepts and symbols. Both the expressive power and the practicality of a technique determine the suitability of a technique for a particular application.

A technique is based on its meta model. The meta model of a technique is a model which describes those aspects which are independent of the domain (universe of discourse) that is described by that technique in a specific case. The description of the domain, which is dependent on a specific case, is called the application model. At any time, such an application model is an instantiation of the meta model of the technique that is used for describing that application model

([FOP91],[FOP92]).

In the comparison of methods or techniques, related work has been done (eg. [OSV82],[OHM$^+$88]). Our approach, however, differs from others in that we do not restrict ourselves to a sort of checklist of some global characteristics. In this paper both the expressive power and the practicality of techniques (with respect to convenience) are discussed in detail.

Expressive power
The expressive power of a modelling technique is a measurement of *what* can be described, ie. the possible universes of discourse that can be described. (For a formal definition of expressive power see eg. [HW91]). To show equal expressive power the notion of equivalence comes into the picture. Two models are equivalent if and only if there exists an isomorphism between these models. Techniques *t1* and *t2* have the same expressive power if any model described in technique *t1* has an equivalent model in *t2*, and vice versa.

Examples of two techniques which do not differ in their expressive power is a dialect of an object-role model in which only binary relationships are allowed, and a dialect of the same object-role model which allows n-ary relationships. These dialects have the same expressive power, because there exists a mapping from any binary model to a n-ary model, and vice versa ([e.g. [Hal89],[Fal88]).

Expressive power of techniques can be compared both on the application level and the meta level. Up to now, expressive power is mostly compared by defining transformations on the application level. That is, given a particular application model described in one technique transformations are defined to an equivalent application model in the other technique, and vice versa.

In [HO92] it is shown that it is also possible to compare the expressive power of different techniques by defining appropriate transformations on the meta level. To compare the expressive power of techniques, transformations are defined between the meta models of these techniques to a common (meta-)technique.

Practicality
One can also compare techniques on the basis of *how* these universes of discourse are described by these techniques. This latter question can be interesting for several mostly pragmatic reasons. There are different contexts in which the how-question can be put forward: convenience of techniques (how is a particular domain modelled?), the efficiciency of techniques (how can it be implemented?), and the learnability of techniques (how can it be communicated to others?). In this paper we restrict ourselves to practicality of modelling techniques with respect to their convenience for modelling different kinds of universes of discourse.

Expressive power can be determined in absolute terms. Convenience on the other hand is a relative notion, depending on the kind of domain to be modelled as well as to some extent on the preferences and background of the user of the technique.

For example, there may be a special-purpose technique which is highly convenient for that special purpose and for a specialist in that special field, but which has a low overall expressive power. There may be a general-purpose technique with a high expressive power, but which is only moderately well-convenient for many or most purposes. Very often, a trade-off between expressive power and convenience is required.

A technique can be more convenient than another for a particular application because of several reasons. One of the reasons is that a special-purpose technique has more specific concepts than a general-purpose technique. As techniques differ in their meta models, given a specific domain, techniques differ in those aspects of the domain which have to be specified explicitly in the application model, and those aspects which are incorporated (by the set of concepts) in the meta model.

An example is the usage of an Entity Relationship Modelling (ERM) technique which does not distinguish the time concept as a distinct concept, and a time-oriented ERM-technique which does make the distinction. In the former technique, time can be modelled as a particular entity in the application model, whereas in the latter technique there is no need to describe the concept of time in the application model because of the existence of a distinct (meta) concept ‘time’ in the meta model.

Thus, some techniques do not differ in the set of domains that can be described (influencing expressive power), but they differ in the extent that has to be specified explicitly in the application model (influencing convenience). In this case, we speak of the *Principle of Conservation of Information.*

In figure 2.1 it is illustrated that different modelling techniques having the same expressive power describe the same domain in a different way. To which extent the relevant elements or phenomena have to specified in the application model explicitly, depends on (the specificness of) the meta model of a modelling technique.

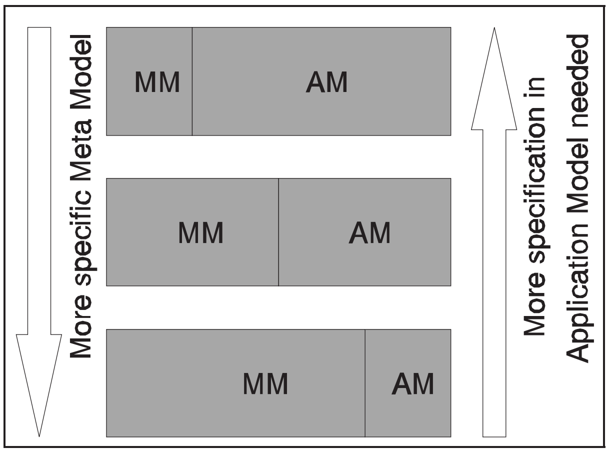


Figure 2.1 Principle of Conservation of information

2.2 Ordering modelling techniques: the Meta Model Hierarchy

The Meta Model Hierarchy Framework that is proposed in this paper orders modelling techniques on the basis of their meta models (see fig 2.2). Both expressive power and convenience play a crucial role in this Meta Model Hierarchy. The order on meta models is based on certain elementary relations defined between meta models, viz. partitioning, restriction and degeneration of the concept structure of meta models. Each relation has a corresponding constructive operator which resp. partitions, restricts, or degenerates a meta model, and an inverse destructive operator which creates out of a partitioned, restricted, and degenerated model its original. These relations defined between meta models and their corresponding operations are formally introduced in section

3.

The first one of these elementary operators partitions a concept in a meta model in more specialized concepts. The meta model resulting from this operation is a child meta model of the original one. It will be obvious that this partitioning operation influences convenience for modelling certain aspects in certain application domains. Note, however, that conform the Principle of Conservation of Information, the expressive power is not necessarily influenced by this operation. With respect to expressive power, it can be said that all operators defined on meta models, and which constitute the Meta Model Hierarchy, guarantee that everything that can be expressed in a technique based upon a specific meta model, can also be expressed in a technique based upon any of its parent meta models in the Meta Model Hierarchy. This property is called upward compatibility of meta models in the Meta Model Hierarchy.

It should be noted that the framework does not (yet) provide an overall solution for the comparison of the expressive power of an arbitrary set of techniques or (semantical) transformations among these techniques. Also for incorporating the manipulation operators of a technique in the comparison, the framework has to be extended. However, in many cases useful conclusions can be drawn from the Meta Model Hierarchy with respect to both the notion of expressive power and convenience of techniques.

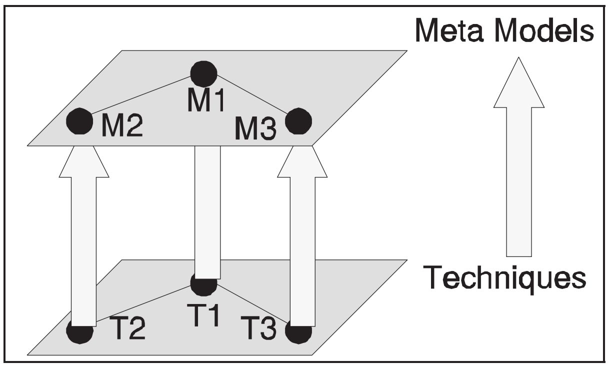


Figure2.2 Relating techniques
by ordering their meta models

Nowadays, meta modelling is a commonly accepted means for information systems research. An overview of applications can be found in [Bri90] and in [TL92]. The ordering of meta models in the Meta Model Hierarchy allows to utilize the mutual dependencies between the meta models. Among other things, the Meta Model Hierarchy can be used for the following purposes:

a) Selection of suitable modelling techniques

Given the characteristics of an application domain, the Meta Model Hierarchy may serve to select the proper modelling technique. The characteristics of the domain are matched with the properties of the techniques. The properties of different techniques can be identified by positioning and comparing the considered techniques in the Meta Model Hierarchy.

b) Transfer between modelling techniques

In a multi-specification project or in a project where the models of former projects are reused, the need for transformation of specifications denoted in the one modelling technique to the other is apparent. The Meta Model Hierarchy defines the relation between the techniques which is taken as the core of the transformation procedure.

c) Support of second-order evolution

First-order evolution is characterized by systems in which the meta model and the corresponding specification language(s) are stable, but each and every aspect of the application model can change, including application-specific types, laws, and rules (e.g. schema evolution) without the need to install a new system and thereby interrupt the primary processes in the organization. A framework for information systems allowing for first-order evolution is described in [FOP91] and [FOP92]. Second-order evolvability is particulary important for large organizations with various sorts of applications, and where also new sorts of applications become necessary from time to time. If such an organization applies a Meta Model Hierarchy of the meta models underlying the applications, specific operations can be performed on the existing Meta Model Hierarchy, to cater for a new sort of application, or even for new aspects within an existing application model. All the old application models or old parts of application models, and the operations thereon will continue to work, and the work in the organization will not be interrupted.

d) Method Engineering

The construction of project specific methods, so called method engineering, requires to have meta models of development methods and of modelling techniques available in a method base. The Meta Model Hierarchy provides a structuring mechanism for this method base, and allows the method engineer to navigate through the space of techniques present and to select one or more techniques for inclusion in the method for the project.

2.3 Meta models and their structure

A meta model used for specifying application models only (not for manipulating them), is composed of a concept structure and a set of constraints. An instantation of such a meta model, i.e. a particular application model, should be conform the concept structure, and should satisfy the set of constraints.

The concept structure is composed of a set of meta concepts, a set of relationships between these meta concepts, which in turn are meta concepts themselves, and a specialization (or subtype) hierarchy of these meta concepts, which is explained further on. (The concepts in the concept structure of a meta model are called meta concepts to distinguish them from concepts in the application model.)

Let us take the data modelling technique NIAM ([NH89], [Win90]) as an example. The concept structure of the meta model of NIAM includes the meta concepts of which object, role, entity, label, fact and reference are sample elements. An example of a relationship between these meta concepts, is the object-role relationship (predicator) which represents that some objects play certain roles. A set of these object-role relationships can be composed to an object which is called a composed object (often called also a relationship). Such a composed object can be either a fact or a reference. This dichotomy is reflected in the specialization hierarchy, which also gives the information that an object which is not a composed object, i.e. it is a simple object, can be either an entity or a label. The concept structure of NIAM plus some constraints are denoted in figure 2.3 in a graphical notation of NIAM itself.

A concept structure, which is part of the meta model of a modelling technique, is used as a frame here for a particular application model described in that technique. A state of the application model corresponds to a so-called instantation or population of the concept structure, and vice versa. In our approach, a population of a concept structure is a value assignment to the meta concepts, conforming to the structure defined by the relationships between these meta concepts, and respecting the specialization hierarchy.

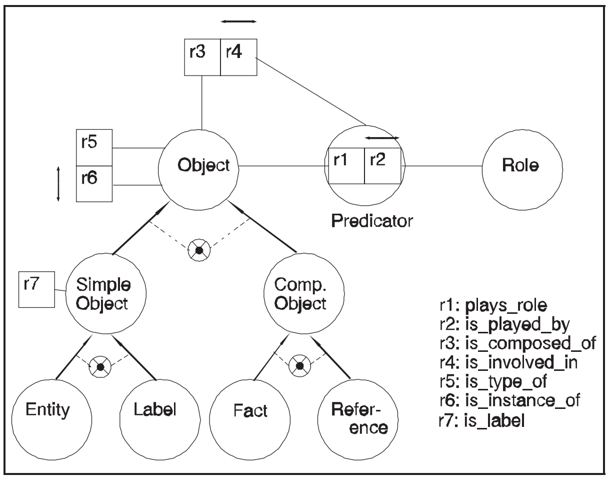
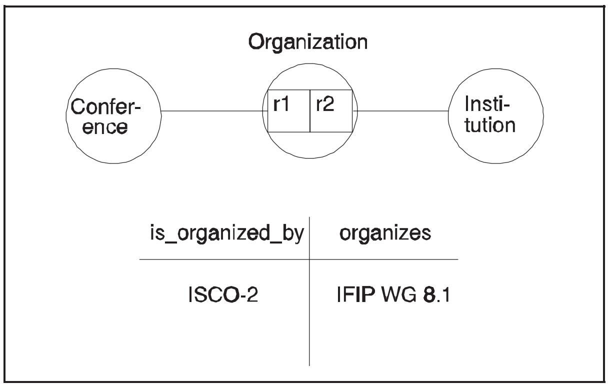


Figure 2.3 Meta model of NIAM

Let us take the IFIP-case concerning conferences ([OSV82]) as a specific case or application. A possible description of part of this case is presented in a graphical NIAM-notation in fig. 2.4.



is_organized_by	organizes
ISCO-2	IFIP WG 8.1

Figure 2.4 Application model in NIAM

It is shown that this application model is an instantiation of the meta model of NIAM as represented in fig. 2.3. The population of the concept-structure of NIAM, which represents the application model of the IFIP-case of fig. 2.4 is as follows:
(Note that for simplicity in this example, all simple objects are considered to be entities, and all composed objects to be facts. Labels and references are omitted in this example.)

Object is_type_of	*Object is_instance_of*
Conf	*ISCO-2*
Inst	*IFIP WG 8.1*
Org(Conf,Inst)	*Org(ISCO-2, IFIP WG 8.1)*

Object plays_role	*Role is_played_by*
Conf	*is_organized_by*
Inst	*organizes*
ISCO-2	*is_organized_by*
IFIP WG 8.1	*organizes*

Object is_composed_of	*Predicator is_involved_in*
Org(Conf, Inst)	*Conf is_organized by*
Org(Conf, Inst)	*Inst organizes*
Org(ISCO-2, IFIP WG 8.1)	*ISCO-2 is_organized by*
Org(ISCO-2, IFIP WG 8.1)	*IFIP WG 8.1 organizes*

A set of constraints, which is the other part of the meta model of a modelling technique, exclude all populations which do not correspond to a correct application model described in that modelling technique.

In NIAM, for example, populations of the concept structure are excluded which assign values to the meta concept 'object', but in which these values are not involved in at least one value assignment to the meta concept 'fact'. For the IFIP-case, it means that the objects 'Conference' and 'Institution' must be involved in at least one fact. This constraint is due to the fact that object types may not be unconnected in a datamodel in NIAM.

To denote meta models many languages can be used. In this paper we have chosen a set-theory-based language to denote meta models, because it allows for more compact and comprehensible definitions than eg. a predicate calculus. This meta language is an adjusted version of the Predicator Model technique ([BHW91]), a modelling technique for object-role models. An essential difference is that the language defined here is used for describing meta models, whereas the Predicator Model technique is used for the specification of application models based upon the schema vs. instance dichotomy. In other words, the language which is defined below is situated one level up in the meta level hierarchy ([BF91]), and is therefore called a meta language. Note that the constructs of the meta language are prefixed by 'meta' to distinguish them from constructs in a meta model of a specific modelling technique.

Definition 2.1
Let MM be the set of all possible meta models,
mm $\in$ MM has the following structure $\langle S,C \rangle$
where S = $\langle P,O,F,Spec,Base \rangle$ is a meta concept structure,
where P is a finite set of meta predicators,
O is a set of meta object types,
F is a partition of the set P. The elements of F are called meta fact types ($F \subseteq O$).
Spec $\subseteq A \times O$ is a partial order on meta object types, capturing specialisation, and where
$A \subseteq O \setminus F$ is called the set of meta atomic objects.
Base : $P \rightarrow O$ is a function. The base of a meta predicator is the meta object part of that meta predicator.
$C \subseteq \Gamma(S)$ is a set of meta constraints, where $\Gamma(S)$ is the set of all possible meta constraints on S

A population Pop, i.e. an instantiation of a meta model mm $\in$ MM, constituting a particular *application model, should be conform the meta concept structure, and satisfy the set of meta constraints:*

$mm = \langle S,C\rangle$

$IsPop(mm,Pop) \equiv IsPop(S,Pop) \wedge \forall c \in C\ [Pop \models C]$

$(Pop: O \rightarrow Powerset(\Omega)$, *where* Ω *is the set of all possible instances)*

For axioms on this structure we refer to [BHW91].

Note that in the meta language we speak about specialization in general, whereas in the Meta Model Hierarchy specializations are restricted by some additional constraints (ie. mutually exclusive and collectively exhaustive). In that case we will speak of partitioning.

In order to simplify the definitions in the following section some auxiliary functions are defined:

Definition 2.2

$Struct(\langle S,C\rangle)=S$

$Constr(\langle S,C\rangle)=C$

$Pred(\langle\langle P,O,F,Spec,Base\rangle,C\rangle)=P$

$Object(\langle\langle P,O,F,Spec,Base\rangle,C\rangle)=O$

$Fact(\langle\langle P,O,F,Spec,Base\rangle,C\rangle)=F$

$Spec(\langle\langle P,O,F,Spec,Base\rangle,C\rangle)=Spec$

$Base(\langle\langle P,O,F,Spec,Base\rangle,C\rangle)=Base$

To illustrate this language for describing meta models, the meta model of NIAM as graphically denoted in fig. 2.3 is described in the meta language.

Let x be the meta model of NIAM, then
x = ⟨Struct(x),Constr(x)⟩,
where
Struct(x) = ⟨Pred(x),Object(x),Fact(x),Spec(x),Base(x)⟩, where
Pred(x) = {plays_role, is_role_of, is_involved_in, is_composed_of, is_type_of, is_instance_of, is_entity},
Object(x) = {Object, Role, Simple_object, Composed_object, Entity, Label, Fact, Reference, Predicator, Composition, Typing},
Fact(x) = {Predicator, Composition, Typing},
Spec(x) = {⟨Simple_Object,Object⟩, ⟨Composed_object,Object⟩, ⟨Entity,Simple_object⟩, ⟨Label,Simple_Object⟩, ⟨Fact,Composed_Object⟩, ⟨Reference, Composed_object⟩},
Base(x) = {⟨Object,plays_role⟩, ⟨Role,is_role_of⟩, ⟨Object,is_base_of⟩, ⟨Predicator,is_involved_in⟩, ⟨Object,is_composed_of⟩, ⟨Object,is_instance_of⟩, ⟨Simple_Object,is_entity⟩}, and
Constr(x) = ⟨unique(is_played_by), unique(is_involved_in), unique(is_instance_of), total(Simple_Object,Composed_Object), total(Entity,Label), total(Fact,Reference), exclusion(Simple_Object,Composed_Object), exclusion(Entity,Label), exclusion(Fact,Reference)⟩

In the following section the meta language defined here is used to define the Meta Model Hierarchy.

3 The Meta Model Hierarchy

In this section the Meta Model Hierarchy is presented. In subsection 3.1 relations and corresponding operators on meta models are defined, which constitute a hierarchy of meta models. In section 3.2 alternative procedures for the construction of this Meta Model Hierarchy are given.

3.1 Ordering meta models

The Meta Model Hierarchy is based on relations and corresponding operators on meta models.
Three relations on meta models are defined: partitioning (Part), restriction (Restr), and degeneration (Deg), which will be explained in the coming subsections. It can easily be proved that the relations defined are irreflexive, asymmetric, and transitive. Furthermore, the intersection of the relations is empty. From these properties it is concluded that the relations constitute a directed acyclic graph between meta models. As in most practical cases only one root is considered, this directed acyclic graph between meta models is called the Meta Model Hierarchy.

Definition 3.1
The Meta Model Hierarchy MMH, which is a directed acyclic graph, has the following structure:
MMH= $\langle \mathrm{MM}, <_{MMH}, Part, Restr, Deg \rangle$
where MM *is the set of possible meta models,*
$Part \subseteq \mathrm{MM} \times \mathrm{MM}$ *is a relation on meta models, capturing partitioning,*
$Restr \subseteq \mathrm{MM} \times \mathrm{MM}$ *is a relation on meta models, capturing restriction,*
$Deg \subseteq \mathrm{MM} \times \mathrm{MM}$ *is a relation on meta models, capturing degeneration,*
$<_{MMH} \subseteq \mathrm{MM} \times \mathrm{MM}$ *is a relation on meta models, defined as follows:*
$x <_{MMH} y$ *iff (x Part y) or (x Restr y) or (x Deg y).*

The Meta Model Hierarchy relations are based on elementary (constructive and destructive) operators on the meta model: the partitioning of meta concepts, the addition of constraints, and the elimination of meta concepts, and their inverse counterpart.

3.1.1 Partitioning applied to meta models

Different modelling techniques are more or less convenient for modelling different kinds of application domains. One reason for this is that modelling techniques differ in the specialization of meta concepts in their concept structure, which is caused by the fact that different modelling techniques apply a different set of subdivisions to their basic concepts. We will restrict ourselves to subdivisions which are mutually exclusive and collectively exhaustive, i.e. the specializations are partitionings. If the set of partitionings distinguished in a meta model is a superset of the set of partitionings distinguished in another meta model, the first meta model is said to be more partitioned than the latter.

Recall from section 2.2 that the concept structure of a meta model contains a set of meta concepts - consisting of meta fact types and meta atomic object types - and a specialization hierarchy of these meta concepts. The basic operator which corresponds to the construction of a partitioning of a meta model is the addition of a subdivision. This addition of a subdivision is performed by partitioning the possible set of instantiations of meta concepts from the concept structure. A subdivision into n parts is also called a n-chotomy, and a subdivision into two parts a dichotomy. Note that a n-chotomy can be treated as a sequence of n-1 dichotomies. Therefore, we do not lose generality by only treating dichotomies in our examples.

In NIAM, for example, one of the dichotomies applied is the distinction of simple objects into entities and labels. As a result, the meta model of NIAM partitions the meta concept ‘simple object’ into the meta concepts ‘entity’ and ‘label’. The meta model of NIAM can be seen as a partitioning of an (fictive) object-role model which does not make a distinction between entities and labels.

Definition 3.2
The partitioning relationship is defined as an irreflexive, asymmetric, and transitive relation Part between meta models (Part $\subseteq$ MM *x* MM*), with the convention that x Part y is interpreted as: x is a partitioning of y.*

$$\begin{aligned} x\ Part\ y \ &\equiv x\ Part'\ y \vee \\ &\quad \exists z\colon z\ Part'\ y \wedge x\ Part\ z \end{aligned}$$

where

$$\begin{aligned} x\ Part'\ y \equiv \quad & Spec(x) \supset Spec(y) \wedge \\ & Object(x) \supset Object(y) \wedge \\ & Constr(x) \backslash Constr(y) = \{total(Object(x) \backslash Object(y)), \\ & \qquad exclusion(Object(x) \backslash Object(y))\} \end{aligned}$$

where total and exclusion as defined in [BHW91].

A meta model *x* is said to be a partitioning of meta model *y* iff the concept structure of meta model *x* can be derived from the concept structure of meta model *y* by the partitioning of one or more concepts. Each partitioning is mutually exclusive and collectively exhaustive subdivision, i.e. every instance in the population of the partitioned concept has to be in the population of precisely one of its parts. This is imposed by the total and exclusion constraint added to the specialization. For example (see fig 3.1), the partitioning of a meta concept A into meta concepts B and C in meta model *Y* results in a meta model *X* which is a partitioning of meta model *Y*. Note that an addition of a meta fact type to a meta model is derivable from partitionings of a more general meta fact type, which is a meta concept itself. (The partitioning of meta fact types is illustrated in section 4.3.1.)

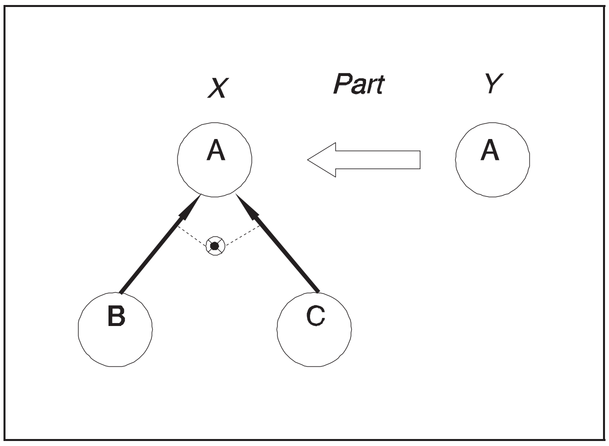


Figure 3.1 X is a partitioning of Y

3.1.2 Restriction applied to meta models

Some modelling techniques are more constrained than others. The binary dialect of NIAM, for example ([VB82]), allows only facts between two objects, whereas the n-ary dialect of NIAM ([NH89]) does not restrict the number of objects involved in a fact. We then can say that the meta model of the binary dialect of NIAM is a restriction of that of the n-ary dialect. A restriction restricts the possible population of the meta model, and therefore the possible application models. Note however, that a restriction does not necessarily imply that the expressive power is influenced. In the example just shown, any application model in the n-ary NIAM-dialect can be transformed into a binary variant without losing information, and vice versa.

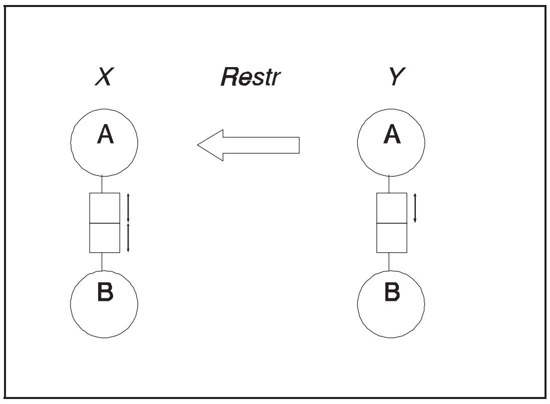


Figure 3.2 X is a restriction of Y

Definition 3.3
The restriction relationship is defined as an irreflexive, asymmetric, and transitive relation Restr between meta models (Restr $\subseteq$ MM *x* MM*), with the convention that x Restr y is interpreted as: x is a restriction of y.*

$x \; Restr \; y \equiv IsPop(x,Pop) \Rightarrow IsPop(y,Pop) \wedge x \neq y$

Restriction of meta models is obtained by adding constraints to the set of constraints of a meta model. Note that restriction only considers addition of constraints which necessarily do constrain the possible population of that meta model.

3.1.3 Degeneration applied to meta models

Sometimes there are meta concepts in the concept structure of a technique, which could only be defined by the grace of the existence of other meta concepts. However, not all of these defining meta concepts need to be explicitly present in the meta model of a technique. For example, the meta concept ‘Transition’ in Petri Nets is derived from relationships which represents the time of birth and time of death of certain objects. However, in Petri-Nets, the meta concept ‘Time’ is not modelled explicitly. The meta model of Petri-nets is now considered to be a degeneration of a meta model which does include the meta concept ‘Time’. Note that some meta concepts which were derivable before, cannot be derived anymore after a degeneration, thus becoming essential meta concepts. In the example of Petri-nets, the derivable meta concept ‘Transition’ becomes essential after the degeneration of the meta concept ‘Time’. (For a more elobarate discussion of the degeneration of time in Petri-nets: see section 4.3.3.)

Definition 3.4
The degeneration relationship is defined as an irreflexive, asymmetric, and transitive relation Deg between meta models (Deg $\subseteq$ MM *x* MM*), with the convention that x Deg y is interpreted as: x is a degeneration of y. A weak and a strong variant of degeneration will be given.*
$x \; Deg \; y \equiv x \; Deg_W \; y$ *or* $x \; Deg_S \; y$

In the weak variant of degeneration meta concepts which are generalizations of other meta concepts are eliminated. Note that, as all partitionings introducing new meta concepts are collectively exhaustive, the possible populations of meta concepts which are leaves in the specialization hierarchy of meta concepts are not changed by this weak degeneration.

Definition 3.5
Weak degeneration is defined as follows:
$x\ Deg_W\ y \equiv \quad Object(x) \subset Object(y) \wedge$
$\qquad \forall o \in Object(y) \backslash Object(x): not(Leaf(o,y))$
where $Leaf(o,x) \equiv o \in Object(x) \wedge \forall \langle o_j, o_i \rangle \in Spec(x) \Rightarrow o_i \neq o$

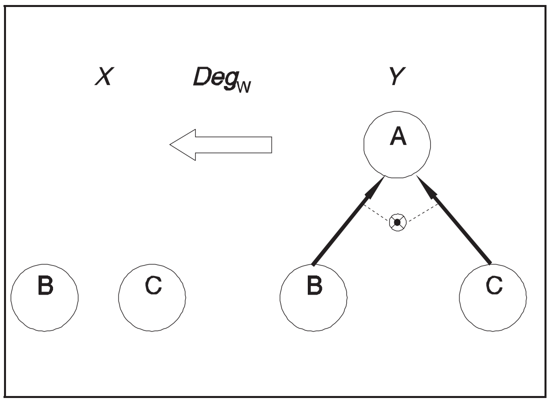


Figure 3.3 X is a weak degeneration of Y

If meta model *x* is obtained from meta model *y* by eliminating meta concept which are not partitioned further in meta model *y*, i.e. non-leaf meta concepts are removed, then meta model *x* is said to be a strong degeneration of meta model *y*. For example in figure 3.4, if one of the specializations of meta concept A in meta model *Y* is eliminated, it results in a meta model *X* which is a strong degeneration of meta model *Y*. The elimination of the time-concept in Petri-Nets is an example of a strong degeneration.

Definition 3.6
Strong degeneration Deg_S *is defined as follows:*
$x\ Deg_S\ y \equiv \quad Object(x) \subset Object(y) \wedge$
$\qquad \forall o \in Object(y) \backslash Object(x): Leaf(o,y)$
where $Leaf(o,x) \equiv o \in Object(x) \wedge \forall \langle o_j, o_i \rangle \in Spec(x) \Rightarrow o_i \neq o$

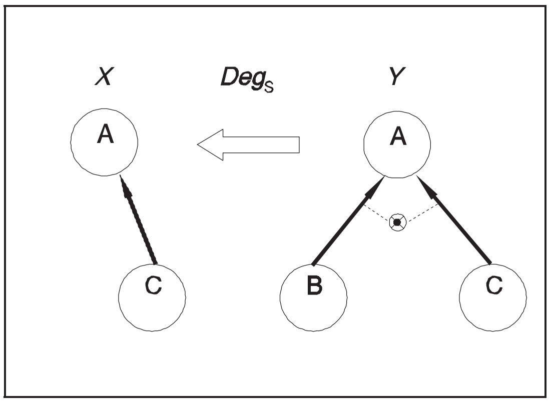


Figure 3.4 X is a strong degeneration of Y

3.2 Constructing the Meta Model Hierarchy

The Meta Model Hierarchy relates meta models of modelling techniques to each other. Due to the evergrowing amount of coexisting techniques nowadays, it is unrealisitic to construct a complete Meta Model Hierarchy. In practice, however, we are interested in relating and comparing a specific (small) set of techniques. In this section both a bottum-up and a top-down procedure are given to construct that part of the Meta Model Hierarchy, being sufficient to position some modelling techniques in relation to each other.

<u>Bottum-up procedure:</u>

Step 1a:	Analysis of the considered techniques
Step 1b:	Construction of the meta models of the techniques
Step 2a:	Analysis of the meta models for similarities in the concept structure and constraint set: - identification of homonyms and synonyms - generalisation of concepts
Step 2b:	Construction of intermediate meta models which can be derived from the existing set of meta models by appropriate constructive and destructive operators (i.e. adding/removing of partitionings, adding/removing of constraints, or expliciting/ impliciting concepts).
Step 2c:	Relating the meta models of step 1 and 2b in the Meta Model Hierarchy applying partitionings, restrictions and degenerations as its basis.
Step 3:	Repeat step 2a-c until all meta models of step 1 are related to each other.

In this bottum-up procedure, the meta models of the techniques to be considered, are taken as a starting point. Intermediate meta models are then constructed from these meta models until a part of the Meta Model Hierarchy is constructed in which all meta models are connected (related). The top of this hierarchy will be called the root meta model. The result of these steps is a connected part of the Meta Model Hierarchy, having a root from which the meta models of step 1 can be derived by a sequence of partitioning, restriction and degeneration operations.

If, a priori, a specific root meta model for the Meta Model Hierarchy is assumed, a top-down procedure can be adopted. In this procedure, a hierarchy is constructed by deriving (intermediate) meta models from the root meta model. The extension of this hierarchy will stop at the moment that the meta models to be positioned become all part of the hierarchy.

<u>Top-down procedure:</u>

Step 1:	Choice of the root meta model for the part of the Meta Model Hierarchy to be constructed.
Step 2a:	Analysis of the considered techniques
Step 2b:	Construction of the meta models of the techniques
Step 3a:	Analysis of the root meta model and the meta models for similarities in the concept structure and constraint set:

- identification of homonyms/synonyms
- partitioning of concepts (adding dichotomies)

Step 3b: Construction of intermediate meta models which can be constructed from the root by partitioning, restriction and degeneration operations.

Step 3c: Relating the root meta model and the meta models of step 3b in the Meta Model Hierarchy.

Step 4: Repeat step 3a-3c until all meta models of step 2 are related to each other. In other words, a connected part of the Meta Model Hierarchy is constructed, having a root from which the meta models of step 2 can be derived by a sequence of partitioning, restriction and degeneration operations.

The basic difference between these procedures is that the top-down procedure starts from a chosen root meta model and partitions meta concepts until the meta concepts of all meta models of the considered techniques are reached, whereas the bottum-up procedure take the meta models of the techniques as a starting point and construct new meta models by generalizing meta concepts, until a common meta model is reached. The choice of the procedure may depend on the presence of a suitable root meta model, and the number of meta concepts distinguished in the set of techniques to be considered. In case of a large number of meta concepts it seems to be preferable to look for generalizations of these meta concepts first, thus following the bottum-up procedure. In case of a limited number of distinct meta concepts, the top-down procedure can be taken.

In the following section, three well-known modelling techniques are positioned in the Meta Model Hierarchy. As simplified meta models are considered with a limited number of meta concepts, the top-down procedure is chosen, assuming a simple object-role model as the root meta model.

4 The Meta Model Hierarchy: an illustrative example

The intention of this section is to illustrate the practical use of the Meta Model Hierarchy. The techniques Entity-Relationship Modelling (ERM), NIAM and Petri-nets will be positioned in the Meta Model Hierarchy following the top-down approach of subsection 3.2. The part of the Meta Model Hierarchy that will stepwise be constructed in this section is illustrated in figure 4.1. Note that some intermediate meta models (marked by an asterisk) are constructed to relate the (simplified) meta models of ERM, NIAM and Petri-nets. Each node in the hierarchy corresponds to a meta model plus the set of meta models which can be derived from it. Meta models on one path in the hierarchy are upward compatible.

In this section the meta models are denoted in the graphical notation of NIAM. Note that to give a more complete description of the meta models and their relationships a more expressive language should be taken, eg. the meta-language of subsection 2.2 or a first-order predicate calculus. As the intention of this section is just to be illustrative, the graphical NIAM-notation suffices for our purpose.

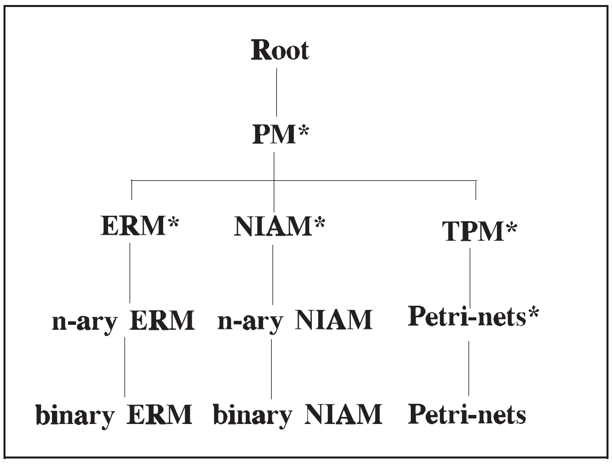


Figure 4.1 The Meta Model Hierarchy
relating ERD, NIAM and Petri-nets

4.1 Choice for the root meta model

To construct a part of the Meta Model Hierarchy which relates the meta models of the techniques to be considered, a particular meta model is chosen to become the root of this part of the hierarchy. The choice for the root has to be a meta model from which it is possible to end up in all the target meta models by an appropriate set of partitionings, restrictions and/or degenerations. Note that the root of the Meta Model Hierarchy has to be a meta model which has the same or more expressive power than all the other meta models of the hierarchy.

In this example, the root that is chosen is a simple object-role model which is able to describe relationships (facts). They are described by composed objects being objects which are composed of at least one object-role pair. Objects which are not composed are somple oblects. This root suffices to relate the (simplified) meta models of the techniques which will be taken into consideration in this example. If we would consider complete meta models with all kind of constraints, a richer meta model had to be chosen, eg. the meta model had to be extended with eg. a predicate calculus.

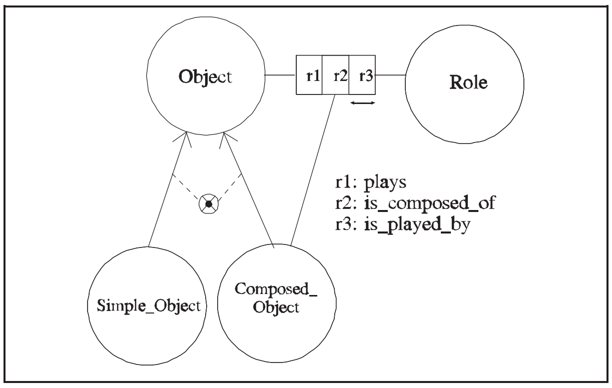


Figure 4.2 The root meta model

4.2 Analysis of the techniques to be considered

The construction of meta models results from an thorough analysis of the techniques to be meta modelled. It is the responsibility of the meta-modeller to construct good meta models, because it is the quality of these meta models which determines the value of the outcome of the Meta Model Hierarchy. In this section the (simplified) meta models of the techniques which will be related, are given. Note that the type-instance dichotomy we identify in our meta models, are often ignored.

4.2.1 A meta model for NIAM

NIAM is an object-role data modelling technique which is based on elementary sentences. One of the characteristics of NIAM is that it distinguishes abstract objects (entities) which are non-lexical from concrete objects (labels) which are denotable. Furthermore, different dialects of this technique allow or disallow objectification, i.e. relationships which are treated as objects themselves. Finally, some dialects are constrained to binary fact types ([VB82]), whereas other allow n-ary fact types (Nijssen89]). In this example a dialect of NIAM is chosen which allows objectification. The simplified meta model of this NIAM can be seen in figure 4.3. Note that this meta model allows n-ary fact types, unless a constraint is added which restricts the possible objects involved in a composed object.

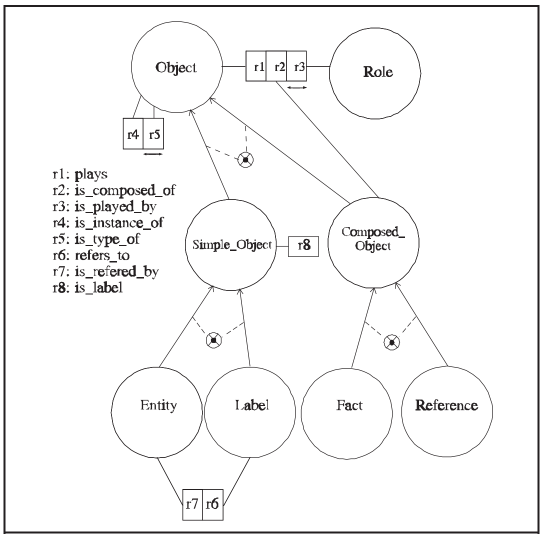


Figure 4.3 A meta model of NIAM

4.2.2 A meta model for Entity Relationship Modelling (ERM)

Another popular data modelling technique is Entity Relationship Modelling. This technique distinguishes entities, relationships, and attributes. Entity Relationship Modelling has even more dialects and extensions. We consider here the original one of [Che76] which requires binary relationships and disallow objectified relationships, i.e. relationships which can also be entities themselves. The metamodel of ERM which is a simplification of the meta model presented in [Bri90] can be found in figure 4.4. The considered meta model abstracts from cardinality constraints and identification structures.

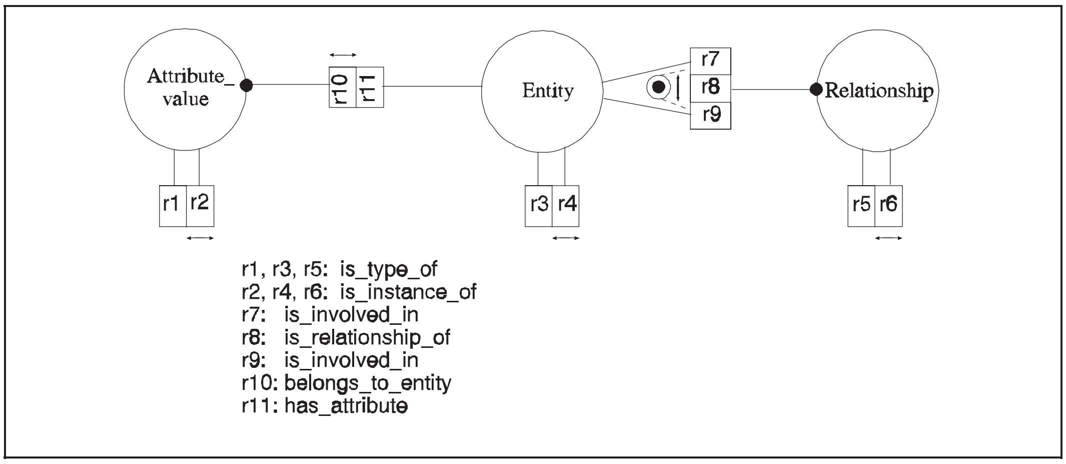


Figure 4.4 A meta model of (binary) ERD

4.2.3 A meta model for Petri-Nets

Petri-Nets ([Petterson81]) is a state-transition oriented modelling technique which is able to model activities and their inter-relationships, in particular concurrency and casualty. The meta model of Petri-Nets which is considered in this example is given in figure 4.5. Note that we abstract from some constraints, eg. firing rules for the firing of transitions.

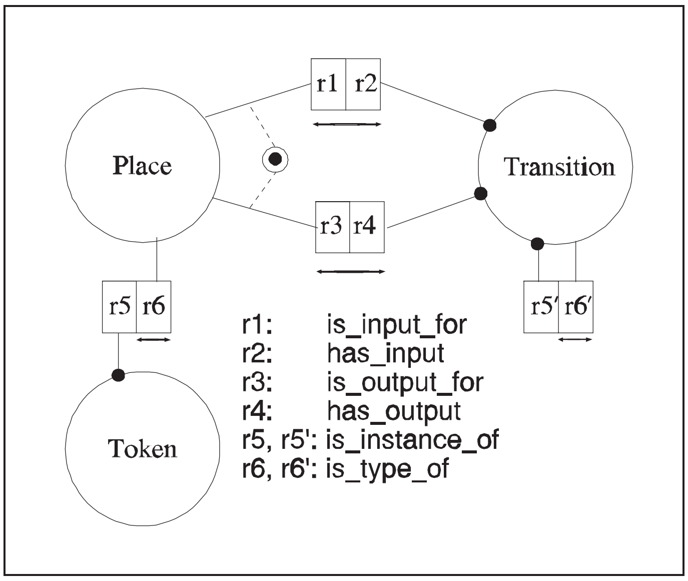


Figure 4.5 A meta model of Petri-nets

4.3 Constructing the Meta Model Hierarchy

In order to relate the meta models of NIAM, ERM, and Petri-Nets in the Meta Model Hierarchy the following procedure is chosen. First of all, the meta model of NIAM is related to the root meta model by a sequence of partitionings, restrictions, and degenerations. In this process some intermediate meta models are constructed. After the construction of this path, the meta model of ERM is connected by applying Meta Model Hierarchy operators on one of these intermediate meta models on this path. Finally, the meta model of Petri-Nets is incorporated in the Meta Model Hierarchy by a set of appropriate Meta Model Hierarchy operations.

4.3.1 Incorporating NIAM

NIAM, like most other modelling techniques, makes distinctions between types and instances. This dichotomy is added to the root meta model by partitioning the meta concept Object into Object_Type and Object_Instance. The resulting meta model is called PM* (see figure 4.6), because it is closely related to the Predicator Model as defined in [BHW91]. The relationship introduced between Object type and Object instance can be derived from the additional partitionings of the meta concepts Role and Composed_Object by identifying the Role instances used in a Composed_object. Subtype defining rules are also defined in terms of the role instances. Representation of partitionings of the meta concept Role and Composed_Object is most of the time ommited in the following, to reduce complexity of the figures. In the following, whenever a n-chotomy is added, only a further partitioning of the meta concept Object is represented together with the relationship(s) between the new concepts. As already stated, these relationships are derivable from a further partitioning of the meta concepts Role and Composed_Object.

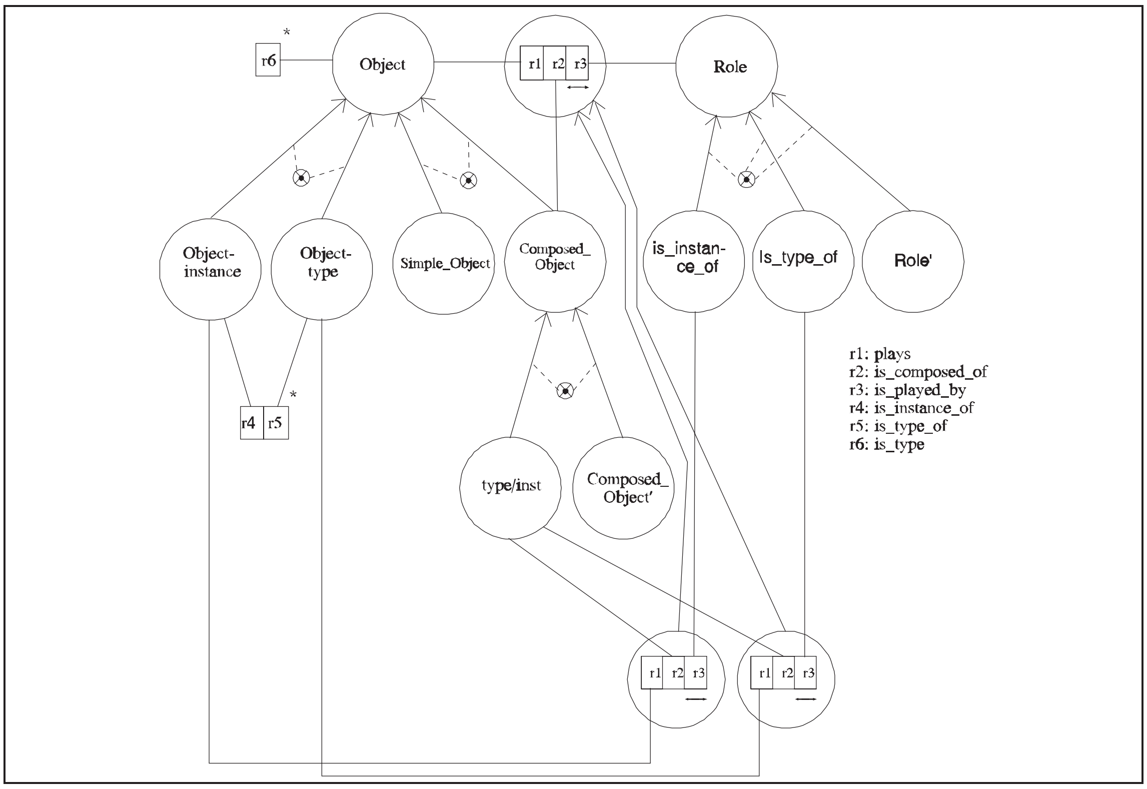


Figure 4.6 The intermediate meta model PM*

As stated before, in NIAM a distinction is made between entities and labels which are denotable and refer to these entities. These relationships between entities and labels are called references, whereas relationships between entities are called facts. These distinctions are introduced by a partitioning of the meta concept Simple_Object into Entity and Label, and a partitioning of the meta concept Composed_Object (Relationship) into the meta concepts Fact and Reference. The meta model resulting from these partitionings can be seen in the simplified notation of figure 4.7.

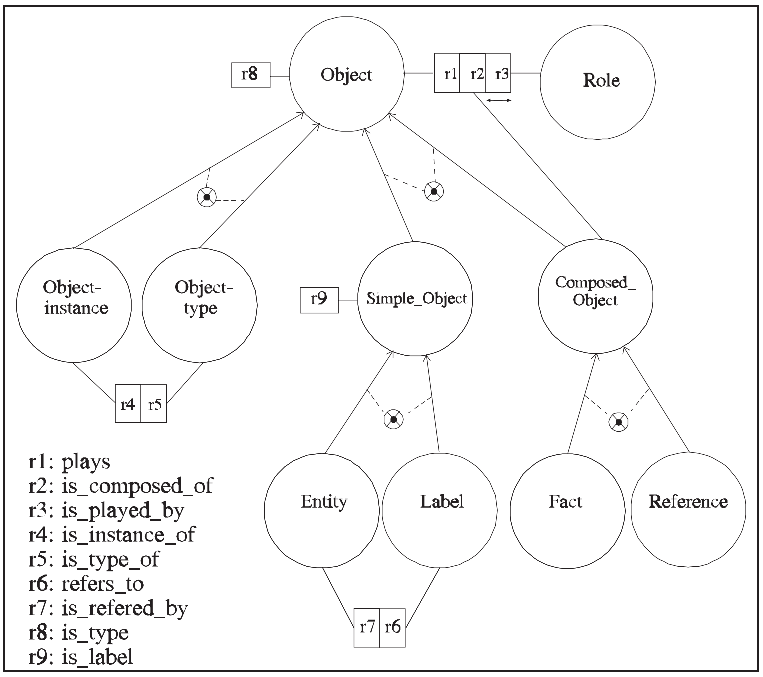


Figure 4.7 The meta model of NIAM*

By a (weak) degeneration of the meta concepts Object_Type and Object_Instance the meta model of NIAM, as represented in figure 4.2, can be derived. Note that a restriction of this meta model by adding the constraint that only two objects can be involved in a Composed_Object, results into the meta model of a binary dialect of NIAM.

4.3.2 Incorporating Entity Relationship Modelling

In contrast to NIAM, Entity Relationship Modelling distinguishes attributes. These attributes can be considered as relationships characterizing the important objects, which are the entitities. (Note that this is different from the dichotomy entity vs. label in NIAM.) From the intermediate meta model PM* of figure 4.6 a meta model ERM* can be constructed by partitioning Composed_Object into Attribute and Relationship.

The meta model of ERM* allows n-ary relationships and objectifications. Restricting this meta model by adding constraints, a meta model can be obtained which disallows objectfication and only allows binary relationships. After (weak) degeneration of some concepts the target meta model of figure 4.3 can be derived.

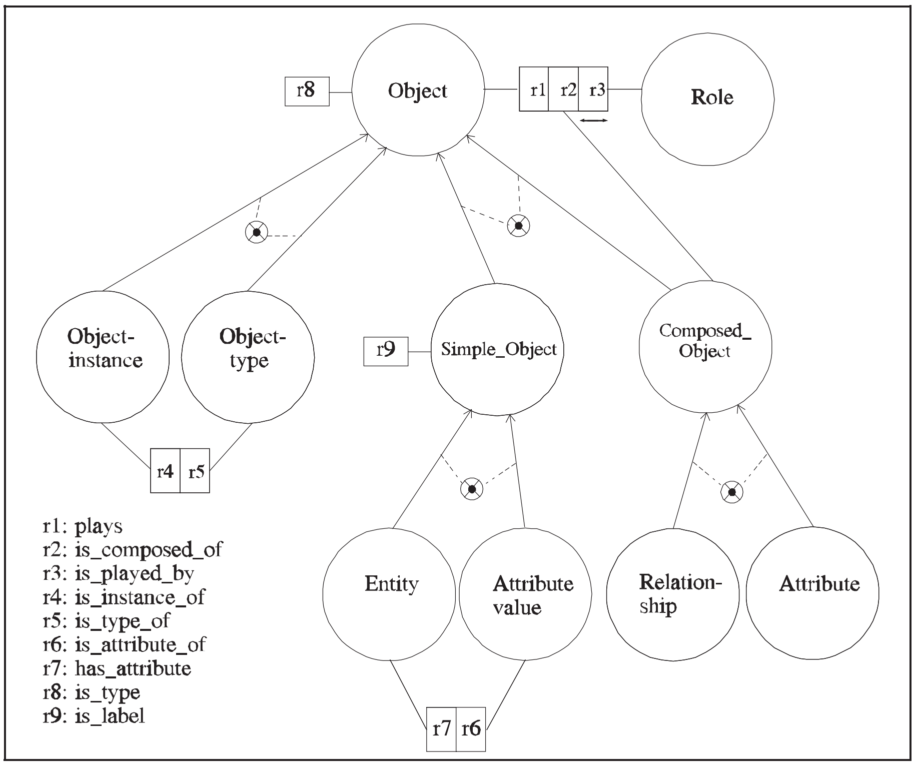


Figure 4.8 The meta model of ERM*

4.3.3 Incorporating Petri-Nets

In Petri-Nets transitions take place by changing the tokens in a place. The death and birth of tokens in a place, and the sequence of transitions, suggest the, at least implicit, existence of the notion of time. Therefore a partitioning of the meta concept Object in Time_Object and Non_time_Object is performed resulting in the meta model TPM* (figure 4.9). Two relationships between these new meta concepts are derived (from partitionings of the meta concepts Role and Composed_Object), representing the birth and death of a Non_time_Object.

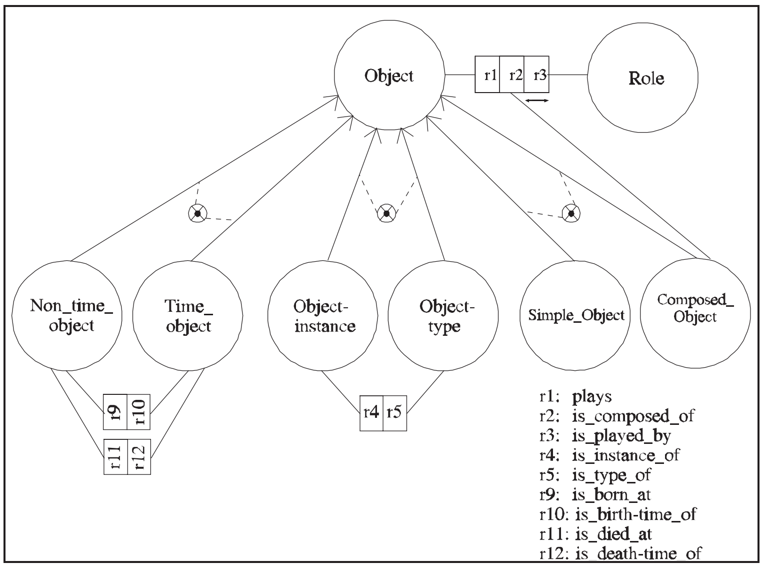


Figure 4.9 The intermediate meta model TPM*

From these birth and death relationships, transitions of Non_time_Composed_Object_Instances can be derived. By a thorough analysis of Petri-nets ([Prabhakaran88]) it can be shown that a token in a Petri-net is a set of these Non_time_Composed_Object_Instances. A token is also an instance of a place, which is a set of corresponding Non_time_Object_Types. A transition in a Petri-Net is now a set of elementary transitions of the Non_time_Composed_Object_Instances in a token. The resulting meta model of all these derivations is given in figure 4.10.

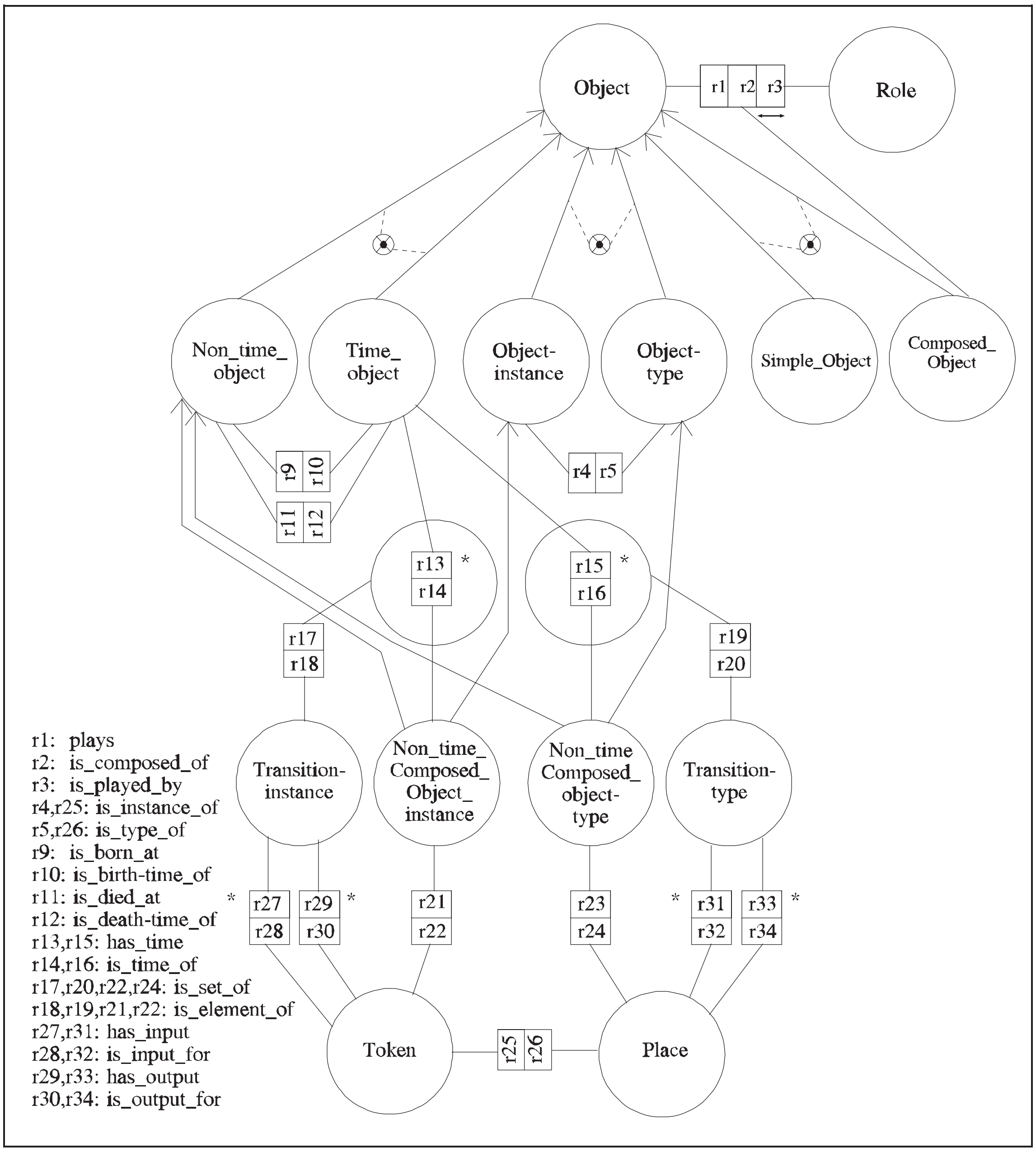


Figure 4.10 The meta model of Petri-nets*

In a Petri-net, however, the concept Time is not explicitly present. Therefore, the concept Time_Object is strongly degenerated. Finally, after a set of weak degenerations the meta model of Petri-Nets of figure 4.4 can be derived.

This concludes the illustrative example of the use of the Meta Model Hierarchy. The part of the Meta Model Hierarchy which has been constructed in this section was illustrated in figure 4.1. Note that a sample population of all meta models in the Meta Model Hierarchy can be taken as a checking mechanism for the constructed Meta Model Hierarchy. Information which is already present in the meta model of a technique down in the Meta Model Hierarchy, must be modelled explicitly in the application model up in the hierarchy. This is a result of the Principle of Conservation of Information as described in section 2.

5 Concluding remarks and further research

In this paper a framework for information systems concepts and modelling techniques has been presented. This framework, called the Meta Model Hierarchy, provides an ordering of meta models of techniques. Relations (viz. partitioning, restriction, and degeneration) on meta models have been defined, formulated in terms of constructive and destructive operations on meta models.

To perform constructive or destructive operations on a meta model, means essentially to shift the borderline between the meta model and the possible set of application models. This Principle of Conservation of Information means that facts, laws and rules of "reality" which are part of an application model before, become part of the meta model. The expressive power of the technique based on that meta model may or may not be affected. If one goes from the root of the Meta Model Hierarchy "downwards", expressive power stays the same or may diminish (due to degeneration), but may not increase. Also, by performing these operations on a meta model in a pragmatically adequate way, the convenience of the resulting meta model and a corresponding specification technique may increase significantly for a specific scope of applications. On the other hand, the breadth of convenience may diminish. For a specific sort of application, there may be one meta model or a few whose convenience can be regarded as optimal. Less partitioning, restriction or degeneration may diminish convenience due to too large generality, and more of that may diminish convenience due to overkill effects.

Probably the most important benefit of a Meta Model Hierarchy of meta models and corresponding specification techniques is the feature of upward compatibility. Due to that, application models, formulated on the basis of some meta model in the Meta Model Hierarchy, are automatically translatable into equivalent application models, formulated on the basis of any meta model closer to the root(s) of the Meta Model Hierarchy.

There is, however, no general downward compatibility which could lead to an automatic downward translatability of application models and operators. Downward translations can be done manually, and can be supported by a suitable modelling discipline.

Another important benefit of the Meta Model Hierarchy approach is the possibility to achieve evolvability of second order. Second-order evolution means that besides the application model, the meta model itself, with the corresponding specification language(s), can change too, without the need to install a new system and thereby interrupt the work in the organization. Second-order evolvability is particulary important for large organizations with various sorts of applications, and where also new sorts of applications become necessary from time to time.

Of course it is realized that the Meta Model Hierarchy at this stage has to be extended to support (automatical) transformations between meta models of different techniques. The ultimate framework should incorporate means to compare concrete syntax, abstract syntax and semantics of a technique. This requires a very rich meta language. The meta language which is used may not be that convenient for our purpose, and may therefore be revised. Possibly new operators on meta models have to be defined. Also to incorporate the operators of techniques in the comparison, the relationships between these operators have to be investigated.

There are more themes for future research in this field: Firstly, a reference root meta model of a Meta Model Hierarchy should be developed, with the aim to be able to reach all meta models suitable for all kinds of information systems, by means of meta model operations. Some important meta models should be specified and positioned in the Meta Model Hierarchy. Secondly, the question should be addressed as to which characteristics of any kind of information system require which meta model is optimally suited for this kind. The final option for future research, that we are currently addressing ourselves, is related to the manipulation of meta models for use in method engineering purposes (see 2.2). The Meta Model Hierarchy structures the method base. An open issue is the creation of links to standard method descriptions and to product standards, which have to be established on a certain level of abstraction. The incorporation of manipulation rules in a meta model for the generation of diagram editors by means of meta-case tools must still be investigated too.

Acknowledgement

We like to thank Terry Halpin, as well as the members of the PhD-consortium AXIS, for their constructive remarks on earlier versions of this paper.

References

[BHW91] Bommel, P. van, A.H.M. ter Hofstede and Th.P. van der Weide, Semantics and Verification of Object-Role Models, Information Systems, 16(5), 1991.

[BGKA89] Brinkkemper, S., M. Geurts, I. van de Kamp and J. Acohen, On a Formal approach to the Methodology of Information Planning. In: Proceedings of the First Dutch Conference on Information Systems, R. Maes (Ed.), November 1989, Amersfoort, the Netherlands.

[Bri90] Brinkkemper, S., Formalisation of Information Systems Modelling. Doctoral Dissertation, University of Nijmegen, June 1990, the Netherlands.

[BF91] Brinkkemper, S. and E.D. Falkenberg, Three Dichotomies in the Information System Methodology. In: P.W.G. Bots, H.G. Sol and I.G. sprinkhuizen-Kuyper (eds.), Informatiesystemen in beweging (Information Systems on the move), Kluwer Bedrijfswetenschappen, 1991.

[Che76] Chen P.P., The Entity-Relationship model - Towards a unified view of data. ACM Transactions on Database Systems, vol. 1, nr. 1, pp. 9-36, 1976.

[Ess86] Essink, L.J.B., A Modelling Approach to Information System Development. In: Information System Design Methodologies: Improving the Practice, T.W. Olle, H.G. Sol and A.A. Verrijn Stuart (Eds.), North-Holland, 1986.

[FGH91] Faerberboeck, H., Th. Gutzwiller and M. Heym, A Comparison of Requirements Engineering Methods using Metamodels. Technical report, Institute fuer Wirtschaftsinformatik, Hochschule St. Gallen, Switzerland, 1991.

[Fal88] Falkenberg, E.D., Deterministic Entity-Relationship Modelling, Technical Report no. 88-13, University of Nijmegen, The Netherlands, 1988.

[FL89] Falkenberg, E.D. and P. Lindgreen (Eds.), Information System Concepts: An In-depth Analysis. Proceedings of the IFIP TC8/WG 8.1 Working Conference on Information System Concepts: An In-depth Analysis, North-Holland, 1989.

[FOP91] Falkenberg, E.D., J.L.H. Oei and H.A. Proper, A Conceptual Framework for Evolving Information Systems. In Proceedings of the Second International Working Conference on Dynamic Modelling of Information Systems, Washington D.C., July 1991.

[FOP92] Falkenberg, E.D., J.L.H. Oei and H.A. Proper, Evolving Information Systems: beyond Temporal Information Systems, Technical Report 92-04, Department of Information Systems, University of Nijmegen, The Netherlands, 1992. (to appear in Proceedings DEXA’92, Valencia, Spain)

[FRE92] Falkenberg, E.D., C. Rolland and E.N. El-Sayed (Eds.), Information System Concepts: Improving the Understanding. Proceedings of the IFIP TC8/WG 8.1 Working Conference on Information System Concepts: Improving the Understanding, North-Holland, 1992.

[Flo86] Floyd, C., A Comparative Evaluation of System Development Methods. In: Information System Design Methodologies: Improving the Practice, T.W. Olle, H.G. Sol and A.A. Verrijn Stuart (Eds.), North-Holland, 1986.

[GS79] Gane, C. and T. Sarson, Structured Systems Analysis: Tools and Techniques. Prentice Hall, Englewood Cliffs, 1979.

[Hal89] Halpin, T.A., A Logical Analysis of Information Systems: static aspects of the data-oriented perspective, PhD thesis, University of Queensland, 1989.

[HO92] Halpin, T.A. and J.L.H. Oei, A Framework for Comparing Conceptual Modelling Languages, University of Nijmegen, The Netherlands, in preparation.

[HW91] ter Hofstede, A.H.M. and Th.P. van der Weide, Expressiveness in Data Modeling. Reportn91-07, SERC, Software Engineering Research Centrum, Utrecht, The Netherlands, July 1991. To be published in Data Knowledge Engineering.

[Lin90] Lindgreen, P. (editor), A Framework of Information System Concepts, Interim Report of the IFIP WG 8.1 Task Group FRISCO, 1990.

[LGN80] Lundeberg, M., G. Goldkuhl, and A.Nilsson, Information Systems Development - A Systematic Approach. Prentice Hall, Englewood Cliffs, 1980.

[Mar88] Martin, J., Information Engineering. Volume 1,2 and 3, Savant Research studies, Lancashire, 1988.

[NH89] Nijssen, G.M. and T.A. Halpin, Conceptual Schema and Relational Database Design: a Fact-Based Approach, Prentice Hall, 1989.

[OSV82] Olle T.W., H.G. Sol, and A.A. Verrijn-Stuart (eds.), Information Systems Design Methodologies: A Comparative Review. Proceedings of the CRIS 82 conference, North-Holland, Amsterdam, 1982.

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

[Pet81] Peterson, J.L., Petri-net theory and the modelling of systems, Prentice Hall, 1991.

[TL92] Tahvanainen, V.P. and K. Lyytinen, Lecture Notes of the First International Summerschool on Meta-Modelling and Method Engineering, University of Jyvaskyla, Finland, June 1992.

[VB82] Verheijen, G.M.A. and J. van Bekkum, NIAM: an Information Analysis Method. In: T.W. Olle, H.G. Sol and A.A. Verrijn Stuart (Eds.), Information Systems Design Methodologies: A Comparative Review. Proceedings of the CRIS 82 conference, North-Holland, Amsterdam, 1982.

[Was83] Wasserman, A.I., Characteristics of Software Development Methodologies. In: Information System Design Methodologies: A Feature Analysis, T.W. Olle, H.G. Sol and C. Tully (Eds.), North-Holland, 1983.

[Win90] Wintraecken, J.J.V.R., The NIAM Information Analysis Method: Theory and Practice. Kluwer Academic Publishers, 1990.

[YC79] Yourdon, E. and L. Constantine, Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design, Prentice Hall, Englewood Cliffs, 1979.