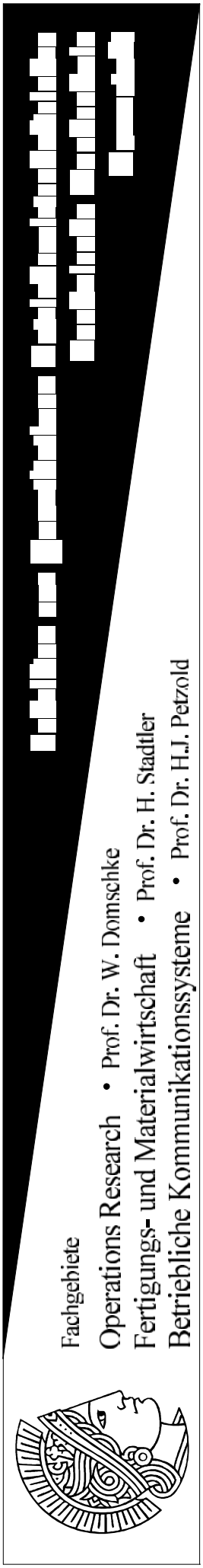



Susanne Strahringer

# Zum Begriff des Metamodells

Nummer 6/95
September 1995
Letzte Änderung: April 1996

Dipl.-Wirtsch.-Inf. Susanne Strahringer

Technische Hochschule Darmstadt
Institut für Betriebswirtschaftslehre
Wirtschaftsinformatik II
Betriebliche Kommunikationssysteme
Hochschulstr. 1, D-64289 Darmstadt
Tel. 06151/164416, Fax: 06151/165162
E-Mail: susanne@bwl.bwl.th-darmstadt.de

# Zum Begriff des Metamodells

Susanne Strahringer
Technische Hochschule Darmstadt
Fachgebiet Informationssysteme und Datenverarbeitung
Fachbereich Rechts- und Wirtschaftswissenschaften
D-64289 Darmstadt
susanne@bwl.bwl.th-darmstadt.de

**Abstract**

Die Informatik beschäftigt sich in vielen Bereichen mit Modellbildung. Werden Modelle und Modellbildung selbst zum Gegenstand der Modellierung, so spricht man von Metamodellen. Sie sind insbesondere in den methodologischen Bereichen der (Wirtschafts-)Informatik verbreitet. Der Begriff des Metamodells entbehrt jedoch einer präzisen Definition.
Im folgenden wird der Versuch unternommen, den Begriff aus der Sprachstufentheorie der Logik bzw. Sprachphilosophie herzuleiten. Der auf diese Weise gewonnene sprachbasierte Metamodellbegriff wird dann in einem zweiten Schritt über die Einführung eines sog. Metaisierungsprinzips verallgemeinert. Hierauf aufbauend läßt sich eine weitere im Bereich der Software-Prozeß-Modellierung anzutreffende Form der Metamodellierung identifizieren. Sie wird als prozeßbasierte Metamodellierung bezeichnet.
Wegen der größeren Relevanz und Verbreitung der sprachbasierten Form wird diese im weiteren Verlauf in den Vordergrund gestellt und anhand zweier Beispiele veranschaulicht. Die beiden Beispiele verdeutlichen, daß auf Metamodellierung basierende Modellhierarchien unterschiedlicher Natur sein können. Deshalb werden Merkmale solcher Modellhierarchien, die deren grundlegende Charakterisierung erlauben, identifiziert. Auf das Spektrum möglicher Einsatzgebiete wird ausblickend eingegangen.

### Inhaltsübersicht



## 1 Einleitung

Metamodellierung ist ein in der Wirtschaftsinformatik weit verbreitetes Instrument zur Beschreibung von Modellierungsmethoden. Insbesondere dann, wenn es um die Beschreibung ganzer Informationssystemarchitekturen geht, sind die Vorteile einer einheitlichen und formalisierten Beschreibung nicht von der Hand zu weisen. Die herausragende Bedeutung der Metamodellierung ist bei diesen phasenübergreifenden Ansätzen darin zu sehen, daß die Vielzahl zu dokumentierender Modellierungsformalismen (Modellierungssprachen) einheitlich und unter Berücksichtigung gegenseitiger Abhängigkeiten beschrieben werden kann. Daß verschiedene Modellierungssprachen existieren, ist als Mittel der Komplexitätsreduktion zu sehen. Zum einen werden über die Phasen hinweg verschiedene Sprachen benutzt und orthogonal dazu auch über verschiedene Sichten bzw. Perspektiven. Die sich dadurch ergebende Komplexitätsreduktion führt zu einem erhöhten Transformationsaufwand über die Phasen hinweg und zu einem erhöhten Integrationsaufwand zwischen den Sichten. Beides kann durch Metamodellierung sowohl in methodologischer Hinsicht wie auch bei der konkreten Methodenanwendung positiv beeinflußt werden, nämlich dadurch daß die entsprechenden Abhängigkeiten auf Metamodellebene explizit formuliert werden. Als Beispiele für Arbeiten, die Metamodellierung in dieser Form einsetzen, lassen sich z.B. Albers (1995), Gutzwiller (1994), Olle et al. (1991), Scheer (1992) anführen. Abb. 1 zeigt exemplarisch ein Metamodell aus Albers (1995).

Trotz der Bedeutung der Metamodellierung in diesem zentralen Bereich der Wirtschaftsinformatik ist der Begriff des Metamodells selbst selten Gegenstand einer ausführlichen Betrachtung. Dies ist vornehmlich darauf zurückzuführen, daß in einem konkreten Anwendungskontext die Idee und Ausgestaltung der Metamodellierung einfach und eingängig vermittelt werden kann. Auf Grundlage des heute zur Formulierung von Metamodellen vornehmlich verwendeten Ansatzes, nämlich dem der Entity-Relationship-Modellierung, lassen sich über konkrete Metamodell-Beispiele grundlegende Ideen der Metamodellierung einfach veranschaulichen. Daß Metamodellierung nicht auf ER-Modellierung beschränkt ist und daß der Begriff definitorisch per se nicht auf diese Grundlage gestellt werden muß, sondern wesentlich allgemeiner definiert werden kann, ist Gegenstand der vorliegenden Arbeit. Zur Veranschaulichung dieser Problematik sollen zunächst einige Metamodellbeispiele vorgestellt und ein Überblick über gängige Metamodelldefinitionen gegeben werden.

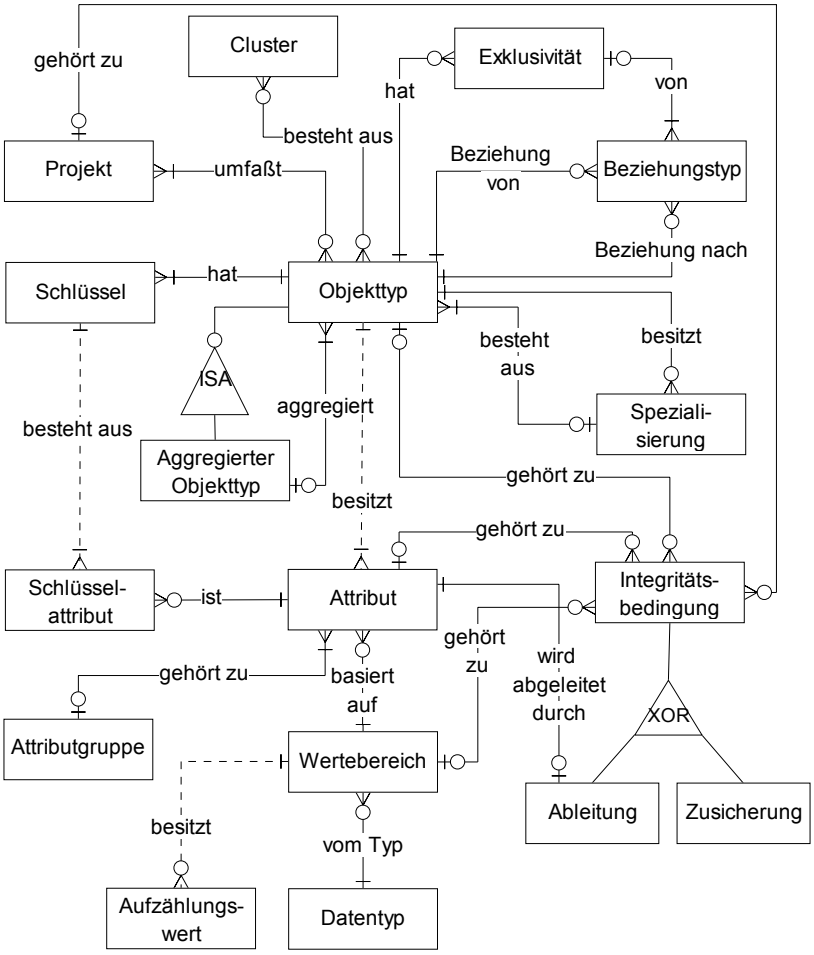



**Abb. 1: Metamodell eines ER-basierten Modellierungsformalismus**
Quelle: Albers (1995, S. 76)

Das in Abb. 1 dargestellte Metamodell stellt ein typisches Beispiel für die Verwendung von Metamodellen in der Wirtschaftsinformatik dar. Es handelt sich um ein Datenmodell eines Datenmodells. Daß auch Datenmodelle von Funktionsmodellen erstellt werden, wie in Abb. 2 dargestellt, ist inzwischen gängige Praxis. Es stellt sich die Frage, ob es Funktionsmodelle von Datenmodellen oder Funktionsmodelle von Funktionsmodellen gibt und ob auch diese als Metamodelle zu bezeichnen sind. In der Tat sind auch solche Fälle anzutreffen. Naheliegend ist es deshalb, Metamodelle sehr allgemein als Modelle anderer Modelle zu definieren. Berücksichtigt man allerdings die Breite des Modellbegriffs an sich, so wird deutlich, daß eine solche Definition zu weit gefaßt ist. Völlig offen ist die Frage nach der *Art* der Modelle, und zwar in doppelter Hinsicht: in bezug auf das Modell, das Abbild im betrachteten Modellierungsprozeß ist, und in bezug auf dasjenige, das den Gegenstand der Modellierung bildet. Zielsetzung ist es daher, bzgl. beider Aspekte eine Präzisierung vorzunehmen, ohne den Metamodellbegriff zu eng als ER-Modell einer Methode zu definieren, auch wenn es sich hierbei um die am häufigsten vorkommende Form von Metamodellierung in der Wirtschaftsinformatik handelt.

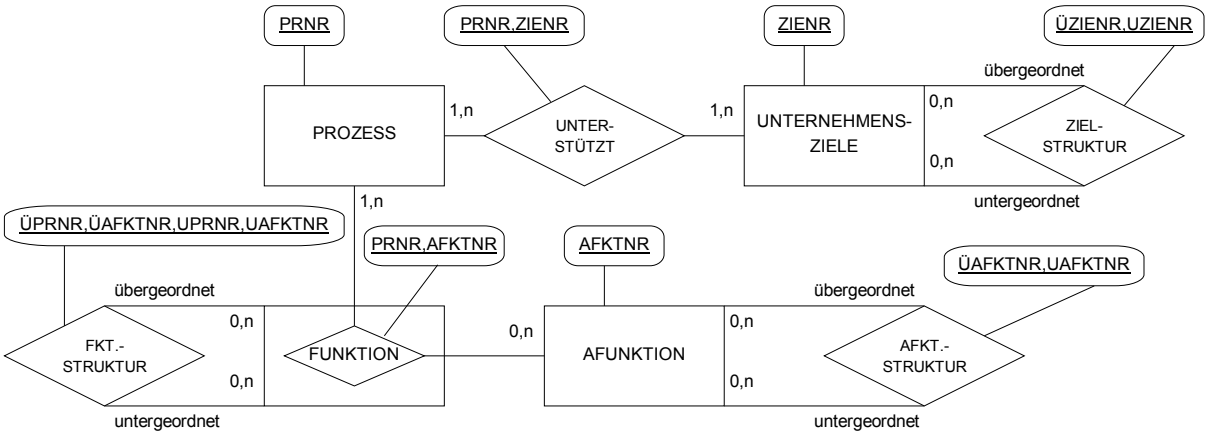
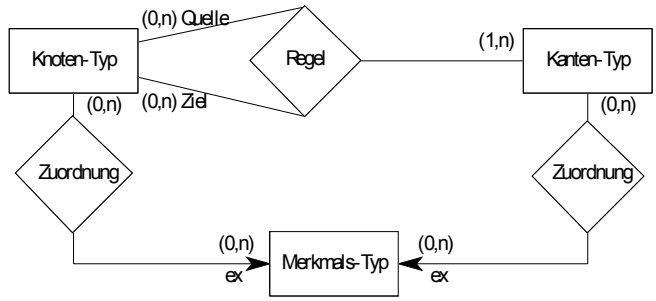


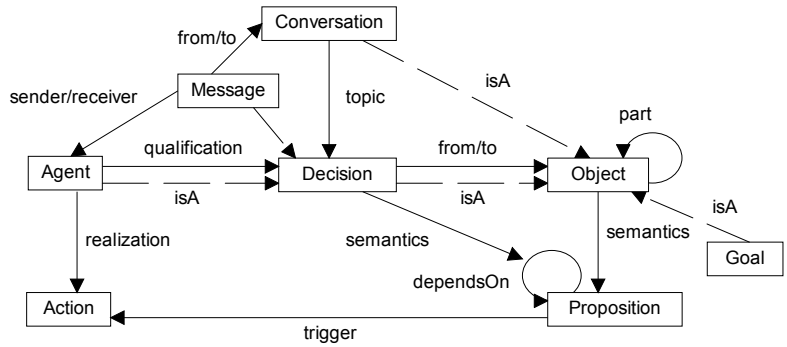

**Abb. 2: Metamodell zur Funktionsmodellierung**
Quelle: Ausschnitt aus Scheer (1992, S. 79)

Daß in der Literatur unter Metamodellierung ein breites und heterogenes Spektrum von Modellen verstanden wird, veranschaulicht die exemplarische Auswahl an Beispielmodellen in Abb. 3. Jedes der dargestellten Modelle wird von den jeweiligen Autoren selbst oder von anderen als Metamodell verstanden. Tab. 1 gibt einen Überblick über in der Literatur anzutreffende Definitionen bzw. Erklärungen des Metamodellbegriffs.

Beispiel 1: Hars (1994, S. 50)




Beispiel 2: Jarke (1992, S. 165)




Beispiel 3:
Oquendo, Zucker, Griffiths
(1992, S. 577)

```
MASP c-prog-pm HAS TYPE
            c_prog_develop:
  (IN _program: c_program;
   OUT _exec: exec_module);

OBJECT MODEL IS c_prog;
END OBJECT MODEL;

OPERATOR MODEL IS
 edit: (INOUT _m:c_module);
 compile: ....
 link: ....
END OPERATOR MODEL;

RULE MODEL IS
 module_linked:
 IF CREATE LINK exec (_p,_e)
 THEN
 set (linked, _p, TRUE);
END RULE MODEL;

   .
   .
   .

END MASP;
```

**Abb. 3: Metamodellbeispiele - Teil 1**

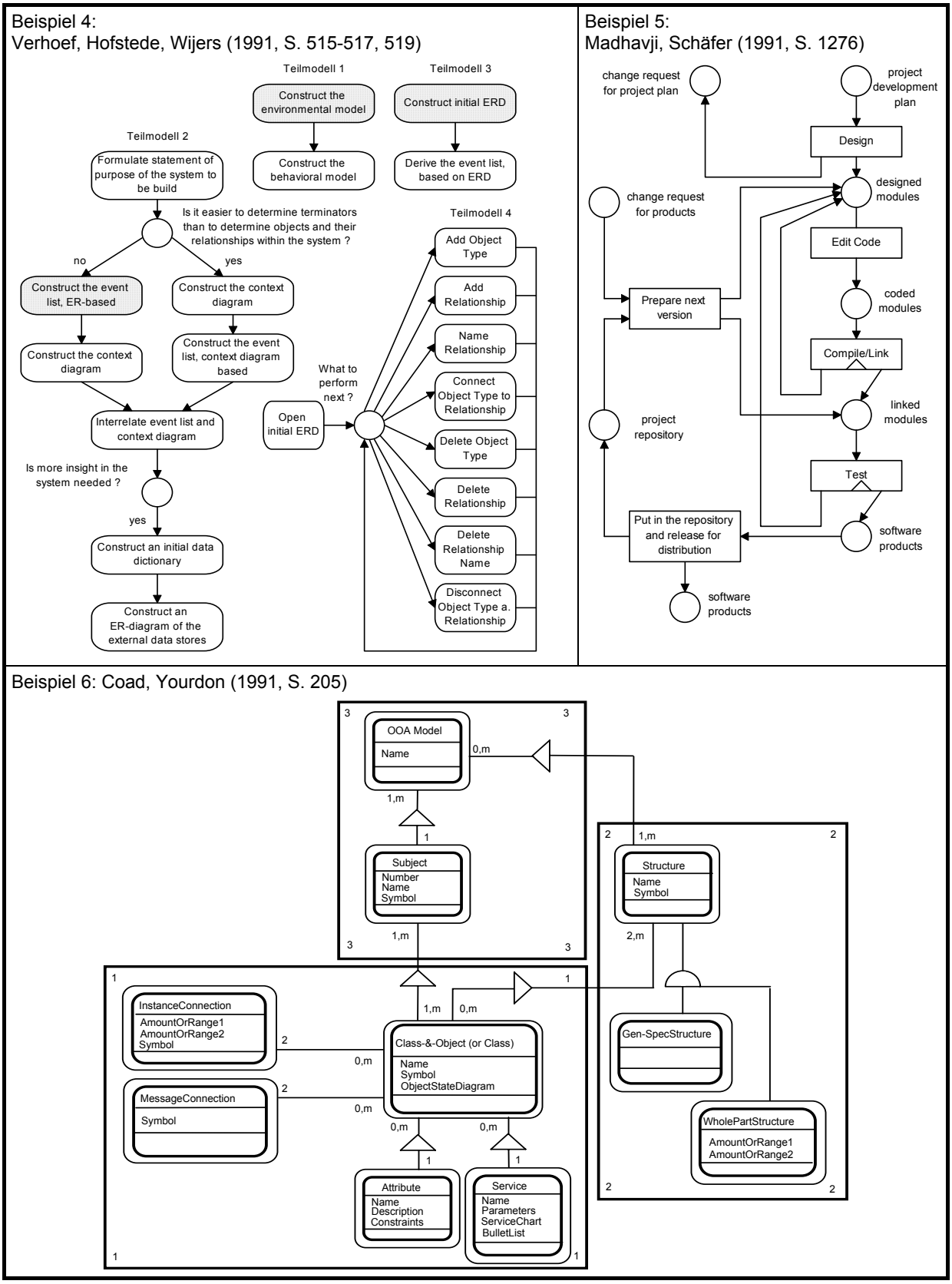



**Abb. 3: Metamodellbeispiele - Teil 2**

| | Autor | Verwend. Begriff | Definition, Erklärung |
|---|---|---|---|
| 1 | Amberg, Raue (1995, S. 58) | Metamodell | definiert (semi-)formales Beschreibungsmittel zur Erstellung eines Modells |
| 2 | Atzeni, Torlone (1993, S. 351) | metamodel | a formalism for the definition of models, model: a formalism for the description of schemes |
| 3 | Blaha (1992, S. 13) | metamodel | model that describes other models |
| 4 | Callender (1986, S. 49) | meta-model | provides a framework for the statement of most of the existing software process models and guidelines for judging and comparing different software process models |
| 5 | Dowson (1987, S. 36) | process metamodel | primitives of a language for expressing software process models |
| 6 | Ferstl, Sinz (1993, S. 88, 89) | Meta-Modell | ein Gestaltungsrahmen, der die verfügbaren Arten von Modellbausteinen (Objekttypen) und Beziehungen zwischen Modellbausteinen zusammen mit ihrer Semantik festlegt sowie Regeln für die Verwendung und Verfeinerung von Modellbausteinen und Beziehungen definiert. . . . Ein Datenmodell ist . . ein Meta-Modell im . . beschriebenen Sinne. |
| 7 | Frank (1994, S. 172) | Metamodell | Modell der zur Modellierung verwendeten Konzepte |
| 8 | Gutzwiller (1994, S. 8, 14, 24) | Metamodell | konzeptionelles Datenmodell einer Methode bzw. der Entwurfsergebnisse oder: im wesentlichen ein Entity-Relationship-Modell, das die Struktur der Entwurfsdaten, über alle Entwurfsergebnisse gesehen, als Datenmodell darstellt |
| 9 | Hesse (1991, S. 3) | metamodel | describes the development process, determines the structure for administering its results, defines the basic categories, relationships, structuring principles and the terminology used for the development of application systems |
| 10 | Hesse et al. (1994, S. 45) | Metamodell | Modell höherer Abstraktionsstufe, Abstraktionsstufen sind: Exemplar-, Typ- und Meta-Ebene |
| 11 | Heym (1995, S. 58) | Meta-Model-lierung | Modellierung und Formalisierung von Entwicklungstechniken für Informations-systeme, Arten: Meta-Daten-Modelle: statische Aspekte von Entwicklungs-techniken, Meta-Aktivitäten-Modelle: dynamische Aspekte von Entwicklungs-techniken |
| 12 | Iscoe, Williams, Arango (1991, S. 341) | meta-model | generalized form of the representation of a set of domain models |
| 13 | Kokol (1993, S. 25) | meta-modeling | software process modeling |
| 14 | Lehner (1995, S. 39) | Meta-Modell | Modell des Modells, das z.B. Teile des Systemplanungs- und -entwicklungs-prozesses und Beziehungen zwischen diesen Teilen zeigt |
| 15 | Moser (1995, S. 63) | metamodel | formally shows the structure of models |
| 16 | Odell (1992, S. 45) | metamodel | a model that defines other models |
| 17 | Rossi et al. (1992, S. 548) | metamodel | representation of a method |
| 18 | Rumbaugh (1995, S. 12) | metamodel | a model that describes other models, metamodel for a method: describes the concepts in the method and their relationship to each other |
| 19 | Scheer (1990, S. 1018f) | Metamodell (Meta-)Infor-mationsm. | Modell des Informationssystems, beschreibt dessen Darstellungselemente mit den zwischen ihnen bestehenden Beziehungen |
| 20 | Smolander et al. (1991, S. 172) | metamodel | representation of a methodology, methodology specification |
| 21 | Starke (1993, S. 13) | (process) meta-model | defines process of process modelling |
| 22 | Steele, Zaslavsky (1994, S. 316) | meta model | abstractions or models of the semantics of a modelling technique |
| 23 | Verhoef, Hofstede, Wijers (1991, S. 504) | meta-model | prescribes the type of modelling process or type of models used at the application level |
| 24 | Wileden (1986, S. 9) | metamodel | provides a vocabulary in which models, approaches, and methods can be described, compared, and contrasted |

**Tab. 1: Exemplarische Auswahl möglicher Metamodelldefinitionen**

Im ersten Beispiel aus Abb. 3 ist ein Metamodell dargestellt, das nicht die konzeptionellen Aspekte eines Modellierungsansatzes darstellt, sondern auf die verwendeten Symbole abzielt. Das zweite Beispiel zeigt in Form eines semantischen Netzes einen Ausschnitt aus einem Softwareentwicklungsprozeß, in dem es um die Konversation zwischen Agenten bzgl. der Objekte eines Repositories geht. Das erste nicht grafische Metamodell zeigt Beispiel 3. Es handelt sich um ein Prozeßmodell zum Editieren, Kompilieren und Linken von C-Programmen. Ein ähnliches etwas umfassenderes und von der Programmiersprache unabhängiges Modell ist im 5. Beispiel dargestellt. Das 4. Beispiel beschreibt die Vorgehensweise der Modernen Strukturierten Analyse. Dargestellt sind vier Verfeinerungen. Die jeweils grau unterlegten Teilschritte sind im darauffolgenden Modell

detaillierter abgebildet. Im 6. Beispiel ist ein Modell eines objektorientierten Modellierungsansatzes zu sehen, das unter Verwendung des Ansatzes selbst modelliert wurde.

Die ausgewählten Beispiele und Definitionen werfen folgende Fragen auf:[1]

- Handelt es sich bei Metamodellen immer um Datenmodelle ?
  Dies wird nur in Definition 8 nahegelegt, die anderen Erklärungsversuche weichen von dieser Einschränkung ab. Auch die Beispiele machen dies deutlich.
- Sind Metamodelle in sich selbst formulierbar ?
  Offenbar wird dies wie in Beispiel 6 dargestellt nicht abgelehnt.[2]
- Sind Metamodelle immer graphische Modelle ?
  Beispiel 3 zeigt, daß dies nicht der Fall sein muß.
- Kann ein Metamodell die graphischen Symbole des abgebildeten Modells definieren ?
  Dies gilt bedingt für Beispiel 1.
- Sind Vorgehensmodelle bzw. Softwareprozeßmodelle Metamodelle ?
  In den Beispielen 3, 4 und 5 wird offenbar davon ausgegangen wie auch in den Definitionen 9, 11, 13, 14, 23.
- Sind Vorgehensmodelle zur Erstellung von Vorgehensmodellen Metamodelle ?
  Der Definition 21 zufolge ist das Modell des Prozesses zur Prozeßmodellierung ein Metamodell und nicht bereits das Prozeßmodell selbst wie in den zuvor genannten Definitionen (9, 11, 13, 14, 23).
- Sind Modelle von Beschreibungsformalismen zur Formulierung von Vorgehensmodellen Metamodelle ?
  Gemäß Definition 5 ist nicht das Modell des Vorgehens bei der Prozeßmodellierung, sondern das Modell der zur Prozeßmodellierung verwendeten Sprachmittel ein Metamodell.
- Sind Beschreibungsformalismen Metamodelle oder erst deren Modelle ?
  Definition 6 bezeichnet bereits die zur Modellierung verwendete Sprache als Metamodell und nicht erst deren Modell. Ähnliches gilt für Definition 5.
- Sind Metamodelle auf Technik-, Methoden- oder Prozeßebene angesiedelt ?
  Die Bezugseinheit eines Metamodells ist unklar. Verschiedene Aussagen machen die Definitionen 11, 22 und 24.
- Was sind Metametamodelle ?
  Naheliegend ist die Definition als Metamodelle von Metamodellen. In Definition 2 wird allerdings eine andere Begriffshierarchie aufgebaut, nämlich eine Einteilung in Schema, Modell und Metamodell.[3] Auch die zuvor genannten Modelle von Prozessen oder Beschreibungsformalismen der Prozeßmodellierung müßten, falls man letztere als Metamodell versteht, als Metametamodell bezeichnet werden. Unter den dargestellten Beispiele wird Beispiel 1 vom entsprechenden Autor als Metametamodell verstanden.

Die vorangegangenen Fragestellungen veranschaulichen die Breite des Spektrums möglicher Interpretationsmöglichkeiten, die der Metamodellbegriff bietet, und die sich dabei abzeichnenden Probleme. Bei fast allen Erklärungsversuchen kann nachvollzogen werden, inwiefern die Autoren eine bestimmte Art von Modell als Metamodell einstufen. Dennoch fehlt ein Ansatz, der den Begriff nicht zu weit faßt, aber trotzdem so allgemein ist, daß verschiedene Fälle abgedeckt sind. Zielsetzung der folgenden Ausführungen ist es, einen solchen Erklärungsansatz zu entwickeln. Hierzu wird zunächst auf den Begriff der Sprache und den aus der Sprachstufentheorie stammenden Begriff der Metasprache eingegangen. Die auf Grundlage von Metasprachen bildbare Hierarchie von Sprachebenen wird nach einer Betrachtung des im vorliegenden Kontext verwendeten Modellbegriffs auf die Modellierung übertragen. Auf diese Weise läßt sich der Begriff des sprachbasierten Metamodells einführen. In einem weiteren Schritt wird dieser sprachbasierte Metamodellbegriff durch Einführung eines sog. Metaisierungsprinzips verallgemeinert. Hinzu kommt die sog. prozeßbasierte Metamodellierung. Da im Rahmen der Wirtschaftsinformatik die sprachbasierte Form vorherrschend ist, wird diese anhand zweier Beispiele veranschaulicht. Diese Beispiele verdeutlichen zudem, daß auf Metamodellierung basierende Modellhierarchien in verschiedener Hinsicht sehr unterschiedlich gestaltet sein können. Deshalb werden im vorletzten Kapitel Merkmale solcher Modellhierarchien, die deren grundlegende Charakterisierung erlauben, identifiziert. Bei konkreter Anwendung der Metamodellierung können diese Merkmale benutzt werden, um die Ausgestaltung der verwendeten Modellhierarchie zu beschreiben. Im letzten Kapitel wird ausblickend auf Einsatzgebiete der Metamodellierung eingegangen.

---

1 Die in der folgenden Diskussion erwähnten Definitionen und Beispiele beziehen sich auf Abb. 3 und Tab. 1.

2 Auf mögliche Probleme in Form eines Zirkelschlusses wird in Kap. 6.3 eingegangen.

3 Dieser Aspekt wird später unter dem Stichwort der Verankerung einer Modellhierarchie behandelt. Auf das spezielle Problem des Schema- vs. Modellbegriffs wird in Fußnote 23 eingegangen.

## 2 Sprache und Metasprache

Begriffsbildung und Sprache spielen eine wesentliche Rolle bei der Gewinnung wissenschaftlicher Aussagen. Nur auf ihrer Grundlage können diese formuliert und mitgeteilt werden. Begriffe werden benutzt, um Gegenstände mit ihren Eigenschaften und Relationen abzubilden, Sätze, um Sachverhalte auszudrücken. Um Begriffe und Sachverhalte mitteilen zu können, bedient man sich einer Sprache als einem "System von Zeichen und Regeln zur Verwendung dieser Zeichen".[4] Man spricht daher auch von Zeichensystemen.[5]

Es werden verschiedene *Arten von Sprachen* unterschieden. Eine sehr grundlegende Unterscheidung ist die in natürliche, also historisch entstandene, und künstliche Sprachen. Sprache kann in verschiedenen Situationen mit unterschiedlichen Zielsetzungen eingesetzt werden, d.h. mit Gesprochenem können verschiedene Wirkungen verbunden sein. Man nennt dies auch *Funktionen oder Leistungen der Sprache.* Nach Bühler kann zwischen den Funktionen Darstellung von Sachverhalten, Appell zur Verhaltenssteuerung und Ausdruck von Gefühlen unterschieden werden.[6] In konkreten, insbesondere natürlich-sprachlichen Sätzen treten mehrere dieser Funktionen gleichzeitig auf und sind oftmals nicht klar voneinander zu trennen. Für die folgenden Ausführungen ist dieses Problem von untergeordneter Bedeutung, da der Schwerpunkt der Betrachtungen nicht auf den natürlichen, sondern künstlichen Sprachen liegt, die lediglich die Darstellungsfunktion erfüllen können. In dieser Funktion ist auch die Leistung der Sprache für die Wissenschaft zu sehen.[7]

Auf der Darstellungsfunktion von Sprache aufbauend, werden Aussagen über einen zu untersuchenden Gegenstandsbereich in Sprache formuliert. Wird Sprache selbst zum Gegenstand der Untersuchung, so wird über Sprache in Sprache gesprochen. Bereits dieser Satz verdeutlicht die Verwirrung, die entstehen kann, wenn Aussagen nicht nur über außersprachliche Gebilde, sondern auch über sprachliche Gebilde selbst gemacht werden. Es ist daher zweckmäßig und in der Logik üblich, verschiedene Sprachebenen, auch semantische Stufen genannt, zu unterscheiden. Diejenige Sprache, die Gegenstand der Untersuchung ist, wird als Objektsprache bezeichnet, diejenige, in der die Untersuchung erfolgt, als Metasprache. Eine Sprache kann in diesem Sinne immer nur in bezug auf eine andere, nämlich die Objektsprache, Metasprache sein.[8]

Das Prinzip des Bildens einer Metasprache zu einer Objektsprache ist rekursiv anwendbar, denn auch die Metasprache einer Objektsprache kann wiederum zum Gegenstand der Untersuchung werden. Bei der für diese Untersuchung verwendeten Sprache handelt es sich dann um die Metasprache einer Metasprache oder um die Metametasprache bezüglich der ursprünglichen Objektsprache. Dieses Prinzip ist beliebig fortsetzbar. Es kann also eine ganze Hierarchie von Sprachebenen gebildet werden. Um dies terminologisch handhaben zu können, spricht man allgemein von Metasprachen n-ter Stufe in Kurzform auch mit "$M^nS$" bezeichnet. Dieser Konvention folgend steht "S" bzw. "$M^oS$" für die Objektsprache, "MS" bzw. "$M^1S$" für die Metasprache und entsprechend "MMS" bzw. "$M^2S$" für die Metametasprache.[9]

Das Differenzieren zwischen Meta- und Objektsprache dient der Bildung von Bezugsebenen und keinesfalls der Klassifikation von Sprachen. Die Eigenschaft einer Sprache, Meta- oder Objektsprache zu sein, ist in diesem Sinne keine absolute Eigenschaft dieser Sprache, sondern lediglich eine relative bzgl. einer anderen Sprache. Das Attribut "Meta-" bzw. "Objekt-" beschreibt die Rolle einer Sprache im Rahmen einer Untersuchung. Folglich kann ein- und dieselbe Sprache im Rahmen einer solchen Untersuchung sowohl die Rolle der Objekt- wie auch die Rolle der Metasprache einnehmen.[10]

Am Beispiel eines Dudens lassen sich diese Eigenschaften anschaulich verdeutlichen. Gegenstandsbereich des Dudens ist die deutsche Sprache, die hier also in der Rolle der Objektsprache auftritt. Der Duden selbst ist in deutscher Sprache geschrieben, also fungiert diese auch als Metasprache. Allerdings wird in einem Duden auch

---

4 Carnap (1960, S. 1).

5 Vgl. Bochenski (1965, S. 9, 11), Wild (1966, S. 45, 49-50), Oudenaarden (1953, S. 228).

6 Vgl. Bühler (1934, S. 26ff), zitiert nach Kroeber-Riel (1969, S. 12), Haberbeck (1986, S. 552), Mönke (1986, S. 551).

7 Vgl. Essler (1972, S. 306), Haberbeck (1986, S. 552), Kroeber-Riel (1969, S. 12).

8 Vgl. Kleinknecht, Wüst (1976, S. 42), Opp (1976, S. 313), Bochenski (1965, S. 59), Essler (1972, S. 302), Lorenz (1980a, S. 875), Lorenz (1980b, S. 1054).

9 Vgl. Kleinknecht, Wüst (1976, S. 43), Opp (1976, S. 313), Schreiber (1960, S. 15), Stachowiak (1973, S. 217), Lorenz (1980a, S. 875), Essler (1972, S. 304). Man beachte, daß manche Autoren, so z.B. Zemanek (1966, S. 140), die Stufe der Objektsprache als 1. und nicht als 0. Stufe bezeichnen.

10 Vgl. Kleinknecht, Wüst (1976, S. 42), Opp (1976, S. 313). Man beachte allerdings, daß der Begriff "Objektsprache" nicht immer in dieser Bedeutung, also als Sprache, die Gegenstand der Untersuchung ist, verwendet wird. Oftmals wird lediglich die unterste Ebene der Sprachstufenhierarchie als Objektsprache bezeichnet im Sinne einer Sprache, mit der ausschließlich über außersprachliche Gegenstände ((physikalische) Objekte) gesprochen wird. So z.B. bei Essler (1972, S. 304). Vgl. zu diesem Problem auch Popper (1993, S. 337) sowie Lorenz (1980b, S. 1054-1055).

diejenige Sprache, die verwendet wird, um Aussagen über die deutsche Sprache zu machen, genauer erklärt, nämlich im Rahmen eines einführenden Kapitels, das mit "Hinweise für die Wörterbuchbenutzung" bezeichnet wird. Es gibt folglich auch eine Metametaebene; die hier verwendete Sprache ist wiederum die deutsche.

Handelt es sich bei der Objekt- und Metasprache um dieselbe Sprache, so ist es notwendig, Konventionen zur Kennzeichnung der benutzten Sprachebene zu verwenden. Man unterscheidet daher zwischen *Gebrauch* und *Erwähnung* von Wörtern. In der geschriebenen Alltagssprache gibt es Konventionen, die eine Unterscheidung von Gebrauch und Erwähnung (Anführung) zulassen. Üblich ist die Verwendung von Anführungszeichen im Falle der Erwähnung. Die Unterscheidung "Gebrauch vs. Erwähnung" verwendend, läßt sich die Metasprache als diejenige Sprache bezeichnen, die man gebraucht, um Aussagen über eine Objektsprache zu machen.[11]

## 3 Modell und Metamodell

Auf der Darstellungsfunktion von Sprachen aufbauend, kann eine sprachliche Beschreibung eines Gegenstandsbereiches als eine Abbildung dieses Bereiches auf ein Zeichensystem verstanden werden. Sind Abbildung und Gegenstandsbereich strukturgleich oder strukturähnlich, so spricht man von einer isomorphen bzw. homomorphen Abbildung, die auch als Modell bezeichnet wird.[12] Sprache ist somit ein mögliches Instrument zur Darstellung von Modellen, wobei die Art der Sprache die Art des Modelles beeinflußt. Nicht jegliche sprachliche Darstellung eines Gegenstandsbereiches hat jedoch Modellcharakter. Die Literatur kennt eine Vielzahl von Modellbegriffen und Abgrenzungsmöglichkeiten.

In den Formalwissenschaften, insbesondere in der Mathematik und Logik, ist der Modellbegriff von zentraler Bedeutung und klar definiert.[13] Auch in den Naturwissenschaften basieren Modelle auf einem präzisen Modellbegriff. Vom mathematisch-logischen Sprachgebrauch stark abweichend und uneinheitlich ist er in den Geistes- und Sozialwissenschaften. Nicht einmal in den verschiedenen Einzelwissenschaften dieses Bereiches hat sich ein allgemein akzeptierter Modellbegriff durchgesetzt.[14] Im folgenden wird aufgrund der Einordnung der vorliegenden Arbeit in die Wirtschaftsinformatik vornehmlich auf betriebswirtschaftliche Literatur Bezug genommen.

Wie bereits dargestellt, ist der Vorgang der Abbildung wesentlich für die Modellbildung. Die Abbildungsfunktion einer Aussage allein kann jedoch nicht ausreichend sein, um von Modellbildung zu sprechen. Ansonsten würde bereits die Begriffsbildung eine einfache Form der Modellbildung darstellen.[15] Kosiol fordert daher, von Modellen erst dann zu sprechen, "wenn es sich um zusammengesetzte Gedankengebilde handelt, die aus der Totalinterdependenz der Wirklichkeit abgegrenzte und übersehbare Teilzusammenhänge ausgliedern".[16] Die Forderung nach zusammengesetzten Gedankengebilden schließt somit die Begriffsbildung als Form der Modellbildung aus. Wesentlich für die Modellbildung ist zudem der Begriff der Abstraktion, denn ein Modell stellt der Kosiol´schen Definition zufolge nur *Teil*zusammenhänge dar im Sinne einer Vereinfachung der darzustellenden Realität. Welche ihrer Eigenschaften bei der Modellierung weggelassen und welche besonders betont werden, also der Umfang der Abstraktion, wird bestimmt vom Zweck der Modellbildung.[17]

Eine bekannte Charakterisierung des Modellbegriffs, die die Zweckorientierung betont, geht auf Stachowiak zurück. Er unterscheidet drei Hauptmerkmale eines allgemeinen Modellbegriffs. Dies sind Abbildungs-, Verkürzungs- und Pragmatisches Merkmal. Das *Abbildungsmerkmal* bezeichnet das Modelloriginal. Der Aspekt des *Verkürzungsmerkmals* betont, daß Modelle nicht alle Eigenschaften des Originals abbilden, sondern nur eine Auswahl, die vom Modellierer und/oder vom Modellbenutzer (Modellsubjekt) abhängt. Erfolgt die Selektion

---

11 Vgl. Runggaldier (1990, S. 63), Stegmüller (1974, S. 30-32), Kleinknecht, Wüst (1976, S. 40-41), Popper (1993, S. 327), Bochenski (1965, S. 60), Essler (1972, S. 303-304).

12 Vgl. Wild (1966, S. 51).

13 Es handelt sich hier um einen Modellbegriff, der nahezu als Umkehrung eines in den Geisteswissenschaften verwendeten Modellbegriffs verstanden werden könnte. In den Formalwissenschaften versteht man unter einem Modell die Realisierung eines Kalküls als ein den Kalkül erfüllendes System von Objekten. Das Modell ist somit "das Besondere im Verhältnis zu einem Allgemeinen, von dem es Modell ist" (Klaus (1968, S. 412)). Vgl. hierzu Wolters (1980, S. 911), Spinner (1969, Sp. 1001-1002), Berthel (1970, Sp. 1122-1123). Repräsentativ für diese Auffassung ist beispielsweise Suppes (1961, S. 165). Vgl. zudem auch Stachowiak (1973, S. 3-4).

14 Vgl. Spinner (1969, Sp. 1000-1002), Wolters (1980, S. 911-912), Stachowiak (1973, S. 3), Ruffner (1972, S. 191), Lehner (1995, S. 27, 32).

15 Vgl. Kosiol (1961, S. 319), Ruffner (1972, S. 191).

16 Kosiol (1961, S. 319).

17 Vgl. Kosiol (1961, S. 319), Berthel (1970, Sp. 1122). Man beachte, daß die Modellierung einer per se existierenden Realität der Sichtweise des Konstruktivismus widerspricht, die Zweckgebundenheit der Modellbildung allerdings nicht. Siehe z.B. Lehner (1995, S. 31).

dieser Eigenschaften nach operationalen Zielsetzungen der Benutzer und werden diese und die Zeiten der Modellbenutzung miteinbezogen, so wird eine pragmatische Betrachtungsdimension erreicht. In diesem Sinne bezeichnet das *pragmatische Merkmal* des Modellbegriffs die Ersetzungsfunktion von Modellen bzgl. ihrer Originale, wobei die Ersetzung für bestimmte Subjekte (auch künstliche) innerhalb bestimmter Zeiträume und zu einem bestimmten Zweck erfolgt. Insgesamt sind Modelle folglich durch folgendes "Frage-Quadrupel" charakterisierbar: Wovon ? - Für wen ? - Wann ? - Wozu ?[18]

Eine sehr grundlegende Unterscheidung von Modellarten ist diejenige nach dem *Abbildungsmittel* (siehe Abb. 4), die sich auf die Art der benutzten Zeichen zurückführen läßt. Werden Zeichen verwendet, die ihrer äußeren Form nach eine anschaulich-bildliche Ähnlichkeit mit dem abgebildeten Objekt aufweisen (z.B. Bilder, Figuren, Gegenstände), so spricht man von (anschaulich-)*ikonischen* Modellen, die als Bildmodelle oder materiale Modelle (Gebildemodelle) vorkommen und zu den *nicht-linguistischen* Modellen gehören.[19] Handelt es sich bei dem verwendeten Abbildungsmittel um Sprache, so spricht man von *sprachlichen Modellen*[20] oder *linguistischen* Modellen. Das einzelne Zeichen hat in solchen Modellen keine anschauliche Bedeutung; die Beziehung zum bezeichneten Gegenstand wird nicht über Ähnlichkeit hergestellt, sondern allein durch Festsetzung definiert. Linguistische Modelle können nach Art der verwendeten Sprache weiter gegliedert werden in verbale, logistische und mathematische Modelle sowie die von Stachowiak eingeführte Modellart der graphischen Darstellung, auch Darstellungsmodell genannt.[21] Darstellungsmodelle werden in Diagrammsprachen formuliert. Für die vorliegende Arbeit sind ausschließlich linguistische Modelle von Bedeutung, insbesondere Darstellungsmodelle und verbale Modelle.

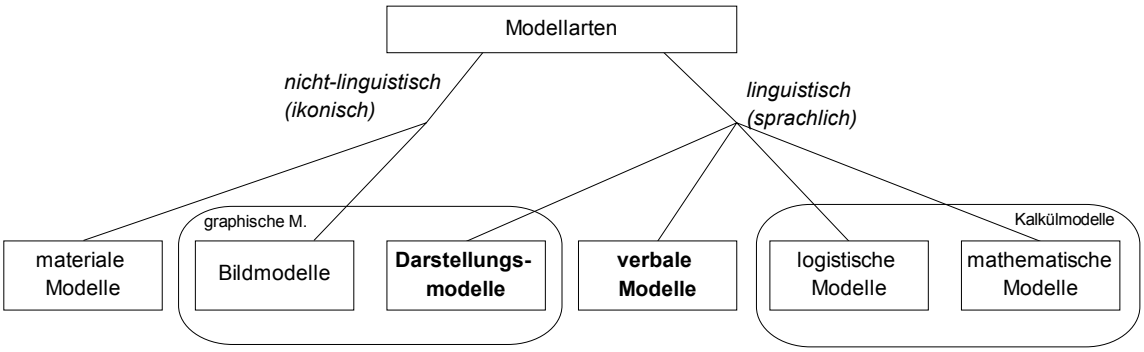



**Abb. 4: Klassifikation von Modellen nach dem Abbildungsmittel**

Unterscheidet man Modelle nach der *Funktion*, die sie erfüllen sollen, bzw. der Zielsetzung, die mit ihnen verfolgt wird, so ist in der Betriebswirtschaftslehre häufig eine Differenzierung in Beschreibungs-, Erklärungs- und Entscheidungsmodelle anzutreffen. Im Kontext der vorliegenden Arbeit sind ausschließlich Beschreibungsmodelle von Bedeutung, die der systematischen Beschreibung eines betrachteten Gegenstandsbereichs dienen und aus deskriptiven Satzsystemen bestehen.

Zusammenfassend läßt sich festhalten, daß im folgenden *linguistische Beschreibungsmodelle in graphischer oder verbaler Form* im Vordergrund stehen.

Versucht man, die Sprachstufentheorie auf die Modellbildung zu übertragen, so läßt sich in erster Näherung ein Metamodell beschreiben als ein Modell eines Modells. Diese Begriffsbildung, obwohl unmittelbar einleuchtend, ist aufgrund des wenig präzisen Modellbegriffs so weit gefaßt, daß eine operationale Handhabung schwierig ist. Ein zu breites und zu heterogenes Spektrum von Modellen wäre als Metamodell zu bezeichnen. Dies ist vornehmlich darauf zurückzuführen, daß in keiner Hinsicht festgelegt wird, welcher Aspekt des zugrundeliegenden Modells im Metamodell abgebildet wird. Deshalb soll im folgenden zunächst ein engerer Metamodellbegriff geprägt werden, welcher der Verwendung dieses Begriffes in den meisten Bereichen der Informatik entspricht. Dieser wird im nächsten Kapitel (Kapitel 4) verallgemeinert, ohne dabei die Eigenschaft der Operationalität zu verlieren.

Ziel der Modellbildung im Rahmen der Softwareentwicklung ist es, einen betrieblichen Gegenstandsbereich in einem Beschreibungsmodell abzubilden. Zur Erstellung dieses Modells wird eine Sprache verwendet, die der Sprachstufentheorie folgend als Objektsprache zu bezeichnen ist. Entsprechend ließe sich das Modell der

---

18 Vgl. Stachowiak (1973, S. 131-133).

19 Vgl. Spinner (1969, Sp. 1001), Kosiol (1964, S. 754), Berthel (1970, Sp. 1124), Stachowiak (1973, S. 163), Lehner (1995, S. 40).

20 Grochla (1969, S. 384) spricht hier von Sprachmodellen.

21 Vgl. Berthel (1970, Sp. 1124-1125), Kosiol (1964, S. 754), Stachowiak (1973, S. 163, 165).

untersten Ebene auch als Objektmodell bezeichnen. Im folgenden soll jedoch vereinfachend und wegen der zweideutigen Verwendung des Begriffs Objektmodell[22] von Modell gesprochen werden. Wird die Objektsprache, in der das Modell der untersten Stufe formuliert ist, abgebildet in einem Beschreibungsmodell, so handelt es sich um ein Metamodell. Dieses Metamodell ist insofern ein Modell des Modells der untersten Stufe, als daß es ein Modell der dort zur Modellierung eingesetzten Objektsprache ist. Entsprechend ist das Metamodell in einer Sprache formuliert. Da dies die Sprache ist, mittels derer die Objektsprache hier in Form eines Modells beschrieben wird, handelt es sich um eine Metasprache bzgl. der Objektsprache. Wird diese Metasprache wiederum in einem Modell abgebildet, so handelt es sich dabei um ein Metametamodell. Ein Metamodell der i-ten Stufe kann somit als ein sprachliches Beschreibungsmodell der Sprache des Modells der (i-1)-ten Stufe definiert werden.

In diesem Sinne ist ein Metamodell also ein Modell eines Modells, *wobei es sich beim übergeordneten Modell um ein sprachliches Beschreibungsmodell handelt, das die Sprache des untergeordneten Modells abbildet*. Abb. 5 veranschaulicht den dargestellten Zusammenhang der identifizierten Modellierungsebenen bei Ableitung des Metamodellbegriffs aus der Stufentheorie der Sprachen.

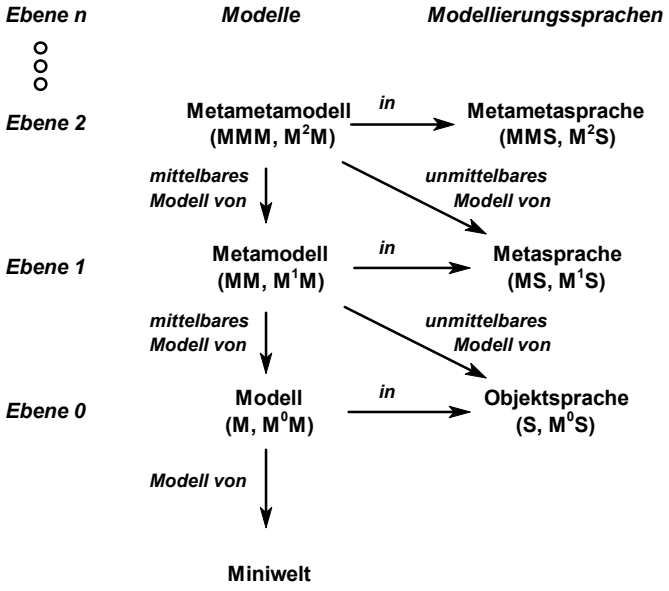



**Abb. 5: Sprachbasierter Metamodellbegriff**

Metamodelle auf Grundlage dieses sprachbasierten Metamodellbegriffs unterscheiden sich im wesentlichen dadurch, welche Eigenschaften der zu modellierenden Sprache sie abbilden und in welcher Sprache sie selbst formuliert sind. Die zur Modellformulierung verwendete Sprache wird als Modellierungssprache bezeichnet. Auch hier kann ein- und dieselbe Sprache unterschiedliche Rollen wahrnehmen, beispielsweise als Objektsprache, Metasprache oder Metametasprache fungieren, je nachdem auf welcher Ebene das zugehörige Modell angesiedelt ist.[23]

Sprachbasierte Metamodelle derselben Sprache können sich also, auch wenn sie in derselben Metasprache formuliert sind, unterscheiden, nämlich dann, wenn der Modellbildung verschiedene Verkürzungsmerkmale zugrundeliegen, so daß von unterschiedlichen Eigenschaften des Modelloriginals abstrahiert wird. Im Rahmen der sprachbasierten Metamodellierung kann in dieser Hinsicht folgende Unterscheidung getroffen werden. Handelt es sich bei der zu modellierenden Objektsprache um eine Diagrammsprache, so können auf der übergeordneten Modellebene entweder die Begriffe oder die Symbole der Diagrammsprache modelliert werden. Im

---

22 Dieser terminologische Konflikt tritt im Zusammenhang mit der objektorientierten Modellierung auf, die in der Sprechweise der Objektorientierung zu Objektmodellen führt unabhängig von der Modellierungsebene.

23 Man beachte zudem, daß es in der Informatik üblich ist, Modellierungssprachen selbst als Modell zu bezeichnen, z.B. das Relationen*modell*, das Entity-Relationship-*Modell*. Im englischsprachigen Raum führt dies insofern nicht zu wesentlichen terminologischen Schwierigkeiten, als daß die mittels solcher Modellierungssprachen erstellten Modelle nicht als "model", sondern oftmals als "schema" bezeichnet werden. Im Deutschen wie auch bei einigen wenigen englischsprachigen Autoren hat sich diese Begriffsbildung nicht durchgesetzt. Man bezeichnet als Modell sowohl das mittels einer Modellierungssprache erstellte Abbild eines Modelloriginals wie auch die Modellierungssprache selbst. Dieser Konflikt ist im Themenbereich der Datenmodellierung besonders evident. Zudem kommt erschwerend hinzu, daß einige Autoren jegliche Modellierungssprache als Metamodell bezeichnen und nicht etwa wie im Rahmen dieser Arbeit deren Beschreibung in Form eines Modells. So z.B. Ferstl, Sinz (1993, S. 89).

ersten Fall spricht man auch von einem *konzeptionellen* Metamodell. Der Aspekt konzeptionell betont dabei in Anlehnung an den englischen Begriff "concept" (engl. Wort für Begriff) den stark ausgeprägten Bezug zu Begriffen.[24] Ein konzeptionelles Modell einer Sprache abstrahiert somit von den grafischen Symbolen und legt statt dessen einen Schwerpunkt auf die mit diesen Symbolen bezeichneten Begriffe.[25] Im zweiten Fall wird in einem Metamodell, das hier im Gegensatz zu einem konzeptionellen als *repräsentationelles* bezeichnet wird, der Aspekt der grafischen Symbole (oder deren topologische Anordnung) betont. Nur sehr wenige Autoren verstehen ein Metamodell in diesem repräsentationellen Sinn.[26] Das erste Metamodell in Abb. 3 ist ein Beispiel für ein repräsentationelles Metamodell. In Abb. 6 ist ein weiteres stark vereinfachtes Beispiel dargestellt. Zu einem ER-Modell (Ebene 0) ist auf der linken Seite der Abbildung ein konzeptionelles und auf der rechten Seite ein repräsentationelles Metamodell abgebildet. Bei Objekt- und Metasprache handelt es sich um eine einfache Form der ER-Modellierung. Je nach Anwendungsgebiet der Metamodellierung kann es durchaus sinnvoll sein, ein konzeptionelles Metamodell um ein repräsentationelles zu ergänzen bzw. beide Aspekte in einem gemeinsamen Metamodell abzubilden.

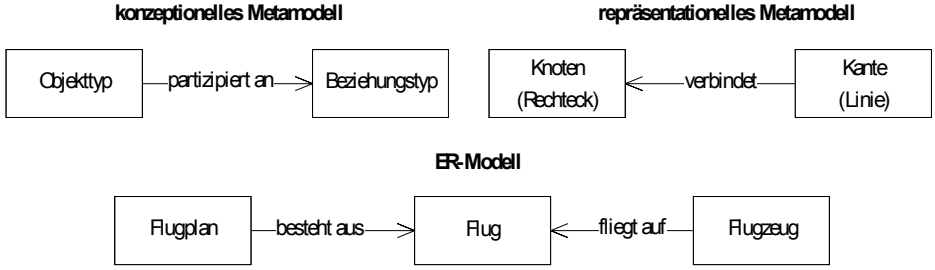



**Abb. 6: Beispiel eines konzeptionellen und eines repräsentationellen Metamodells**

Andere Formen von Metamodellen können identifiziert werden, wenn man von der Sprache als Modelloriginal absieht. In diesem Sinne läßt sich der hier vorgestellte sprachbasierte Metamodellbegriff durch die Einführung eines sog. Metaisierungsprinzips verallgemeinern.

## 4 Begriff des Metaisierungsprinzips

Im vorangegangenen Kapitel wurde ein Metamodellbegriff geprägt, auf den die meisten Verwendungen dieses Begriffs in der Informatik zurückgeführt werden können. Neben dieser Form ist jedoch noch mindestens eine andere Variante zu finden, die von praktischer Relevanz ist und dem Themengebiet der Softwareprozeßmodellierung[27] entstammt. Betrachtet man die Beispiele 4, 5 und 6 aus Abb. 3, wird deutlich, daß alle drei Modelle einen Ausschnitt aus dem Softwareentwicklungsprozeß beschreiben. Den Definitionen 9, 11, 13, 14, 23 (Tab. 1) zufolge handelt es sich bei dieser Form von Modellen um Metamodelle, wobei der Prozeß der Modellbildung im Vordergrund steht und nicht wie zuvor die Modellierungssprache.[28] Auf der Metamodellebene wird dann entsprechend der Prozeß der Modellbildung modellhaft abgebildet. Auf der darüberliegenden Ebene wird, das Prinzip der Metaisierung fortsetzend, der Prozeß der Prozeßbildung modelliert.[29] Abb. 7 zeigt eine Abb. 5 entsprechende Darstellung dieses Metamodellbegriffs.[30]

---

24 Dieser Bezug ist in der deutschen Sprache leider nicht evident, da das deutsche Wort Konzept nicht denselben Begriff bezeichnet wie das englische "concept" (engl. für Begriff).

25 Ausdrücklich betont wird dieser Aspekt beispielsweise bei Verhoef, Hofstede, Wijers (1991, S. 506), die sprachbasierte Metamodelle als "concept structures" bezeichnen. Ähnliches ist auch bei Steele, Zaslavsky (1993, S. 325) zu finden.

26 Die nachrangige Bedeutung des repräsentationellen Aspekts betonen beispielsweise Steele, Zaslavsky (1993, S. 317).

27 Auf eine Darstellung dieses Gebietes wird an dieser Stelle verzichtet. Eine grundlegende Einführung geben beispielsweise Curtis, Kellner, Over (1992).

28 Man beachte, daß es sich hier um verschiedene Arten von Prozessen handeln kann, z.B. solche, die den gesamten Softwareentwicklungsprozeß beschreiben, wie auch solche, die das Vorgehen im Rahmen der Anwendung einer Methode definieren. In diesem Sinn unterscheiden beispielsweise Pohl, Jarke (1992, S. 8) und Jarke (1992, S. 164) Prozeßdefinitionen nach der Granularität in "in the small (methods)", "in the large (versions & configurations)" und "in the many (teamwork protocols)".

29 In der Literatur lassen sich nur wenige Ansätze finden, die eine prozeßbasierte Metaisierung bis zur Ebene 2 vornehmen. Als Beispiele sind die Arbeiten von Madhavji, Schäfer (1991) und Verhoef, Hofstede, Wijers (1991) zu nennen. Ansätze, bei denen ein Vorgehensmodell als Metamodell bezeichnet wird, also eine prozeßbasierte Metaisierung bis zur Ebene 1 vorliegt, sind dagegen häufiger anzutreffen, z.B. bei Wileden (1986) und Pohl, Jarke (1992).

30 Nur in wenigen Arbeiten ist ein ausgewogenes Verhältnis zwischen diesen beiden Aspekten anzutreffen. Verhoef, Hofstede, Wijers (1991) beispielsweise betonen ausdrücklich die Sprach/Prozeß-Dualität bei Betrachtung einer

Der wesentliche Unterschied in den beiden Ansätzen ist in der Verwendung verschiedener Metaisierungsprinzipien zu sehen. Für beide Formen gilt jedoch, daß über die Modellhierarchie hinweg immer dasselbe Metaisierungsprinzip verwendet wird. Zudem stellt auch die Offenlegung des Metaisierungsprinzips einen wichtigen Aspekt dar. Denn letztlich kann nur in bezug auf ein gegebenes Metaisierungsprinzip entschieden werden, ob es sich um ein Modell der Ebene 0 oder höherer Ebenen, also um ein Metamodell, handelt. Dies soll an einem Beispiel einer gängigen, aber diesen Forderungen widersprechenden Verwendung des Metamodellbegriffs erläutert werden.

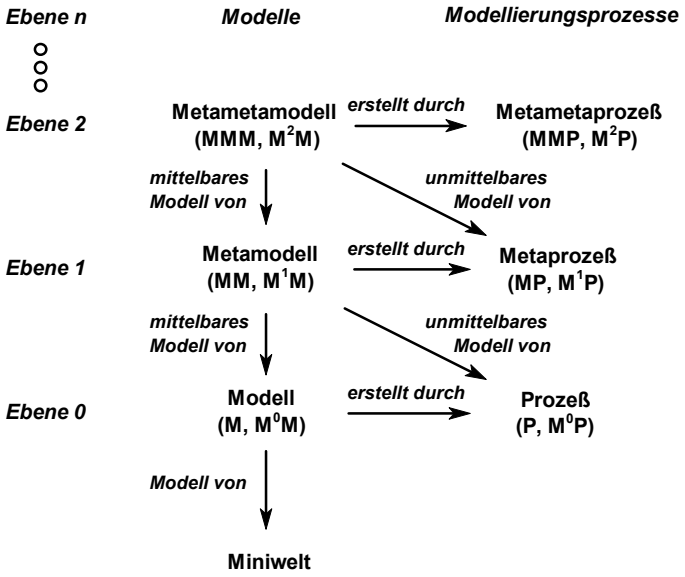



**Abb. 7: Prozeßbasierter Metamodellbegriff**

Gegenstand der Betrachtung sei eine Modellierungsmethode. Diese kann in zweierlei Hinsicht beschrieben werden, nämlich in bezug auf die verwendete Modellierungssprache und in bezug auf den Modellierungsprozeß, also sprach- und prozeßbasiert.

In der Tat stellen beide Aspekte wichtige Eigenschaften eines Modellierungsansatzes dar. Die Modellierung dieser beiden Aspekte wird als Metamodellierung bezeichnet, wobei jeweils verschiedene Sprachen als Modellierungssprachen verwendet werden, z.B. eine datenorientierte Sprache aus dem ERM-Bereich für den sprachlichen Aspekt und eine Petri-Netz-Variante für die Prozeßbeschreibung. Eine weitere Ebene darüber (Ebene 2) können die verwendeten Sprachen, also die ERM- und PN-Varianten, modellorientiert, z.B. unter Verwendung einer weiteren ERM-Variante, beschrieben werden. Abb. 8 veranschaulicht den dargestellten Zusammenhang.

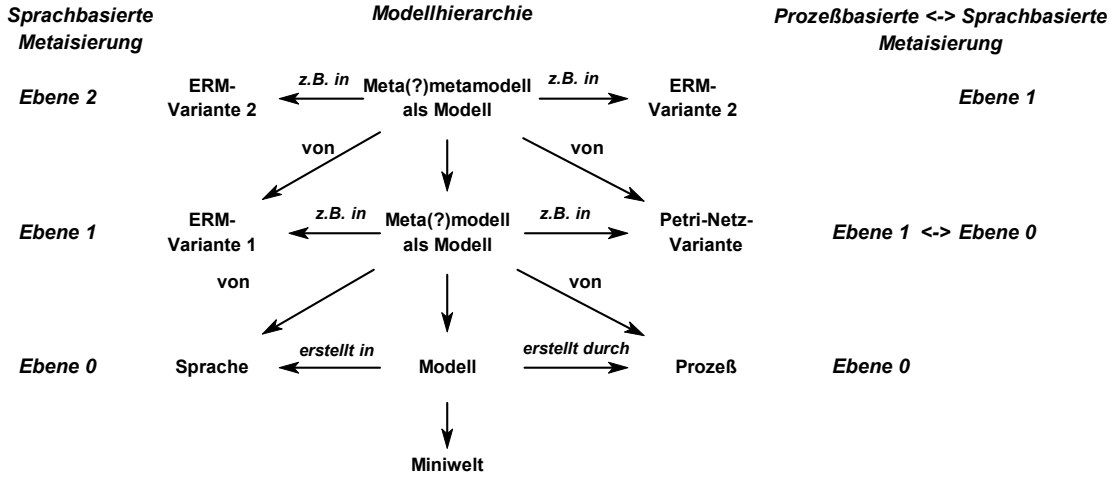



**Abb. 8: Widersprüchlicher Metamodellbegriff durch gleichzeitige Verwendung verschiedener Metaisierungsprinzipien**

---

Methode - in ihren Worten als "the way of modelling" vs. "the way of working" bezeichnet. Sie wenden Metamodellierung auch konsequenterweise auf beide Aspekte an.

In Abb. 9 ist eine stark vereinfachte Modellhierarchie für ein Datenflußdiagramm dargestellt. Es handelt sich um ein konkretes Beispiel der in Abb. 8 schematisch abgebildeten Struktur. Im Gegensatz zu Abb. 8 wurde im konkreten Beispiel für die Prozeßmodellierung eine einfachere Modellierungssprache als ein Petri-Netz-Ansatz gewählt, die sich allerdings nicht mit einem feststehenden Begriff bezeichnen läßt. Abgebildet ist auf Ebene 0 ein Datenflußmodell eines außersprachlichen Gegenstandsbereichs. Es handelt sich um ein stark vereinfachtes Modell des Flugkartenverkaufs einer Fluggesellschaft.[31] Auf der darüberliegenden Ebene ist links ein Modell der Sprachmittel der Datenflußmodellierung in Form eines ER-Modells abgebildet und rechts ein Modell, das den Prozeß der Erstellung eines Datenflußmodells zeigt. Auf der höchsten Ebene schließlich sind die für die Formulierung der Modelle der darunterliegenden Ebene verwendeten Sprachen als ERD dargestellt.

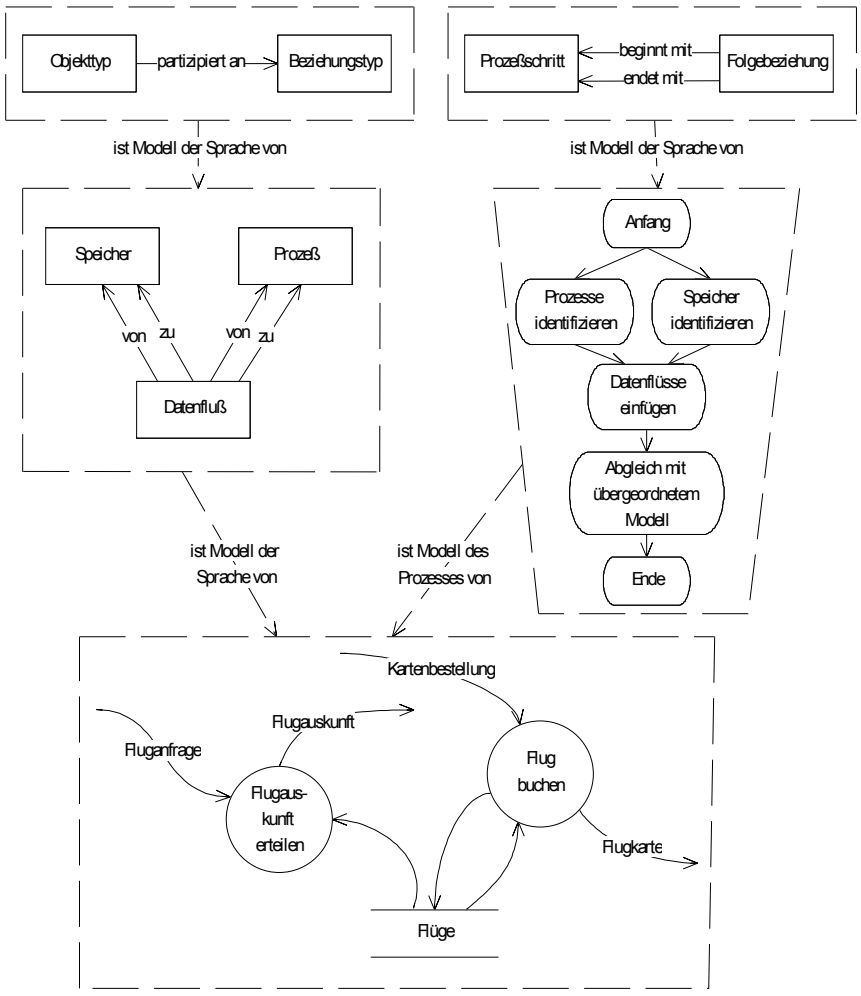



**Abb. 9: Modellhierarchie mit Datenflußdiagramm auf Ebene 0**

Das Bilden von Modellhierarchien in der Form, wie sie in den vorhergehenden Abbildungen aufgezeigt wurden, kann durchaus zweckmäßig sein. Bei der Identifizierung von Modellierungsebenen können jedoch Probleme auftreten. Es stellt sich die Frage, ob die beiden oberen Ebenen wirklich als Meta- und Metametaebene einzustufen sind. Bezüglich der linken Hälfte von Abb. 8 und 9 läßt sich diese Frage leicht beantworten, da es sich um eine herkömmliche sprachbasierte Metamodellierung handelt. Die rechte Seite bereitet jedoch in folgender Hinsicht Probleme. Dort findet ein Wechsel des Metaisierungsprinzips statt. Von Ebene 0 zu 1 erfolgt die Metamodellbildung prozeßbasiert und von Ebene 1 zu 2 sprachbasiert. Betrachtet man ein Prozeßmodell zu einer Modellierungsmethode, so ist dieses lediglich bei prozeßbasierter Metaisierung ein Metamodell. Legt man Sprachbasierung zugrunde, so handelt es sich bei diesem um ein Modell nicht um ein Metamodell. Wäre es ein Metamodell, müßte sich nach dem gewählten Metaisierungsprinzip eine zugehöriges Modell finden lassen. Dies ist aber unmöglich. Die Ebenennumerierung auf der rechten Seite von Abb. 8 verdeutlicht die korrekte Ebenenbildung. Das aus prozeßbasierter Sicht als Metamodell bezeichenbare Modell ist aus sprachbasierter Sicht ein Modell, folglich ist auch die darüberliegende Ebene keine $M^2$-Ebene, sondern lediglich die Metamodellebene der sprachbasierten Betrachtungsweise. Die Überlegung macht deutlich, daß die Entscheidung

31 Das Datenflußdiagramm zum Flugkartenverkauf ist an ein Beispiel aus Raasch (1993, S. 91) angelehnt.

darüber, ob es sich bei einem Modell um ein Metamodell handelt, davon abhängt, von welchem Metaisierungsprinzip man ausgeht.

Modellierungsebenen lassen sich unter Zugrundelegung eines Metaisierungsprinzips rekursiv definieren. Eine solche rekursive Definition könnte für eine sprachbasierte Hierarchie wie folgt lauten

$M^iM$ := Modell_der_Modellierungssprache ($M^{i-1}M$) für i = 1,..., n

und entsprechend im prozeßbasierten Fall

$M^iM$ := Modell_des_Modellierungsprozesses ($M^{i-1}M$) für i = 1,..., n.

Die Möglichkeit der rekursiven Definition der Modellierungsebenen ist nur bei Vorliegen eines durchgängigen Metaisierungsprinzips möglich. Wird innerhalb einer Modellhierarchie das Metaisierungsprinzip variiert, so ist die rekursive Ebenenbildung auf jede der durchgängig auf demselben Prinzip beruhenden Teilhierarchien separat anzuwenden. Entsprechend wurde in Abb. 8 (rechte Seite) verfahren.

Zusammenfassend läßt sich sagen, daß die Offenlegung eines Metaisierungsprinzips und eine diesem Prinzip folgende Ebenenbildung insofern wichtig sind, als daß nur auf diese Art und Weise das Verhältnis zwischen Modell und Metamodell innerhalb von Modellhierarchien zum Ausdruck gebracht werden kann.

# 5 Beispiele für sprachbasierte Modellhierarchien in der Informatik

Im folgenden werden zwei Beispiele für Modellhierarchien in der Informatik vorgestellt, denen ein sprachbasiertes Metaisierungsprinzip zugrunde liegt. Die Einschränkung auf sprachbasierte Modellhierarchien läßt sich mit deren stärkeren Verbreitung und ihrer größeren praktischen Relevanz begründen.

## 5.1 Am IRDS-Standard orientierte Repository-Architekturen

Unter einem Repository ist ein datenbankbasiertes Softwaresystem zu verstehen, das Informationen über Objekte, die in der Softwareentwicklung anfallen, verwaltet. Das Information Resource Dictionary System (IRDS) ist ein von ANSI und ISO geprägter Begriff, die für ein solches Repository eine standardisierte Architektur, die mehrere Modellierungsebenen umfaßt, vorgeschlagen haben.

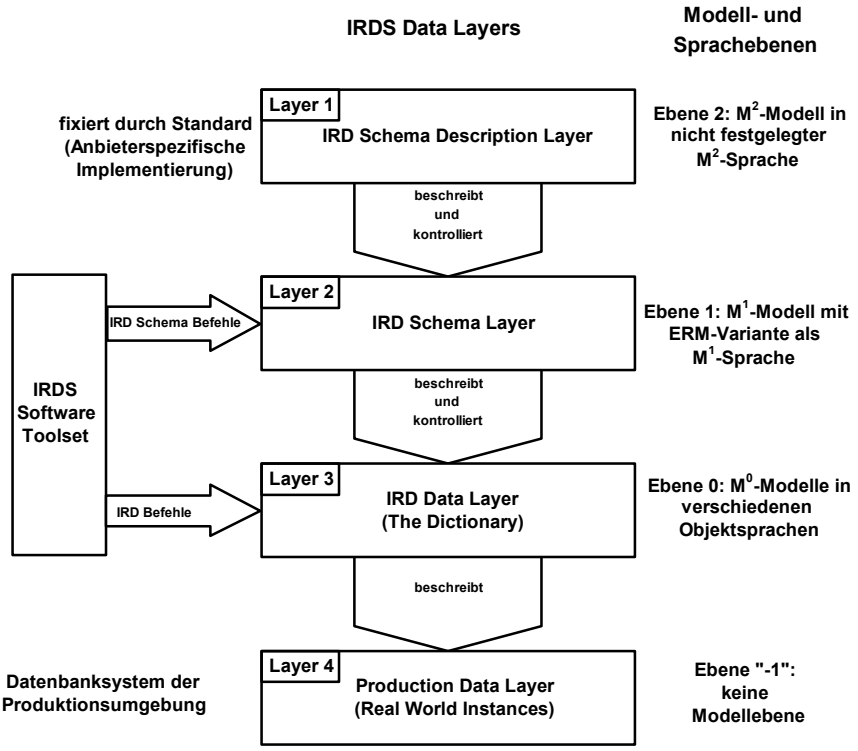



**Abb. 10: Übersicht über den ANSI-IRDS-Standard**
Quelle: in Anlehnung an Stülpnagel (1991, S. 21)

Ein IRDS basiert auf einer Hierarchie von Datenmodellen, die in Abb. 10 dargestellt ist. Es werden vier Ebenen unterschieden, die in der ANSI-Terminologie "Layer" heißen. Die niedrigste Ebene (Layer 4) gehört im hier definierten Sinn nicht zur Modellhierarchie und auch nicht zum Umfang des IRDS, sondern bezeichnet die Ebene der konkreten Ausprägungen, also den Datenbestand der Produktionsumgebung (Produktions- oder

Benutzerdaten). Layer 3 ist die Ebene des Dictionaries, umfaßt also die Daten, welche die Daten der Produktionsumgebung beschreiben. Hier sind die anwendungsspezifischen Modelle abgelegt, beispielsweise Daten- und Funktionsmodelle des interessierenden Realitätsausschnitts. Welche Art von Modellen dies sein können, wird in Layer 2, dem Schema des Dictionaries, festgelegt. Layer 2 definiert also die Sprachen, mittels derer die Modelle des Layer 3 erstellt werden können. Layer 1 schließlich definiert den Datenmodellierungsansatz, im ANSI-Fall eine Variante des ER-Modells, mit dem das Dictionary Schema, also Layer 2, definiert wird. Die Existenz dieser höchsten Ebene ist Voraussetzung für die Pflegbarkeit der darunterliegenden Ebenen. Über sog. IRD-Schema-Befehle kann das IRD-Schema gepflegt werden. Dies entspricht also der Funktionalität, die notwendig ist, um das Repository an konkrete Anforderungen eines CASE-Tools oder Anwenders anzupassen. Über IRD-Befehle wird das eigentliche Dictionary verwaltet. Diese Befehle entsprechen folglich der Funktionalität eines herkömmlichen Dictionary-Systems.[32]

Betrachtet man das IRDS-Architekturmodell aus der Perspektive der im vorangegangenen Kapitel eingeführten Modell- und Sprachebenen, dann stellt sich die Architektur wie folgt dar. Ebene 0 als niedrigste Modellebene entspricht hier dem Layer 3 und beschreibt in Form von Modellen die interessierende Diskurswelt. Diese Modelle sind in ein oder mehreren Sprachen erstellt, die bzgl. ihrer Einordnung in die Modellhierarchie als Objektsprachen zu bezeichnen sind. Modelle dieser Sprachen, also Metamodelle, sind auf Ebene 1 (Layer 2) abgebildet.[33] Diese Metamodelle selbst sind wiederum unter Verwendung einer Sprache, genauer gesagt der Metasprache, erstellt. Im Falle des (ANSI-)IRDS-Modells handelt es sich hierbei um eine ERM-Variante. Diese Metasprache ist vom Standard fest vorgegeben und in Form eines Metametamodells auf Ebene 2 definiert, das nicht pflegbar ist. Folglich muß auch die Sprache, in der es formuliert ist, also die Metametasprache, weder offengelegt noch vereinheitlicht sein. Der Standard fixiert somit die Metasprache und damit auch z.T. das Metametamodell als Modell dieser Sprache, legt aber die Sprache, in der das Metametamodell formuliert wird, also die Metametasprache, nicht fest. Die verwendete MMS kann vom Anbieter einer konkreten IRDS-Implementierung frei gewählt werden und bedarf keiner Offenlegung.[34]

Die vorangegangene Beschreibung der IRDS-Architektur orientiert sich stark an der Terminologie des IRDS-Standards. Dieser betrachtet das Dictionary als eine datenbankbasierte Anwendung, die wie jede Anwendung dieser Art über ein Schema verfügt, das die von der Anwendung zu verwaltenden Daten beschreibt. Daß es sich hierbei allerdings um Metadaten handelt, also selbst wiederum um Schemata, wird zwar nicht ignoriert, ändert aber nichts daran, daß die Benennung der Metaebenen aus Sicht des Dictionaries als zu betrachtendes Softwaresystem erfolgt. Als Metamodell wird daher in der IRDS-Terminologie Layer 1 bezeichnet.[35] Dieser Aspekt wird in Kapitel 6 unter dem Gesichtspunkt der Verankerung einer Modellhierarchie wieder aufgegriffen.

## 5.2 Metamodelle im Zusammenhang mit Compiler-Compilern

Das folgende aus einem mit Metamodellierung nicht unmittelbar verbundenen Themenbereich stammende Beispiel verdeutlicht, daß es in einer Modellhierarchie notwendig sein kann, ein- und dieselbe Sprache auf der darüberliegenden Modellebene in mehreren Modellen abzubilden. In diesen verschiedenen Modellen desselben Originals werden jeweils unterschiedliche Aspekte der modellierten Sprache betrachtet, und die Modelle selbst sind dann meist auch in unterschiedlichen Sprachen formuliert.

Compiler-Compiler (CC), auch Meta-Compiler oder Compiler-Generatoren genannt, sind Werkzeuge, welche die langwierige Aufgabe des Schreibens von Compilern beschleunigen sollen. CC-Sprachen können als spezielle Form von Programmiersprachen verstanden werden, die dem Zweck der Beschreibung und Übersetzung anderer Programmiersprachen gerecht werden. Die Idee besteht also darin, auch im Compilerbau wie zuvor in anderen Gebieten die dort verbreitete Assemblersprache durch eine höhere Programmiersprache abzulösen.[36]

Abb. 11 veranschaulicht die Funktionsweise eines Compiler-Compilers. Ein CC verwendet die Definition einer Programmiersprache P als Input und liefert einen Compiler für dies Sprache als Output. Der so erzeugte Compiler verwendet P-Programme (in P geschriebene Programme) und übersetzt diese in gleichwertige Programme in einer Maschinensprache (MSP).

---

32 Vgl. Habermann, Leymann (1993, S. 77-85), Winkler (1989, S. 10-12), Stülpnagel (1991).

33 Man beachte allerdings, daß das Schema eines Dictionary-Systems in der Regel auch Strukturen zur Verwaltung nichtmodellbezogener Informationen zur Verfügung stellt. Dies können beispielsweise die Projektorganisation betreffende Informationen sein. Berücksichtigt man diesen Aspekt, so hat lediglich ein Ausschnitt des Dictionary-Schemas Metamodellcharakter.

34 Vgl. Winkler (1989, S. 10).

35 Vgl. Habermann, Leymann (1993, S. 85).

36 Vgl. Hopgood (1970, S. 149-150), Zima (1989, S. 40), Tremblay, Sorenson (1985, S. 722-723), Gries (1971, S. 436).

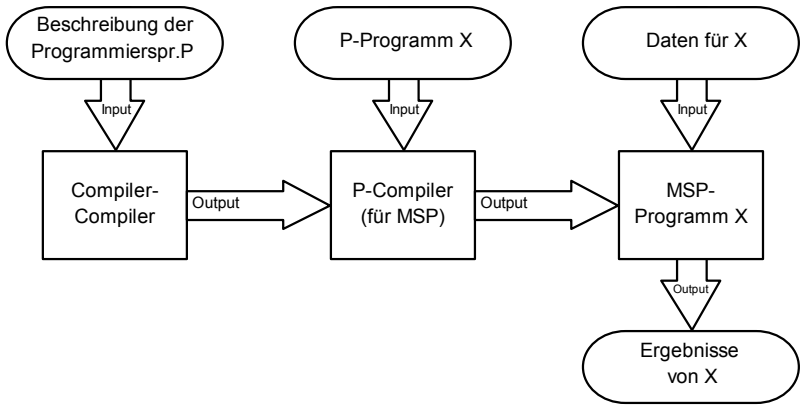



**Abb. 11: Prinzip des Compiler-Compilers**
Quelle: in Anlehnung an Hopgood (1970, S. 151)

In der Terminologie der Metamodellierung läßt sich der Sachverhalt wie in Tab. 2 dargestellt beschreiben. Man beachte, daß sich der Prozeß der Kompilierung selbst ausschließlich auf einer Modellierungsebene abspielt. Er übersetzt das Programm X aus einer Objektsprache (nämlich P) in eine andere Objektsprache[37] (nämlich die Maschinensprache (MSP)). Die CC-Sprache ist, sieht man zunächst von den zu spezifizierenden Eigenschaften der Zielrechenanlage ab oder geht von einem maschinenabhängigen CC aus, allerdings nur bezüglich P Metasprache.

| | **Modelle** | **Sprachen** |
|---|---|---|
| **Ebene 1** | Beschreibungsmodell von P (MM) | in CC-Sprache (MS) |
| **Ebene 0** | Programm X (M) | in P (OS) |

**Tab. 2: Ebenen der Modellierung bei Erzeugung eines P-Compilers unter Verwendung eines Compiler-Compilers**

Die bisherige Darstellung eines Compiler-Compilers ist stark idealisiert. Konkrete Produkte in diesem Bereich generieren in der Regel lediglich Teile eines Compilers, die genau eine Phase des Kompiliervorgangs unterstützen, z.B. das Scannen, Parsen, Codegenerieren oder -optimieren. Entsprechend handelt es sich bei dem Meta-Compiler z.B. um einen Scanner- oder Parser-Generator. Erst durch Integration dieser Teil-CC ($CC_1$, $CC_2$, ..., $CC_n$) entsteht ein vollständiger Compiler-Compiler. Die Gliederung eines CC in mehrere Teil-CC ist auch insofern sinnvoll, als daß die Sprache P, für die ein Compiler generiert werden soll, nicht in einem einzigen Modell beschreibbar ist. Für die verschiedenen Teil-CC sind verschiedene Beschreibungsmodelle der Sprache notwendig, insbesondere eine Syntax- und eine Semantikbeschreibung. Üblicherweise werden für die Syntax- und Semantikdefinition verschiedene Sprachen verwendet, d.h. man benötigt mehrere Metamodelle in verschiedenen Metasprachen ($CC_1$-, $CC_2$-, ... , $CC_n$-Sprache).[38] Die Syntaxbeschreibung erfolgt oft in einer BNF-ähnlichen Notation, die Semantikdefinition in einer höheren Programmiersprache in Form von semantischen Routinen.[39]

Geht man davon aus, daß der CC Compiler für verschiedene Maschinentypen erzeugt, also maschinenunabhängig ist, so sind auch Eigenschaften der Rechenanlage zu spezifizieren, d.h. der CC (Teil-CC für die Codegenerierung) benötigt auch ein Beschreibungsmodell der Maschinensprache, "a formalized description of the machine"[40], in der der Code erzeugt werden soll. In diesem Sinne wäre eine weitere Metasprache notwendig, um die Zielmaschine in ihren Strukturen und ihrem Verhalten zu beschreiben.[41] Tab. 3 gibt einen Überblick über die Modellierungs- und Sprachebenen bei Verwendung eines solchen maschinenunabhängigen CC.

---

37 Man beachte, daß der Begriff Objektsprache auch in diesem Zusammenhang wie zuvor definiert benutzt wird und im Themenbereich der Sprachübersetzung nicht mit dem Begriff Objekt-Code verwechselt werden sollte.

38 Vgl. Hopgood (1970, S. 151-158).

39 Letztlich handelt es sich um manuell programmierte Routinen, die vom CC an der entsprechenden Stelle in den Compiler eingebunden werden. Vgl. Tremblay, Sorenson (1985, S. 723), Gries (1971, S. 436-437), Hopgood (1970, S. 158).

40 Tremblay, Sorenson (1985, S. 723).

41 Beispiele für solche Sprachen sind TMDL (target machine description language) im Production-Quality Compiler-Compiler System und CGGL (code generator generator language), bei der die zu modellierende Maschine als endlicher Automat betrachtet wird. Man beachte, daß im Gegensatz zu diesem deskriptiven Ansatz auch hier wiederum die

| | Modelle | Sprachen |
|---|---|---|
| **Ebene 1** | Beschreibungsmodelle von P u. MSP:<br>Syntaxmodell von P ($MM_1$ bzgl. $M_1$)<br>Semantikmodell von P ($MM_2$ bzgl. $M_1$)<br>Modell der MSP ($MM_1$ bzgl. $M_2$) | <br>in $CC_1$-Sprache ($MS_1$ bzgl. $OS_1$)<br>in $CC_2$-Sprache ($MS_2$ bzgl. $OS_1$)<br>in $CC_3$-Sprache ($MS_1$ bzgl. $OS_2$) |
| **Ebene 0** | Programm X ($M_1$)<br>Programm X ($M_2$) | in P ($OS_1$)<br>in MSP ($OS_2$) |

**Tab. 3: Ebenen der Modellierung bei der Erzeugung eines P-Compilers mit MSP als Zielsprache**

Am Beispiel eines CC wurde in diesem Kapitel die Anwendbarkeit des vorgestellten Metamodellbegriffs in einem mit diesem Begriff selten verbundenen Gebiet demonstriert.

### 5.3 Vergleichende Betrachtung der Beispiele

Wendet man das Stachowiak´sche Frage-Quadrupel auf die zwei vorgestellten Beispiele an, so wird deutlich, daß insbesondere das Modellsubjekt und die Modellierungszeit über die Ebenen hinweg stark variieren. Tab. 4 gibt einen entsprechenden Überblick. Bei der Betrachtung des IRDS wurde die Ebene der Ausprägungen, hier als Ebene "-1" bezeichnet, hinzugezogen, da sich einige der Prinzipien der Modellebenen in diese Ebene hinein fortsetzen.[42]

| Ebene | Wovon ? (Modelloriginal) | Für wen ? (Modellsubjekt, Modellierer) | Wann ? (Modellierungszeit) | Wozu ? (Modellzweck) |
|---|---|---|---|---|
| | **Beispiel:** | **I R D S** | | |
| **"-1"** | | Benutzer der Software | Laufzeit, Betriebszeit der Software | |
| **0** | dv-technisch zu unterstützende Miniwelt | Systemanalytiker, Software-Ingenieur des IRDS-Nutzers | zum Zeitpunkt der Anwendungsentwicklung | Erstellung einer (datenbankgestützten) Anwendung |
| **1** | Objektsprachen zur Systembeschreibung im Rahmen der Softwareentwicklung | Methodenentwickler, Methodeningenieur des IRDS-Anbieters und/ oder -Nutzers | zum Zeitpunkt d. IRDS-Erstellung und/oder zum Zeitpunkt der Einführung beim IRDS-Nutzer | Vereinheitlichung u. Formalisierung der verwendeten Objektsprachen |
| **2** | Metasprache bzgl. Objektsprachen zur Systembeschreibung im Rahmen der Softwareentwicklung | Standardisierungsorganisation (ANSI, ISO), in der konkreten Realisierung IRDS-Hersteller | zum Zeitpunkt der Standardisierung bzw. zum Zeitpunkt der IRDS-Entwicklung | Vereinheitlichung u. Formalisierung der verwendeten Metasprache(n) |
| | **Beispiel:** | **Compiler-Compiler** | | |
| **0** | dv-technisch zu unterstützende Miniwelt | Programmierer, Compiler | vor "compile time", zur "compile time" | Programm- u. Maschinencodeerstellung |
| **1** | Programmiersprache, Maschinentyp | Programmiersprachen-entwickler, Compilerbauer | vor "meta compile time"[43] | Compilerbau |

**Tab. 4: Anwendung des Stachowiak´schen Frage-Quadrupels auf konkrete Modellhierarchien**

Die Tabelle verdeutlicht, daß sich die Modellebenen sehr gut durch die Kriterien Modellsubjekt und Zeit abgrenzen lassen. Die Modelle auf jeder der Ebenen werden von einer grundsätzlich anderen Gruppe von

---

Möglichkeit der Einbindung von in einer prozeduralen Programmiersprache manuell programmierter Routinen besteht, bzw. daß die zuvor erwähnten semantischen Routinen die maschinenspezifischen Aspekte umfassen. Vgl. Tremblay, Sorenson (1985, S. 742, 757, 763), Gries (1971, S. 437-439).

42 Bei Betrachtung der Tabelle ist zu beachten, daß die Kriterien Modellsubjekt und Zeit interpretierungsbedürftig sind. Das Modellsubjekt kann zum einen der Modellersteller sein und zum anderen derjenige, der zu einem späteren Zeitpunkt das Modell verwendet, um auf dessen Grundlage ein Modell der darunterliegenden Ebene zu erstellen. Für die Tabelle wurde von ersterem ausgegangen also vom Modellierer und vom Zeitpunkt der Modellerstellung. Die Verwendung des zweiten Vorschlags würde lediglich zu einer Verschiebung der Einträge um eine Zeile nach unten führen. Auch beim zeitlichen Aspekt sind verschiedene Interpretationen möglich, wie der Zeitpunkt der Gültigkeit, der Zeitpunkt der Benutzung, der Zeitpunkt der Erstellung. In Analogie zum Modellsubjekt wurde hier vom Zeitpunkt der Erstellung ausgegangen.
Der Modellzweck läßt sich insbesondere auf der Ebene 0 nur sehr allgemein formulieren, wäre aber bei Berücksichtigung einer konkreten Aufgabenstellung wesentlich präziser anzugeben.
Zur " -1"- Ebene ist anzumerken, daß es sich bei dieser nicht um eine Modellebene handelt und deshalb außer der Fragestellung nach der Entstehungszeit und dem Ersteller die anderen Kriterien nicht beantwortet werden können.

43 Die Begriffe "compile" vs. "meta compile time" stammen von Gries (1971, S. 437).

Personen (z.T. auch künstlichen Subjekten im Falle des Compilers) erstellt, wobei sich auch die Zeitpunkte der Modellierung unterscheiden. Bezüglich des zeitlichen Aspekts ist dies auch insofern einsichtig, als daß die Modelle der höheren Ebenen Voraussetzung sind für die Modellerstellung auf den unteren Ebenen.

Die beiden Beispiele für Modellhierarchien weisen in einigen Bereichen Ähnlichkeiten auf, in anderen unterscheiden sie sich grundsätzlich. Dies wirft die Frage nach typischen Eigenschaften solcher Modellhierarchien auf, die im folgenden Kapitel behandelt werden sollen.

# 6 Allgemeine Merkmale sprachbasierter Modellhierarchien

Die beschriebene Art der Modellebenenbildung führt zu einer Hierarchie von Modellen. Solche Modellhierarchien können sehr verschieden beschaffen sein und unterschiedliche Eigenschaften aufweisen. Im folgenden sollen grundlegende Merkmale, die sich aus der Ebenenbildung ergeben, vorgestellt werden. Anhand der Ausprägungen dieser Merkmale lassen sich Modellhierarchien grundlegend charakterisieren und - da einige der Merkmale Anforderungscharakter haben - beurteilen.

## 6.1 Eigenschaften der Inter-Ebenen-Beziehungen

Im folgenden sollen zunächst diejenigen Eigenschaften formuliert werden, die das Verhältnis der Ebenen zueinander betreffen, also die Beziehungen zwischen den Ebenen, die hier als Inter-Ebenen-Beziehungen bezeichnet werden. Die wesentliche Eigenschaft betrifft die *Art* dieser Beziehung. Im Falle des IRDS-Beispiels handelt es sich um eine Instantiierungsbeziehung, wobei sich eine Ebene bzgl. ihrer übergeordneten als Extension und bezüglich ihrer untergeordneten als Intension verhält. In der Terminologie der Informatik kann man eine Ebene auch als Typsystem für die darunterliegende bezeichnen. Das IRDS-Beispiel verdeutlicht zudem auch, daß die Art der Inter-Ebenen-Beziehung über verschiedene Ebenen hinweg homogen sein kann. Bei mehr als zwei Modellierungsebenen kann also die *Homogenität* der Inter-Ebenen-Beziehungen über die Ebenen hinweg betrachtet werden.

Die nächste grundlegende Eigenschaft, welche die klare Trennung der Modellebenen gewährleistet, betrifft die *Abgrenzung der Inter- von Intra-Ebenen-Beziehungen*. Werden Beziehungen, die zwischen Objekten verschiedener Modellebenen bestehen, nicht unterschieden von solchen, die zwischen Objekten derselben Ebene bestehen, so können die Modellebenen selbst nicht klar voneinander abgegrenzt werden. Es sind grundsätzlich zwei Möglichkeiten denkbar, wie diese Abgrenzung gewährleistet werden kann.

Die erste Möglichkeit besteht darin, daß Inter- und Intra-Ebenen-Beziehungen artverschieden sind. Beispielsweise könnten als Intra-Ebenen-Beziehungen Generalisierungen, Aggregationen und Assoziationen zulässig sein, als Inter-Ebenen-Beziehungen Instantiierungen. Die zweite Möglichkeit muß benutzt werden, wenn die Eigenschaft der Artverschiedenheit nicht gegeben ist. In einem solchen Fall müssen Inter-Ebenen-Beziehungen explizit gekennzeichnet werden, entweder unmittelbar dadurch, daß sie selbst als solche gekennzeichnet werden, oder mittelbar dadurch, daß die betrachteten Modellelemente eindeutig Ebenen zuordenbar sind.

Als Beispiel für Vertreter der Auffassung, daß eine Artverschiedenheit zwischen Inter- und Intra-Ebenen-Beziehungen nicht gegeben sein muß, sind Tardieu, Franckson (1987) zu nennen. Sie vertreten die Ansicht, daß Subtypenbildung und Instantiierung die gleiche Art von Beziehung bezeichnen. Sie nennen Ansätze, bei denen auf eine Ebenentrennung verzichtet wird, endogene Ansätze und solche, bei denen explizit Ebenen unterschieden werden, exogen.[44]

Eine ähnliche Auffassung ist bei Martin, Odell (1992) anzutreffen.[45] Als Vertreter eines sog. "Single Framework"-Ansatzes wollen sie Instantiierungsbeziehungen grundsätzlich auf Modellebene und nicht nur zwischen

---

44 Vgl. Tardieu, Franckson (1987, S. 543, 550). Als Beispiele für exogene Ansätze, bei denen die Modellierungsebenen über Spezialisierungen untereinander verbunden sind, sind die Arbeiten von Potts (1989) und Oivo, Basili (1992) zu nennen. Insbesondere der Ansatz von Potts (1989, S. 218) bereitet bzgl. seiner Einordnung Schwierigkeiten. Er geht von einem sog. generischen Schema, das methodenunabhängig ist, aus. Über einen Prozeß, den er als "customization" bezeichnet, entsteht aus dem generischen Schema ein methodenspezifisches Schema, und zwar dadurch, daß die Objekttypen des ursprünglichen Schemas spezialisiert werden. Durch Instantiierung schließlich entsteht dann ein konkretes problembezogenes Modell. Bei diesem handelt es sich eindeutig um ein Modell der Ebene 0, und das methodenspezifische Schema ist ein Metamodell. Unklar bleibt, wie das generische Schema einzuordnen ist: als Metamodell oder als Kernmetamodell, das situationsabhängig anzupassen ist. Bei Potts liegt also die Tendenz vor, eine Intra-Ebenen-Beziehung (Spezialisierung) als Inter-Ebenen-Beziehung zu nutzen. Genau das gegenteilige Anliegen ist bei Martin, Odell (1992) anzutreffen, die eine Inter-Ebenen-Beziehung (Instantiierung) auch innerhalb einer Ebene benutzen wollen.

45 Vgl. Martin, Odell (1992, S. 489-493), Odell (1992).

verschiedenen Ebenen zulassen:[46] "Why have .. boundaries at all ?"[47] Ihr Hauptargument, auf das sie ihren Ansatz stützen, ist im Bereich der Implementierungstechnologien zu sehen, wo die Ebenenbildung zu einem erheblichen Verlust an Flexibilität führt. Eine dort oftmals vorzufindende "rigid physical separation"[48] wird von ihnen kritisiert, ist aber keineswegs zwingendes Resultat einer Trennung von Modellebenen auf konzeptioneller Ebene, sondern vielmehr ein dv-technisches Problem.

Grundsätzlich ist es also möglich, auf die Trennung der Ebenen zu verzichten und Objekte der Metaebene mit Objekten der Modellebene in ein- und demselben Modell abzubilden. Solche Ansätze sind insbesondere im Bereich der Konzeption objektorientierter und noch deutlicher im Bereich reflektiver Programmiersprachen[49] anzutreffen und tragen dort zu der von Martin, Odell (1992) geforderten Flexibilität bei.

## 6.2 Population der Modellhierarchie

Unter der Population der Modellhierarchie wird die "Besiedlung" der einzelnen Ebenen mit mehreren Modellen verstanden. Grundsätzlich gilt, daß die Population auf tieferen Ebenen zunimmt. Oft sind Hierarchien so gestaltet, daß an der Spitze der Modellhierarchie (Ebene n) genau ein Metamodell steht (oder eine natürlich-sprachliche Beschreibung der Metasprache der unmittelbar darunterliegenden Ebene), dieses aber bereits auf der nächsten Ebene mehrmals instantiiert werden kann, also auf Ebene n-1 mehrere Modelle existieren, die als Extension der darüberliegenden Ebene verstanden werden können. Eine weitere Ebene tiefer führt dies auch bei einfacher Instantiierung jedes übergeordneten Modells zu mehreren Modellen. Zusätzlich kann aber auch hier wieder eine mehrfache Instantiierung anzutreffen sein. Abb. 12 zeigt, welche Gestalt eine solche Modellhierarchie annehmen kann.

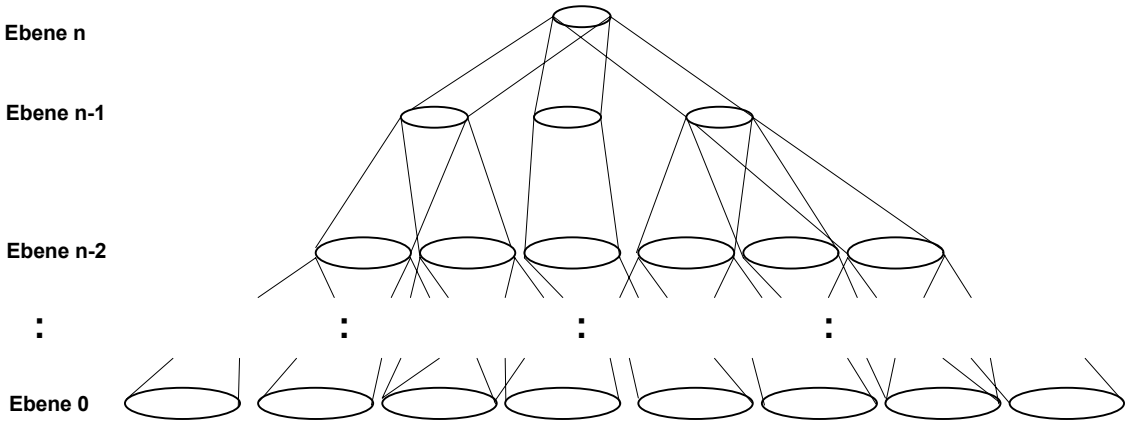



**Abb. 12: Modellhierarchie mit stark populierten Modellebenen**

Daß solche Strukturen nicht nur bei Hierarchien auftreten können, bei denen die Intra-Ebenen-Beziehung einer Instantiierung entspricht, zeigt das CC-Beispiel. In Abb. 13 wurde für dieses Beispiel in Entsprechung zu den in Tab. 3 abgebildeten Sprach- und Modellebenen die sich ergebende Modellhierarchie dargestellt. Bei diesem Beispiel fällt zudem auf, daß zwischen Modellen der Ebene 0 und denen der Ebene 1 nicht immer eine eindeutige Zuordnungsbeziehung besteht (wie im rechten Teil von Abb. 13), sondern auch mehrdeutige Beziehungen (wie im linken Teil von Abb. 13) vorkommen können. Eine solche Situation tritt insbesondere immer dann auf, wenn die Sprache einer Ebene i auf Ebene i+1 zweisprachig (im Beispiel in $CC_1$- und $CC_2$-Sprache) modelliert wird.

---

46 Der Unterschied dieser Auffassung zum endogenen Ansatz nach Tardieu, Franckson (1987) ist also darin zu sehen, daß Martin, Odell (1992) durchaus zwischen Subtypenbildung und Instantiierung unterscheiden, aber keine Notwendigkeit darin sehen, die Instantiierung zu einer Inter-Ebenen-Beziehung zu machen.

47 Odell (1992, S. 46).

48 Odell (1992, S. 46).

49 Auf den Themenbereich reflektiver Programmiersprachen soll hier nicht detaillierter eingegangen werden. Statt dessen wird auf Maes (1987), Foote, Johnson (1988), Ferber (1988), Danforth, Forman (1994) verwiesen.

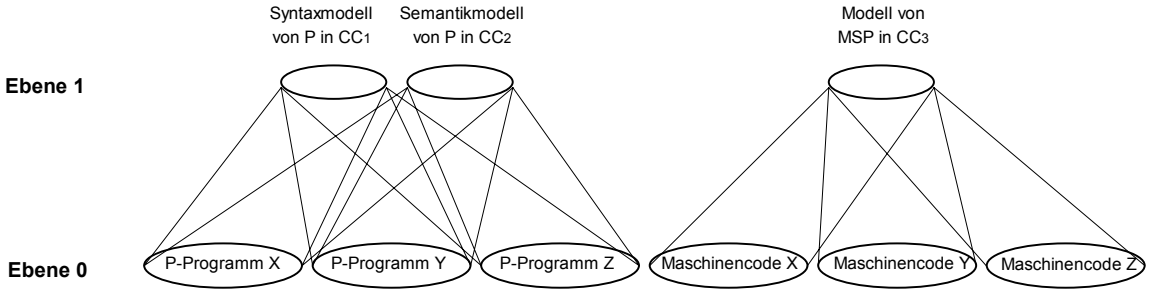



**Abb. 13: Population der CC-Modellhierarchie bei drei Programmen auf Ebene 0**

Die vorangegangenen exemplarischen Überlegungen werfen die Frage nach grundlegenden Eigenschaften auf, die eine Modellhierarchie bzgl. ihrer Population charakterisieren. Daß es in der Regel je Modell einer übergeordneten Ebene mehrere Modelle der darunterliegenden Ebene gibt, liegt im Wesen der Modellhierarchie an sich. Die entgegengesetzte Betrachtungsrichtung, nämlich die Frage danach, ob es zu einem Modell einer Ebene auf der darüberliegenden ein oder mehrere Modelle gibt, ist ein wesentlich besser geeignetes Kriterium, um Modellhierarchien zu typisieren. In diesem Sinne kann zwischen zwei *Populationsarten*, nämlich Ein- und Multi-Modell-Ansätzen unterschieden werden. Bei Multi-Modell-Ansätzen stellt sich die Frage nach dem Grund für die Beschreibung in mehreren Modellen. Als Möglichkeiten bieten sich die Wahl verschiedener Verkürzungsmerkmale und bzw. oder verschiedener Sprachen an. Unterscheiden sich die zu einem Modell formulierten Metamodelle lediglich bzgl. ihres Verkürzungsmerkmales nicht jedoch in der Sprache, so handelt es sich um einen einsprachigen Ansatz. Sollen bei der Modellierung einer Sprache grundsätzlich verschiedene Abstraktionen vorgenommen werden, so bietet es sich auch meist an, zur Modellformulierung spezifische Sprachen zu verwenden. In einem solchen Fall liegt dann also ein mehrsprachiger Multi-Modell-Ansatz vor.

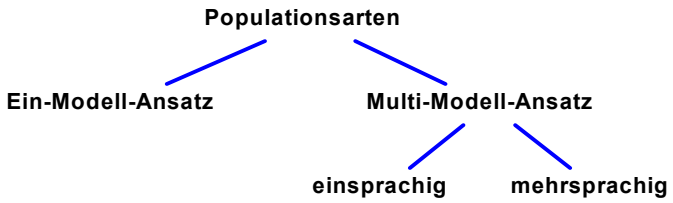



**Abb. 14: Arten der Population einer Modellhierarchie**

Abb. 14 gibt einen zusammenfassenden Überblick über die unterschiedenen Formen. Für die zwei betrachteten Beispiele folgt, daß es sich bei der IRDS-Hierarchie um einen Ein-Modell-Ansatz handelt. Bei der CC-Hierarchie hingegen liegt ein mehrsprachiger Multi-Modell-Ansatz vor.

## 6.3 Umfang und Abgrenzung der Modellhierarchie

Ein wesentliches Merkmal einer Modellhierarchie ist in der Zahl der sie konstituierenden Ebenen zu sehen. Die *Ebenenanzahl* ist meist klein. Häufig umfaßt eine Hierarchie wie im Falle des CC-Beispiels zwei Ebenen, manchmal drei Ebenen wie im Falle des IRDS-Beispiels. Die Ebenenanzahl wird bestimmt von der Begrenzung der Hierarchie nach oben und unten. Auf beide Aspekte soll im folgenden eingegangen werden.

Wie im Kapitel zu Objekt- und Metasprachen betont, handelt es sich bei diesen Begriffen um relative Ausdrücke. Entsprechendes gilt für die Begriffe Modell und Metamodell. Deshalb ist es besonders wichtig, daß die Ebene 0, also die niedrigste betrachtete Modell- bzw. Sprachebene, klar definiert wird. In diesem Zusammenhang wird von der *Verankerung* der Modellhierarchie gesprochen. Es ist nicht zwingend, daß die niedrigste Sprachebene ein Modell außersprachlicher Gegenstände darstellt, aber üblich. Man beachte, daß es Autoren gibt, die die Verankerung eine Ebene höher ansetzen, z.B. im IRDS-Bereich. Ihr Metamodellbegriff entspricht dann dem hier verwendeten Metametamodellbegriff. Die Verankerung der Modellhierarchie sollte klar definiert sein. Zudem muß gelten, daß es sich bei der niedrigsten Modellebene bereits um eine solche handelt. Die Verankerung darf also nicht bereits auf einer Ebene angesiedelt werden, die noch keine Modellebene darstellt.

Eine solche "-1"-Ebene wurde im IRDS-Beispiel angesprochen, nämlich die Ebene der Ausprägungen in der Produktionsumgebung. Dem hier zugrundegelegten Modellbegriff zufolge handelt es sich dabei um kein Modell. Auffallend ist dennoch, daß sich die Homogenität der Inter-Ebenen-Beziehungen auf diese Ebene ausdehnen läßt.

Ebenso wie die Modellhierarchie nach unten abgeschlossen ist, sollte auch eine *Terminierung* nach oben gefordert werden. Die Ebenenbildung ist zwar grundsätzlich nach oben offen, könnte also prinzipiell beliebig fortgeführt werden. Jedoch ist es in konkreten Situationen wenig hilfreich, davon auszugehen, eine Modellhierarchie als nach oben nicht abgeschlossen zu betrachten. Wird eine Modellhierarchie nicht terminiert, so hat man das Problem der Definition einer Modellebene lediglich um eine oder mehrere Ebenen nach oben verschoben, aber nicht endgültig gelöst. Zemanek (1966, S. 140) sieht wie viele andere Autoren die Lösung darin, "that on some points we will ever have to step back to our ultimate metalanguage, i.e., to English". Als ultimative Sprache wird häufig Alltagssprache vorgeschlagen, die erweitert werden kann um einige fachsprachliche Termini oder Symbole und in Bereichen auch zu präzisieren ist.[50]

## 7 Zusammenfassung und Ausblick

Das wesentliche Merkmal der in der Literatur eingeführten Metamodellbegriffe ist deren entweder sehr weite, meist unpräzise Defintion oder deren zu enge Eingrenzung auf einen bestimmten Kontext.[51] Obwohl der Begriff häufig benutzt wird, ist er selten Gegenstand einer ausführlichen Betrachtung. In der vorliegenden Arbeit stand dies deshalb im Vordergrund, wobei versucht wurde, den Begriff weder zu weit noch zu eng zu fassen. Anhand der vorgestellten Merkmale kann in einem konkreten Kontext durch Beschreibung der Merkmalsausprägungen eine Präzisierung vorgenommen werden. Welche Kontexte dies sein können, also welche Einsatzgebiete der Metamodellierung existieren bzw. denkbar sind, wurde bislang nicht betrachtet. Auf diese soll daher hier ein kurzer abschließender Ausblick gegeben werden.

Das wesentliche Einsatzgebiet der Metamodellierung ist in den methodologischen Bereichen der Informatik, insbesondere der Wirtschaftsinformatik, zu sehen. Dieses Gebiet läßt sich in seiner Gesamtheit auch als Methoden-Engineering bezeichnen, worunter in Analogie zum Software-Engineering die ingenieurmäßige Entwicklung von Softwareentwicklungsmethoden verstanden werden kann,[52] oder nach Heym (1995) der "*systematische und und strukturierte Prozeß zur Entwicklung, Modifikation und Anpassung von Software-Entwicklungsmethoden durch die Beschreibung der Methodenkomponenten und ihrer Beziehungen*".[53] Im Vordergrund des Methoden-Engineerings stehen also Softwareentwicklungsmethoden, und zwar deren Beschreibung, Entwicklung, Bewertung, vergleichende Gegenüberstellung, Integration, Anpassung, Harmonisierung und Standardisierung. Der Einsatz von Metamodellierung ist in all diesen Bereichen naheliegend. Beispiele für Arbeiten, die Metamodellierung als wichtiges Element nutzen, sind z.B. in folgenden Gebieten anzutreffen.

**Beschreibung von Methoden und Informationssystemarchitekturen**

Dieser bereits in der Einleitung betrachtete Bereich ist in der Wirtschaftsinformatik als das klassische Gebiet der Metamodellierung anzusehen. Neben der metamodellbasierten Beschreibung phasenübergreifender Ansätze[54] wird Metamodellierung hier auch im Kleinen eingesetzt. Bei der Beschreibung von Entwicklungsmethoden werden Modellierungsansätze häufig auf sich selbst angewandt. Metasprache und Metametasprache stimmen also überein. Solche Metamodelle stellen somit sowohl Modellierungsbeispiele dar wie auch eine formalisierte Darstellung zusätzlich zur natürlich-sprachlichen Beschreibung des Ansatzes. Als Beispiele sind hier Embley, Woodfield, Kurtz (1992) und Coad, Yourdon (1991) zu nennen. Der Detaillierungsgrad kann dabei stark variieren und ist insbesondere dann sehr gering, wenn das Metamodell wie beispielsweise bei Coad, Yourdon (1991, S. 205)[55] lediglich exemplarischen Charakter hat und nicht integraler Bestandteil der Methodenbeschreibung ist. Einen wesentlichen Anteil an den Methodenbeschreibungen haben Metamodelle insbesondere bei den bereits betrachteten phasenübergreifenden Ansätzen und Informationssystemarchitekturen. Die herausragende Bedeutung der Metamodellierung ist in diesem Bereich darin zu sehen, daß die Vielzahl zu beschreibender Modellierungssprachen einheitlich und unter Berücksichtigung gegenseitiger Abhängigkeiten modelliert werden kann. Dies ist unter anderem auch für die Automatisierung des Transformationsprozesses

---

50 Vgl. Stegmüller (1974, S. 32), Schreiber (1960, S. 16). Zur Begründung der Eignung der Umgangssprache als ultimative Sprache siehe Kleinknecht, Wüst (1976, S. 42).

51 Bezüglich der verwendeten Begrifflichkeiten herrscht - was auch Tab. 1 zeigt - allerdings keine große Heterogenität. Das Wort Metamodell ist durchaus verbreitet. Alternativ wird gelegentlich von Metaschema oder Informationsmodell gesprochen. Insbesondere der letzte Begriff, der z.B. von Mercurio et al. (1990), Merbeth (1991) und in der AD/Cycle-Literatur im Sinne von Metamodell benutzt wird, ist in der Informatik mißverständlich und sollte daher vermieden werden.

52 Vgl. Gutzwiller (1994, S. 11).

53 Heym (1995, S. 4).

54 Hier sind als Beispiele die in der Einleitung bereits erwähnten Arbeiten zu nennen: Albers (1995), Gutzwiller (1994), Olle et al. (1991) ), Scheer (1992).

55 Siehe Beispiel 6 in Abb. 3.

über die Phasen hinweg von Bedeutung. Wird der Transformationsprozeß in der Richtung umgekehrt, bieten sich zudem Möglichkeiten für ein z.B. von Kaufmann (1994) vorgestelltes metamodellbasiertes Reengineering.

**Methodenvergleiche**

Benutzt man eine einheitliche Metasprache zur Modellierung verschiedener Modellierungsansätze, so stellt dies eine gute Grundlage für eine vergleichende Betrachtung dar.[56]

**Methoden-Werkzeug-Kompatibilität**

Durch den Vergleich der Datenstrukturen eines Entwicklungswerkzeuges mit den Metamodellen einzelner Methoden kann ermittelt werden, inwieweit Werkzeuge geeignet sind, bestimmte Methoden zu unterstützen. Ähnlich wie beim Methodenvergleich ist hier der Vorteil in der einheitlichen Metasprache zu sehen.[57]

**Methodenharmonisierung**

In Anbetracht der Vielfalt herrschender Entwicklungsmethoden und der damit verbundenen zahlreichen Modellierungssprachen, ist es nicht verwunderlich, daß Versuche unternommen werden, die grundsätzliche Vereinbarkeit verschiedener Modellierungsansätze zu begründen. Hierzu können Metamodelle ausgearbeitet werden, die den gemeinsamen Kern eines gegebenen Spektrums von Modellierungsmethoden darstellen.[58]

Da im Mittelpunkt des Methoden-Engineerings *Software*-Entwicklungsmethoden stehen, sind zwei Aspekte besonders naheliegend:
- die entwickelten Methoden sind durch Werkzeuge zu unterstützen und
- der Methoden-Entwicklungsprozeß selbst ist durch Werkzeuge zu unterstützen.

Der erste Bereich, nämlich die Unterstützung der Methoden durch Werkzeuge, stellt das traditionelle Gebiet der Metamodellierung in der Informatik dar. Diesem Bereich sind auch die in Kapitel 5 vorgestellten Beispiele zuzuordnen. Im Vordergrund stehen hier insbesondere Repositories, CASE- und Meta-CASE-Systeme, auch CASE-Shells genannt. Die Systeme, die dem zweiten Bereich zuzuordnen sind, kann man allgemein als CAME-Systeme (Computer Aided Methodology Engineering) bezeichnen. DV-technisch handelt es sich bei diesen zwar meist um herkömmliche Meta-CASE- oder sogar nur CASE-Systeme; sie werden allerdings mit einer anderen Zielsetzung eingesetzt, nämlich zur Modellierung von Methoden und nicht zur Modellierung einer betrieblichen Diskurswelt.[59]

Auch bei solchen Software-Systemen, die nicht unmittelbar Werkzeugcharakter haben, also nicht direkt der Software*entwicklung* dienen, aber dennoch in bestimmten Bereichen generischer Natur sind, kann Metamodellierung dazu beitragen, gerade diese Generizität herbeizuführen.[60] Als Beispiele sind hier Workflow-Systeme und betriebswirtschaftliche Standardsoftware zu nennen.

Als letztes Einsatzgebiet der Metamodellierung soll hier das Gebiet des Software-Engineerings erwähnt werden, das sich mit dessen quantitativen Aspekten beschäftigt. Metamodelle lassen sich hier beispielsweise im Bereich der Ermittlung von Software-Metriken und anderer quantitativer Maßzahlen sinnvoll einsetzen.[61] Auch Anwendungsmöglichkeiten im Bereich der Aufwandsschätzung zeichnen sich ab. Dies gilt insbesondere für sog. Meta-Schätzverfahren, die Anleitung zur Konstruktion vorgehensmodellspezifischer Schätzverfahren geben.[62]

Die genannten Beispiele verdeutlichen die Breite und Offenheit des Spektrums möglicher Einsatzgebiete.[63] Der Vorteil der Metamodellierung ist in vielen dieser Gebiete darin zu sehen, daß bestimmte Aspekte explizit

---

56 Färberböck, Gutzwiller, Heym (1991), Hess, Brecht (1995), Goor, Hong, Brinkkemper (1992), Hong, Goor, Brinkkemper (1993) sind Beispiele für Arbeiten aus diesem Gebiet.

57 Dieser Aspekt steht in Brinkkemper et al. (1990), Steele, Zaslavsky (1994) im Vordergrund.

58 Als Beispiel ist hier Carmichael (1994) zu nennen.

59 Arbeiten zu Meta-CASE sind beispielsweise Alderson (1991), Verhoef, Hofstede, Wijers (1991), Pocock (1991), Oquendo, Zucker, Griffiths (1992), entsprechend zu CAME Smolander et al. (1991), Rossi et al. (1992), Harmsen, Brinkkemper, Oei (1994) und Heym (1995).

60 Vgl. Loos, Scheer (1995), Loos (1995), Partridge (1994).

61 Als Beispiele sind hier die Arbeiten von Moser (1995), Oivo, Basili (1992) und Kaufmann (1994, S. 96-109) zu nennen.

62 Ein solchen Meta-Schätzverfahren ist beispielsweise die Object-Point-Methode, die in Oriolo (1990) und Strahringer (1991, S. 79-92) vorgestellt wird. Sie ist zwar als Meta-Schätzverfahren konzipiert, jedoch bislang ohne ausdrücklichen Bezug zur Metamodellierung. Unter Verwendung von Metamodellierung ließe sich das Verfahren wesentlich transparenter gestalten.

63 Zur Veranschaulichung der Breite dieses Spektrums soll hier abschließend eine repräsentative Auswahl von Arbeiten, die sich mit Metamodellierung beschäftigen, genannt werden. Neben den in diesem Kapitel bereits angeführten Arbeiten sind zusätzlich folgende zu nennen: Atzeni, Torlone (1991), Blaha (1992), Hesse (1991), Hesse, Bosman,

ausgedrückt und in den Vordergrund gestellt werden, die ansonsten nur implizit oder unpräzise formuliert würden. Erst das explizite Benennen und Beschreiben wichtiger Zusammenhänge erlaubt es, über diese systematische Aussagen zu machen. Metamodellierung kann hierzu einen wesentlichen Beitrag leisten.

## Abkürzungsverzeichnis

| | |
|---|---|
| ANSI | American National Standards Institute |
| BNF | Backus-Naur-Form |
| CAME | Computer Aided Methodology Engineering |
| CASE | Computer Aided Software Engineering |
| CC | Compiler-Compiler |
| CGGL | code generator generator language |
| ER(D/M) | Entity-Relationship(-Diagramm/-Modell) |
| IRDS | Information Resource Dictionary System |
| ISO | International Standardization Organization |
| $M^0M$, M | Modell |
| $M^1M$, MM | Metamodell |
| $M^2M$, MMM | Metametamodell |
| $M^nM$ | Modell n-ter Stufe |
| $M^0P$, P | Prozeß |
| $M^1P$, MP | Metaprozeß |
| $M^2P$, MMP | Metametaprozeß |
| $M^nP$ | Prozeß n-ter Stufe |
| $M^0S$, S, OS | Sprache, Objektsprache |
| $M^1S$, MS | Metasprache |
| $M^2S$, MMS | Metametasprache |
| $M^nS$ | Sprache n-ter Stufe |
| MSP | Maschinensprache |
| PN | Petri Netz |
| TMDL | target machine description language |

## Literaturverzeichnis


ALBERS (1995) Albers, S., Modellbasiertes Prototyping - Entwicklung betrieblicher Anwendungssysteme auf der Basis von Metamodellen, Dissertation TH Darmstadt 1995.

ALDERSON (1991) Alderson, A., Meta-CASE Technology, in: Endres, A., Weber, H., Hrsg., Software Development Environments and CASE Technology, LNCS 509, Berlin usw. 1991, S. 81-91.

AMBERG, RAUE (1995) Amberg, M., Raue, H., Ein Beschreibungsformalismus für Informationssystem-Architekturen, in: Informationssystem Architekturen 2 (1995) 1, S. 67-75.

ATZENI, TORLONE (1993) Atzeni, P., Torlone, R., A Metamodel Approach for the Mangement of Multiple Models and the Translation of Schemes, in: Information Systems 18 (1993) 6, S. 349-362.

BERTHEL (1970) Berthel, J., Modell, allgemein, in: Kosiol, E., Hrsg., Handwörterbuch des Rechnungswesens, Stuttgart 1970, Sp. 1122-1129.

BLAHA (1992) Blaha, M., Models of models, in: JOOP 5 (1992/93) 5, S. 13-17.

BOCHENSKI (1965) Bochenski, I.M., Die zeitgenössischen Denkmethoden, 3. Auflg., Bern und München 1965.

BRINKKEMPER ET AL. (1990) Brinkkemper, S., Lange, M. de, Looman, R., Steen, F.H.G.C. van der, On the Derivation of Method Companionship by Meta-Modelling, in: ACM SIGSOFT Software Engineering Notes 15 (1990) 1, S. 49-58.

BÜHLER (1934) Bühler, K., Sprachtheorie, Die Darstellungsfunktion der Sprache, Jena 1934.

CALLENDER (1986) Callender, E.D., A Meta-Model for the Software Processes, in: ACM SIGSOFT Software Engineering Notes 11 (1986) 4, S. 49.

CARMICHAEL (1994) Carmichael, A., Toward a Common Object-Oriented Meta-Model for Object Development, in: Carmichael, A., Hrsg., Object Development Methods, New York 1994, S. 321-331.

CARNAP (1960) Carnap, R., Symbolische Logik, 2. Auflg., Wien 1960.

COAD, YOURDON (1991) Coad, P., Yourdon, P., Object-Oriented Analysis, 2. Auflg., Englewood Cliffs 1991.

CURTIS, KELLNER, OVER (1992) Curtis, B., Kellner, M., Over, J., Process Modeling, in: CACM 35 (1992) 9, S. 75-90.

DANFORTH, FORMAN (1994) Danforth, S., Forman, I.R., Reflections on Metaclass Programming in SOM, in: OOPSLA `94, S. 440-452.

DOWSON (1987) Dowson, M., Iteration in the Software Process: Review of the 3rd International Software Workshop, in: Proceedings of the 9th International Conference on Software Engineering, Los Alamitos 1987, S. 36-39.

EMBLEY, WOODFIELD, KURTZ (1992) Embley, D., Kurtz, B.D., Woodfield, S.N., Object-Oriented Systems Analysis: A Model-Driven Approach, Englewood Cliffs 1992.


---

Damme (1988), Jarke (1992), Kokol (1993), Kratz (1994a), Kratz (1994b), Mark, Roussopoulos (1986), Mylopoulos et al. (1990), Odell (1992), Potts (1989), Rossi et al. (1992), Scheer (1990), Sommerville, Welland, Beer (1987), Wileden (1986).

| | |
|---|---|
| ESSLER (1972) | Essler, W.K., Die Sprache der Logik, in: Speck, J., Hrsg., Philosophie der Gegenwart I, Göttingen 1972, S. 300-339. |
| FÄRBERBÖCK, GUTZWILLER, HEYM (1991) | Färberböck, H., Gutzwiller, T., Heym, M., Ein Vergleich von Requirements Engineering Methoden auf Metamodell-Basis, in: Timm, M., Hrsg., Requirements Engineering ´91: "Structured Analysis" und verwandte Ansätze, IFB 273, Berlin usw. 1991. |
| FERBER (1988) | Ferber, J., Computational Reflection in Class based Object Oriented Languages, in: OOPSLA ´88, ACM SIGPLAN Notices 23 (1988), S. 317-326. |
| FERSTL, SINZ (1993) | Ferstl, O., Sinz, E., Grundlagen der Wirtschaftsinformatik, Band 1, München und Wien 1993. |
| FOOTE, JOHNSON (1988) | Foote, B., Johnson, R.B., Reflective Facilities in Smalltalk-80, in: OOPSLA ´88, ACM SIGPLAN Notices 23 (1988), S. 327-335. |
| FRANK (1994) | Frank, U., Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung, München und Wien 1994. |
| GOOR, HONG, BRINKKEMPER (1992) | Goor, G. van den, Hong, S., Brinkkemper, S., A Comparison of Six Object-Oriented Analysis and Design Methods, Method Engineering Institute, University of Twente, Enschede 1992. |
| GRIES (1971) | Gries, D., Compiler Construction for Digital Computers, New York usw. 1971. |
| GROCHLA (1969) | Grochla, E., Modelle als Instrument der Unternehmensführung, in: ZfbF 21 (1969), S. 382-397. |
| GUTZWILLER (1994) | Gutzwiller, T., Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen, Heidelberg 1994. |
| HABERBECK (1986) | Haberbeck, R., Sprache, menschliche, in: Schneider, H.-J., Hrsg., Lexikon der Informatik und Datenverarbeitung, 2. Auflg., München und Wien 1986, S. 552. |
| HABERMANN, LEYMANN (1993) | Habermann, H.-J., Leymann, F., Repository: eine Einführung, München und Wien 1993. |
| HARMSEN, BRINKKEMPER, OEI (1994) | Harmsen, F., Brinkkemper, S., Oei, H., Situational Method Engineering for Information System Project Approaches, in: Verrijn-Stuart, A.A., Olle, T.W., Hrsg., Methods and Associated Tools for the Information Systems Life Cycle, Amsterdam 1994. |
| HARS (1994) | Hars, A., Referenzdatenmodelle: Grundlagen effizienter Datenmodellierung, Wiesbaden 1994. |
| HESS, BRECHT (1995) | Hess, T., Brecht, L., State of the Art des Business Process Redesign: Darstellung und Vergleich bestehender Methoden, Wiesbaden 1995. |
| HESSE (1991) | Hesse, W., Two metamodels for application system development - conventional vs. object-oriented approach, in: Broy, M., Wirsing, M., Hrsg., Methods of Programming, Selected Papers on the CIP-Project, Berlin usw. 1991, S. 3-18. |
| HESSE, BOSMAN, DAMME (1988) | Hesse, W., Bosman, J.W., Damme, ten A.B.J., A Four-Level Metamodel for Application System Development, in: Bullinger, H.-J., Hrsg., EUROINFO ´88: Information Technology for Organisational Systems, Amsterdam 1988, S. 575-581. |
| HESSE ET AL. (1994) | Hesse, W., Barkow, G., von Braun, H., Kittlaus, H.-B., Scheschonk, G., Terminologie der Softwaretechnik: Ein Begriffsystem für die Analyse und Modellierung von Anwendungssystemen, Teil 1: Begriffsystematik und Grundbegriffe, in: Informatik-Spektrum 17 (1994), S. 39-47. |
| HEYM (1995) | Heym, M., Prozeß- und Methoden-Management für Informationssysteme: Überblick und Referenzmodell, Berlin usw. 1995. |
| HONG, GOOR, BRINKKEMPER (1993) | Hong, S., Goor, G. van den, Brinkkemper, S., A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies, in: Proceedings of the 26th Hawaii International Conference on Systems Sciences 1993, S. 689-698. |
| HOPGOOD (1970) | Hopgood, F.R.A., Compiler: Die Übersetzung von Programmiersprachen, München 1970. |
| ISCOE, WILLIAMS, ARANGO (1991) | Iscoe, N., Williams, G.B., Arango, G., Domain Modeling for Software Engineering, in: Proceedings of the 13th International Conference on Software Engineering, Los Alamitos 1991, S. 340-343. |
| JARKE (1992) | Jarke, M., Metamodellierung: Werkzeuge für das Engineering von Unternehmensprozessen, in: Hansmann, K.-W., Scheer, A.-W., Hrsg., Praxis und Theorie der Unternehmung: Produktion - Information - Planung, Wiesbaden 1992, S. 157-175. |
| KAUFMANN (1994) | Kaufmann, A.H., Software-Reengineering: Analyse, Restrukturierung und Reverse-Engineering von Anwendungssystemen, München und Wien 1994. |
| KLAUS (1968) | Klaus, G., Wörterbuch der Kybernetik, Berlin 1968. |
| KLEINKNECHT, WÜST (1976) | Kleinknecht, R., Wüst, E., Lehrbuch der elementaren Logik, Band 1: Aussagenlogik, München 1976. |
| KOKOL (1993) | Kokol, P., Metamodeling: How, Why and What ?, in: ACM SIGSOFT Software Engineering Notes 18 (1993) 2, S. 25-26. |
| KOSIOL (1961) | Kosiol, E., Modellanalyse als Grundlage unternehmerischer Entscheidungen, in: ZfbF 13 (1961), S. 318-334. |
| KOSIOL (1964) | Kosiol, E., Betriebswirtschaftslehre und Unternehmensforschung, in: ZfB 34 (1964) 12, S. 743-762. |
| KRATZ (1994a) | Kratz, G., Ein Ansatz zur einheitlichen Darstellung der Objekte einer beliebiger (!) Instantiierungsstufe und des Übergangs zur nächstniedrigeren Instantiierungsstufe, Teil I: Objekte, in: EMISA Forum (1994) 1, S. 43-59. |

KRATZ (1994b) Kratz, G., Ein Ansatz zur einheitlichen Darstellung der Objekte einer beliebiger (!) Instantiierungsstufe und des Übergangs zur nächstniedrigeren Instantiierungsstufe, Teil II: Übergangsformalismus, in: EMISA Forum (1994) 2, S. 63-70.

KROEBER-RIEL (1969) Kroeber-Riel, W., Wissenschaftstheoretische Sprachkritik in der Betriebswirtschaftslehre, Berlin 1969.

LEHNER (1995) Lehner, F., Grundfragen und Positionierung der Wirtschaftsinformatik, in: Lehner, F., Hildebrand, K., Maier, R., Wirtschaftsinformatik: theoretische Grundlagen, Wien 1995, S. 1-72.

LOOS (1995) Loos, P., Metainformationen - Generische Strukturen für Informationssysteme, in: EMISA Forum (1995) 2, S. 56 -58.

LOOS, SCHEER (1995) Loos, P., Scheer, A.-W., Vom Informationsmodell zum Anwendungssystem - Nutzenpotentiale für den effizienten Einsatz von Informationssystemen, in: König, W., Hrsg., Wirtschaftsinformatik ´95: Wettbewerbsfähigkeit, Wirtschaftlichkeit, Innovation, Heidelberg 1995, S. 185-201.

LORENZ (1980a) Lorenz, K., Metasprache, in: Mittelstraß, J., Hrsg., Enzyklopädie Philosophie und Wissenschaftstheorie, Bd. 1, Mannheim usw. 1980, S. 875.

LORENZ (1980b) Lorenz, K., Objektsprache, in: Mittelstraß, J., Hrsg., Enzyklopädie Philosophie und Wissenschaftstheorie, Bd. 1, Mannheim usw. 1980, S. 1054-1055.

MAES (1987) Maes, P., Concepts and Experiments in Computational Reflection, in: OOPSLA ´87, ACM SIGPLAN Notices 22 (1987) 12, S. 147-155.

MADHAVJI, SCHÄFER (1991) Madhavji, N.H., Schäfer, W., Prism - Methodology and Process-Oriented Environment, in: IEEE ToSE 17 (1991) 12, S. 1270-1283.

MARK, ROUSSOPOULOS (1986) Mark, L., Roussopoulos, N., Metadata Management, in: Computer 19 (1986) 12, S. 26-36.

MARTIN, ODELL (1992) Martin, J., Odell, J.J., Object-Oriented Analysis and Design, Englewood Cliffs 1992.

MERBETH (1991) Merbeth, G., On the Functional and Architectural Integration of CASE Systems, in: Endres, A., Weber, H., Hrsg., Software Development Environments and CASE Technology, LNCS 509, Berlin usw. 1991, S. 31-43.

MERCURIO ET AL. (1990) Mercurio, V.J., Meyers, B.F., Nisbet, A.M., Radin, G., AD/Cycle strategy and architecture, in: IBM Systems Journal 29 (1990) 2, S. 170-188.

MÖNKE (1986) Mönke, H., Sprache, in: Schneider, H.-J., Hrsg., Lexikon der Informatik und Datenverarbeitung, 2. Auflg., München und Wien 1986, S. 551.

MOSER (1995) Moser, S., Metamodels for Object-Oriented Systems: A Proposition of Metamodels Describing Object-Oriented Systems at Consecutive Levels of Abstraction, in: Software - Concepts and Tools 16 (1995), S. 63-80.

MYLOPOULOS ET AL. (1990) Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M., Telos: Representing Knowledge About Information Systems, in: ACM Transactions on Information Systems 8 (1990) 4, S. 325-362.

ODELL (1992) Odell, J.J., Object types as objects - and vice versa, in: JOOP 4 (1991/92) 9, S. 45-48.

OIVO, BASILI (1992) Oivo, M., Basili, V.R., Representing Software Engineering Models: The TAME Goal Oriented Approach, in: IEEE ToSE 18 (1992) 10, S. 886-898.

OLLE ET AL. (1991) Olle, T.W., Hagelstein, J., Macdonald, I.G., Rolland, C., Sol, H.G., Van Assche, F.J.M., Verrijn-Stuart, A.A., Information Systems Methodologies: A Framework for Understanding, 2. Auflg., Wokingham usw. 1991.

OPP (1976) Opp, K.-D., Methodologie der Sozialwissenschaften, Hamburg 1976.

OQUENDO, ZUCKER, GRIFFITHS (1992) Oquendo, F., Zucker, J.-D., Griffiths, P., A Meta-CASE-Environment for Software Process-centered CASE Environments, in: Loucopoulos, P., Hrsg., Advanced Information Systems Engineering, Third International Conference CAiSE ´92, LNCS 593, Berlin und Heidelberg 1992, S. 568-588.

ORIOLO (1990) Oriolo, M., Schätzverfahren in 4GL- und CASE-Umgebungen, in: Reuter, A., Hrsg., GI - 20. Jahrestagung - Informatik auf dem Weg zum Anwender, Berlin usw. 1990, IFB 258, S. 475-485.

OUDENAARDEN (1953) Oudenaarden, P.C., Zeichen und Struktur in der Naturwissenschaft, in: Studium Generale 6 (1953) 4, S. 228-235.

PARTRIDGE (1994) Partridge, C., Modelling the Real World, in: JOOP 7 (1994/95) 7, S. 39-45.

POCOCK (1991) Pocock, J.N., VSF and ist Relationship to Open Systems and Standard Repositories, in: Endres, A., Weber, H., Hrsg., Software Development Environments and CASE Technology, LNCS 509, Berlin usw. 1991, S. 53-68.

POHL, JARKE (1992) Pohl, K., Jarke, M., Quality Information Systems: Repository Support for Evolving Process Models, Aachener Informatik-Berichte Nr. 92-37, Aachen 1992.

POPPER (1993) Popper, K., Objektive Erkenntnis, Hamburg 1993.

POTTS (1989) Potts, C., A Generic Model for Representing Design Methods, in: Proceedings of the 11th International Conference on Software Engineering, Los Alamitos 1989, ICSE-11, S. 217-226.

RAASCH (1993) Raasch, J., Systementwicklung mit Strukturierten Methoden, 3. Auflg., München und Wien 1993.

ROSSI ET AL. (1992) Rossi, M., Gustafsson, M., Smolander, K., Johansson, L.-A., Lyytinen, K., Metamodeling Editor as a Front End Tool for a CASE Shell, in: Loucopoulos, P., Hrsg., Advanced Information Systems Engineering, Third International Conference CAiSE ´92, LNCS 593, Berlin und Heidelberg 1992, S. 546-567.

| | |
|---|---|
| RUFFNER (1972) | Ruffner, A., Wissenschaftstheoretische Überlegungen zur Betriebswirtschaftlichen Organisationslehre, in: Dlugos, G., Eberlein, G., Steinmann, H., Hrsg., Wissenschaftstheorie und Betriebswirtschaftslehre, Düsseldorf 1972, S. 185-207. |
| RUMBAUGH (1995) | Rumbaugh, J., What is a method ?, in: JOOP 8 (1995) 6, S. 10-26. |
| RUNGGALDIER (1990) | Runggaldier, E., Analytische Sprachphilosophie, Stuttgart usw. 1990. |
| SCHEER (1990) | Scheer, A.-W., Konzept für ein betriebswirtschaftliches Informationsmodell, in: ZfB 60 (1990) 10, S. 1015-1030. |
| SCHEER (1992) | Scheer, A.-W., 2. Auflg., Architektur integrierter Informationssysteme: Grundlagen der Unternehmensmodellierung, Berlin und Heidelberg 1992. |
| SCHREIBER (1969) | Schreiber, R., Erkenntniswert betriebswirtschaftlicher Theorien: Einführung in die Methodik der Betriebswirtschaftslehre, Wiesbaden 1960. |
| SMOLANDER ET AL. (1991) | Smolander, K., Lyytinen, K., Tahvanainen, V.-P., Marttiin, P., MetaEdit - A Flexible Graphical Environment for Methodology Modelling, in: Andersen, R., Bubenko, J.A., Solvberg, A., Hrsg., Advanced Information Systems Engineering, Third International Conference CAiSE ´91, LNCS 498, Berlin und Heidelberg 1991, S. 168-193. |
| SOMMERVILLE, WELLAND, BEER (1987) | Sommerville, I., Welland, R., Beer, S., Describing Software Design Methodologies, in: The Computer Journal 30 (1987) 2, S. 128-133. |
| SPINNER (1969) | Spinner, H., Modelle und Experimente, in: Grochla, E., Hrsg., Handwörterbuch der Organisation, Stuttgart 1969, Sp. 1000-1010. |
| STACHOWIAK (1973) | Stachowiak, H., Allgemeine Modelltheorie, Wien und New York 1974. |
| STARKE (1993) | Starke, G., Urgent Research Issues in Software Process Engineering, in. ACM SIGSOFT Software Engineering Notes 18 (1993) 4, S. 13-15. |
| STEELE, ZASLAVSKY (1994) | Steele, P.M., Zaslavsky, A.B., The Role of Meta Models in Federating System Modelling Techniques, in: Elmasri, R.A., Kouramajian, V., Thalheim, B., Hrsg., Entity-Relationship - ER´93, 12th International Conference on the Entity-Relationship Approach, LNCS 823, Berlin usw. 1994, S. 315-326. |
| STEGMÜLLER (1974) | Stegmüller, W., Wissenschaftliche Erklärung und Begründung, Berlin usw. 1974. |
| STRAHRINGER (1991) | Strahringer, S., Aufwandsschätzung von DV-Projekten in 4GL-Umgebungen, Studienarbeit TH Darmstadt 1991. |
| STÜLPNAGEL (1991) | Stülpnagel, A. von, Repositories - Konzepte, Architekturen, Standards, in: HMD 161/1991, S. 10-25. |
| SUPPES (1961) | Suppes, P., A Comparison of the Meaning and Uses of Models in Mathematics and the Empirical Sciences, in: The Concept and the Role of the Model in Mathematics and Natural Sciences, Proceedings of the Colloquium sponsored by the Division of Philosophy of Sciences of the International Union of History and Philosophy of Sciences organized at Utrecht, January, 1960, Dordrecht 1961, S. 163-177. |
| TARDIEU, FRANCKSON (1987) | Tardieu, H., Franckson, M., Contribution of the Entity-Relationship Approach to Object Management in an Information System Design Workbench, in: Spaccapietra, S., Hrsg., Entity-Relationship Approach, Amsterdam 1987. |
| TREMBLAY, SORENSON (1985) | Tremblay, J.-P., Sorenson, P. G., The Theory and Practice of Compiler Writing, New York usw. 1985. |
| VERHOEF, HOFSTEDE, WIJERS (1991) | Verhoef, T.F., Hofstede, A.H.M. ter, Wijers, G.M., Structuring modelling knowledge for CASE shells, in: Andersen, R., Bubenko, J.A., Solvberg, A., Hrsg., Advanced Information Systems Engineering, Third International Conference CAiSE ´91, LNCS 498, Berlin und Heidelberg 1991, S. 502-524. |
| WILD (1966) | Wild, J., Grundlagen und Probleme der betriebswirtschaftlichen Organisationslehre, Berlin 1966. |
| WILEDEN (1986) | Wileden, J.C., This is *IT*: A Meta-Model of the Software Process, in: ACM SIGSOFT Software Engineering Notes 11 (1986) 4, S. 9-11. |
| WINKLER (1989) | Winkler, J., The Entity-Relationship-Approach and the Information Resource Dictionary System Standard, in: Batini, C., Hrsg., Entity-Relationship Approach, Amsterdam 1989. |
| WOLTERS (1980) | Wolters, G., Modell, in: Mittelstraß, J., Hrsg., Enzyklopädie Philosophie und Wissenschaftstheorie, Bd. 1, Mannheim usw. 1980, S. 911-913. |
| ZEMANEK (1966) | Zemank, H., Semiotics and Programming Languages, in: CACM 9 (1966) 3, S. 139-143. |
| ZIMA (1989) | Zima, H., Compilerbau I: Analyse, 2., korr. Auflg., Mannheim, Wien und Zürich 1989. |