erschienen in: Proceedings of the International Knowledge Technology Forum (KnowTechForum) '99, 16.-18. September 1999, Potsdam.

Probleme und Gefahren im Umgang mit "Meta"-Begriffen: ein Plädoyer für eine sorgfältige Begriffsbildung

Susanne Strahringer, TU Darmstadt, Wirtschaftsinformatik II, e-mail: susanne@bwl.tu-darmstadt.de

1 Einleitung

In verschiedenen wissenschaftlichen Disziplinen ist mit Erreichen eines gewissen Reifegrades der gewonnenen Erkenntnisse zu beobachten, daß zunehmend versucht wird, diese Erkenntnisse zu verallgemeinern, sie vom eigentlichen Betrachtungsgegenstand zu lösen und sie auf eine höhere Abstraktionsstufe zu übertragen. Typischerweise werden sie somit selbst zum Gegenstand der Betrachtung und dabei sogar auf sich selbst angewandt. Meist wird dieser Wechsel der Bezugsebene, also der Übergang von einem Betrachtungsgegenstand auf einen abstrakteren – hinsichtlich eines bestimmten Prinzips "darüberliegenden" – Gegenstand über den Präfix "Meta" gekennzeichnet. Solche "Meta"-Begriffe sind im Umfeld der Wirtschaftsinformatik und in den benachbarten Disziplinen in vielen Bereiche anzutreffen. Man spricht z.B. von Metadaten, Metawissen, Metasprachen, Metamodellen, Metaprozessen und Metastrategien, in instrumentellen Gebieten auch von Meta-Suchmaschinen, Meta-Compilern oder Meta-(CASE)-Tools, um nur einige Beispiele zu nennen.

In manchen Anwendungsbereichen läßt sich diese Ebenenbildung sogar über mehrere Stufen hinweg fortführen, man kommt von Meta- zu Meta-Meta-Begriffen usw. Probleme bei der Bildung solcher Metaebenen ergeben sich, wenn nicht deutlich wird, auf welchem Kriterium die Ebenenbildung basiert. Konzeptionelle Mißverständnisse und Widersprüche drohen insbesondere in solchen Fällen, in denen die Ebenenbildung über mehrere Ebenen fortgeführt wird und das Prinzip der Metaebenenbildung dabei variiert. Ein weiterer Problembereich ist darin zu sehen, daß Meta-Begriffe sehr häufig zu weit gefaßt werden, dadurch an Prägnanz und Abgrenzungsschärfe verlieren und zunehmend verwässern.

Im folgenden wird eine Auswahl an Problemkreisen aus dem Umfeld von Meta-Begriffen behandelt. Dabei wird jeder der Problemkreise durch ein Beispiel aus der (Wirtschafts-)Informatik veranschaulicht, die sich in diesem Kontext ergebenden Gefahren werden genannt. Die Beispiele stammen aus dem Umfeld von Metasprachen, Metamodellen und Metawerkzeugen. Im Hinblick auf diese Gebiete erweist es sich als hilfreich, zunächst den Begriff der Metasprache völlig losgelöst von der Informatik aus der Perspektive der Logik bzw. Sprachphilosophie zu betrachten. Der aus der Sprachstufentheorie stammende Begriff der Metasprache und die auf Grundlage von Metasprachen bildbare Hierarchie von Sprachebenen kann unmittelbar auf die Informatik, insbesondere die Modellierung, übertragen werden. Deshalb folgt zunächst in einem Grundlagenkapitel eine kurze Einführung in die Sprachstufentheorie, bevor im darauffolgenden Abschnitt die einzelnen Problemkreise dargestellt werden.

2 Sprachstufentheorie als Grundlage verschiedener Meta-Begriffe

Aussagen über einen zu untersuchenden Gegenstandsbereich werden üblicherweise in Sprache formuliert. Wird Sprache selbst zum Gegenstand einer 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.[1]

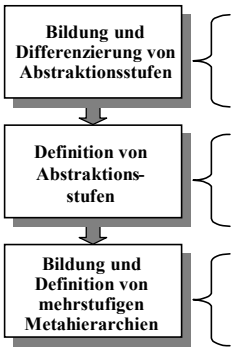
Das Prinzip des Bildens einer Metasprache zu einer Objektsprache ist rekursiv anwendbar, denn auch die Metasprache einer Objektsprache kann wiederum zum Gegenstand der Untersuchung werden. Bei der für diese Untersuchung verwendeten Sprache handelt es sich dann um die Metasprache einer Metasprache oder um die Metametasprache bezüglich der ursprünglichen Objektsprache. Dieses Prinzip läßt sich fortsetzen, so daß eine ganze Hierarchie von Sprachebenen gebildet werden kann. Um dies terminologisch handhaben zu können, spricht man allgemein von Metasprachen i-ter Stufe, in Kurzform auch mit "M^iS" 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.

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. Die Attribute "Meta-" bzw. "Objekt-" kennzeichnen also Rollen, die Sprachen in einem bestimmten Kontext einnehmen können. Verschiedene Rollen können auch von ein und derselben Sprache eingenommen werden. In einem solchen Fall ist es zweckmäßig, Konventionen zur Kennzeichnung der einzelnen Sprachebenen zu verwenden. In der geschriebenen Alltagssprache unterscheidet man daher zwischen *Gebrauch* und *Erwähnung* (Anführung) von

Wörtern und verwendet zur Kennzeichnung Anführungszeichen im Falle der Erwähnung. Die Metasprache ist also diejenige Sprache, die man gebraucht, um Aussagen über eine Objektsprache zu machen.

3 Problemkreise

Bereits bei der Darstellung der Sprachstufentheorie wurden einige der im folgenden zu diskutierenden Problemkreise angedeutet. Diese Problemkreise und die sich daraus ergebenden Gefahren werden anhand eines Beispieles aus dem Umfeld von Metasprachen, -modellen oder -werkzeugen veranschaulicht. Aus den beispielhaften Ausführungen wird jeweils abschließend ein verallgemeinertes Fazit mit Blick auf die Vermeidung solcher Gefahren gezogen. Tab. 1 gibt eine Übersicht über die behandelten Themen.



	Problemkreise	Beispiel
1	Mangelndes Bewußtsein für die Existenz verschiedener Abstraktionsstufen	Meta-sprachen
2	Verwechslung von "harten" und "weichen" Abstraktionsstufen	Meta-modelle
3	Verkennung der kontextabhängigen Rolle von Meta-Gegenständen	Meta-sprachen
4	Mangelnde Abgrenzung und Fundierung von Meta-Begriffen	Meta-modelle
5	Inkonsistente Fortführung von Meta-Begriffen über mehrere Abstraktionsstufen	Meta-modelle
6	Unklare Verankerung von Meta-Begriffen	Meta-werkzeuge

Tabelle 1: Behandelte Problemkreise und Beispiele

3.1 Problemkreis 1: Mangelndes Bewußtsein für die Existenz verschiedener Abstraktionsstufen

In diesem ersten Themenbereich wird die grundlegende Frage der Existenz von Metaebenen behandelt und dargestellt, daß es im Falle des Vorhandenseins verschiedener Abstraktionsstufen zweckmäßig ist, dies explizit hervorzuheben. An dem im folgenden zu behandelnden Beispiel aus dem Umfeld von Metasprachen wird folgende ***Gefahr*** veranschaulicht: *Ein fehlendes Bewußtsein für die Existenz und Abgrenzung verschiedener Metaebenen kann zu einer völligen Fehleinschätzung von Potentialen und Risiken einer Technologie führen.*

Beispiel: XML und verwandte Sprachen

Dieser Problemkreis ist in einem sehr aktuellen Themenbereich, nämlich im Umfeld von HTML, SGML und XML anzutreffen.[2] XML wird häufig in einem Atemzug mit HTML genannt, ohne daß der entscheidende Unterschied zwischen diesen zwei Sprachen hervorgehoben wird: XML (e**x**tensible **m**arkup **l**anguage) stellt ebenso wie ihre Vorläuferin SGML (**s**tandard **g**eneralized **m**arkup **l**anguage) eine Metasprache zur Definition von Auszeichnungssprachen dar – ein Beispiel für eine solche Auszeichnungssprache ist HTML (**h**yper**t**ext **m**arkup **l**anguage). Abb. 1 zeigt die korrekte Zuordnung dieser Sprachen hinsichtlich ihrer Einstufung als Meta- bzw. konkrete Auszeichnungssprache.

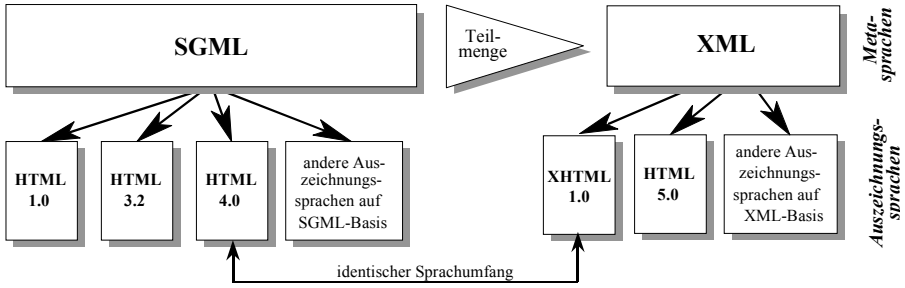


Abb. 1: SGML vs. XML vs. HTML

Etwas irreführend ist in diesem Zusammenhang die Benennung der Metasprachen als Auszeichnungssprachen: "ML" steht jeweils für "markup language". Im Falle von SGML weist der Zusatz "generalized" auf den metasprachlichen Charakter hin, im Falle von XML der Zusatz "extensible". Gerade letzteres verstärkt aber eher die Verwirrung. Die Metasprache XML selbst ist nicht erweiterbar, lediglich die in ihr definierten Sprachen. Nutzer solcher Sprachen können diesen Vorteil allerdings nur dann ausschöpfen, wenn sie ihre Systeme auf die Ebene der Metasprache umstellen und somit Sprachdefinitionen einlesen und interpretieren können. Ist dies gegeben, dann tritt die Flexibilität hinsichtlich der unterstützen Sprachen ein: sie können erweitert werden, oder es können gänzlich neu definierte Sprachen hinzukommen. Das in dieser Flexibilität liegende Potential ist im Zusammenhang mit dem WWW nicht darin zu sehen, daß SGML durch eine einfachere Metasprache wie XML abgelöst werden soll, sondern im Umstieg des Webs von der Ebene einer eingeschränkten Menge festgeschriebener Auszeichnungssprachen – die in Form von SGML-DTDs[3] definierten HTML-Versionen – auf die Metaebene. Erst dadurch, daß in Web-Browsern die unterstützten HTML-Elemente und Formen der Verarbeitung, denen sie zum Darstellungszeitpunkt zu unterziehen sind, nicht "hart-codiert" und festgeschrieben sind, sondern diese Definitionen in den Browser eingelesen werden, wird Erweiterbarkeit gewährleistet. Das Problem einer sich in immer kürzeren Abständen als notwendig erweisenden Überarbeitung des Sprachstandards HTML mit einem fortwährend anwachsenden Sprachumfang, was letztlich in der Version 4.0 zu einer Spezifikation im Umfang von mehr als 350 Seiten geführt hat, ist nicht durch Ablösung von SGML durch eine einfachere Variante dieser Sprache in

Form von XML zu beheben. Die entscheidende Innovation besteht im Umstieg von der Ebene der Auszeichnungssprache auf die Ebene der zugehörigen Metasprache. Genau in diesem Bereich ist folglich das Potential der XML-Technologien angesiedelt. Gleichzeitig läßt sich hieraus aber auch die zunehmende Komplexität zukünftiger Browsergenerationen und anderer Webtechnologien ableiten.

Fazit: Existieren Abstraktionsstufen, so sollten diese explizit zum Ausdruck gebracht werden, z.B. durch die Verwendung von Meta-Begriffen.

3.2 Problemkreis 2: Verwechslung von "harten" und "weichen" Abstraktionsstufen

In einigen Bereichen können sogar verschiedene Formen von Abstraktionsstufen auftreten, die sich grundlegend in der "Schärfe" der Ebenenbildung voneinander unterscheiden. Hier scheint es hilfreich, diese Formen deutlich voneinander abzugrenzen. Gerade im Bereich der Modellierung droht z.B. folgende ***Gefahr*****:** *Bei der Entwicklung sog. generischer Modelle (z.B. flexible Datenmodelle, Referenzmodelle, Muster) werden häufig Abstraktionsstufen eingesetzt, die eine ganz andere Qualität haben als Abstraktionsstufen in einer Metamodellhierarchie. Wird zwischen diesen Formen nicht klar differenziert, kann es zu Verwechslungen kommen. Die Abgrenzung zwischen generischen Modellen verschiedener Varianten und Metamodellen wird schwierig.*

Beispiel: Metamodelle vs. generische Modelle

Von Softwaresystemen wird zunehmend eine möglichst komfortable Anpaßbarkeit an sich ändernde fachliche Anforderungen erwartet. Dies kann z.B. dadurch erreicht werden, daß man versucht, Softwaresysteme in Teilbereichen generisch zu gestalten. Betrachtet man exemplarisch die Datenmodellierung, so kann ein Datenmodell z.B. durch "Metaisierung" flexibilisiert werden. Metaisierung bedeutet, daß Datenmodelle so gestaltet werden, daß sie die Hinzunahme neuer – eigentlich auf Typebene – angesiedelter Objekte zulassen. Der entsprechende modellierungstechnische Ansatz besteht darin, daß diejenigen Bereiche des Datenmodells, in denen die denkbaren Objekttypen nicht eingeschränkt werden sollen, flexibel gestaltet werden müssen, d.h. daß Objekttypen zu modellieren sind, deren Ausprägungen die ursprünglichen Objekttypen repräsentieren. Metaisierung bedeutet in diesem Kontext also die Verdrängung von Objekttypen auf die Ausprägungsebene durch Einführung problemadäquater Metaobjekttypen.[4] Das ursprünglich angestrebte Datenmodell wird folglich auf zwei Ebenen verteilt und ist vollständig beschrieben durch das metaisierte Datenmodell und den zu den Metaobjekttypen gehörenden Instanzen.[5] Das Beispiel aus Abb. 2 veranschaulicht diesen Zusammenhang. Fowler bezeichnet diese zwei Bereiche eines Modells, die im Beispiel durch eine graue Linie getrennt sind, als Wissensebene und operationelle Ebene und lehnt den Begriff Metamodellierung in diesem Zusammenhang ab.[6]

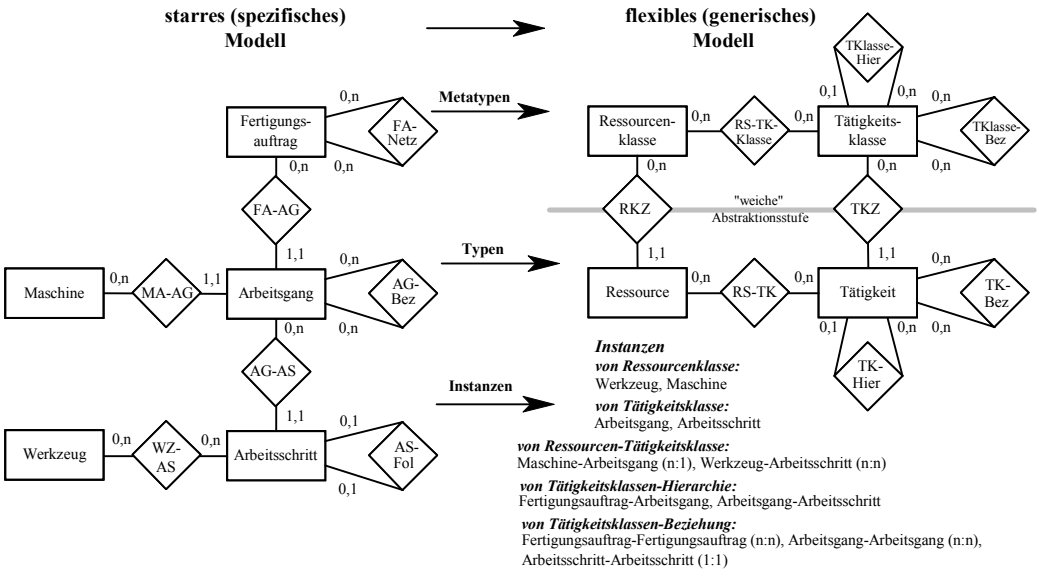


Abb. 2: Beispiele für "metaisierte" Datenmodelle (vgl. Loos (1996, S. 168))

Unabhängig davon, ob man in diesem Zusammenhang von Metamodellierung spricht oder nicht, ist entscheidend, daß zwei verschiedene Formen der Abstraktionsstufenbildung vorliegen, die als "harte" vs. "weiche" Abstraktionsstufen bezeichnet werden. Charakteristikum der weichen Abstraktionsstufenbildung ist, daß Beziehungen, die zwischen Elementen verschiedener "weicher" Ebenen definiert werden können, qualitativ übereinstimmen mit Beziehungen zwischen Elementen innerhalb ein und derselben Ebene. Im Beispiel handelt es sich bei den Beziehungstypen "RKZ" und "TKZ", die über die "weiche" Ebenengrenze hinweggehen, um herkömmliche Beziehungstypen der Entity-Relationship-Modellierung, nicht etwa um Instantiierungsbeziehungen, die man über "harte" Ebenengrenzen hinweg antreffen würde. Diese "weichen" Ebenen lassen sich also nicht durch einen qualitativen Unterschied zwischen Inter- und Intra-Ebenen-Beziehungen abgrenzen.[7] Generische Modelle können auch intuitiv in der Entwicklungsarbeit entstehen, ohne daß explizit auf sie hingearbeitet wird.

Jeder Objekttyp mit einem Suffix der Form "-klasse", "-typ", "-gruppe" oder "-art" trägt potentiell zu einer solchen Flexibilisierung bei. Die Übergänge zwischen spezifischen und generischen Modellen sind also fließend, so daß die Frage nach der Existenz einer "weichen" Abstraktionsstufe im Gegensatz zu einer "harten" nicht zweifelsfrei geklärt werden kann.

Fazit: Arbeitet man mit gänzlich verschiedenen Formen von Abstraktionsstufen, so sollten diese explizit differenziert werden.

3.3 Problemkreis 3: Verkennung der kontextabhängigen Rolle von Meta-Gegenständen

Ist man sich der Existenz von Abstraktionsstufen bewußt und hebt diese explizit z.B. durch Einsatz von Meta-Begriffen hervor, so kann es trotzdem zur Verwechslung der Abstraktionsstufen kommen. Wie schon in den Grundlagen zur Sprachstufentheorie dargestellt, handelt es sich bei der "Meta"-Eigenschaft eines Gegenstandes nicht um eine absolute Charakterisierung, sondern um eine Rollenzuweisung. So ist es möglich, daß ein und derselbe Gegenstand in einem bestimmten Kontext der Metaebene zugeordnet wird, nicht jedoch in einem anderen, d.h., daß er unter Meta-Begriffe verschiedener Stufen fallen kann. Zur Veranschaulichung läßt sich hier das XML-Beispiel fortführen. Vernachlässigt man diese Problematik, so droht z.B. im Umfeld von Metasprachen folgende ***Gefahr:*** *Kann ein und dieselbe Sprache auf verschiedenen Metastufen auftreten, so besteht ein hohes Verwechslungspotential hinsichtlich der in einem gegebenen Kontext eingenommenen Rolle.*

Beispiel: Wohlgeformtes XML

Wie betont, handelt es sich bei XML um eine Metasprache, mittels der zunächst konkrete Auszeichnungssprachen auf Grundlage von sog. DTDs definiert werden müssen, bevor ein Dokument mit den entsprechenden Markierungen versehen werden kann. Ein solches Dokument wird auch allgemein als XML-Dokument bezeichnet und ist Instanz einer XML-DTD, die die zugehörige Sprachdefinition enthält. Um das WWW hinsichtlich der Portabilität seiner Web-Seiten nicht zu stark einzuschränken, wurde jedoch folgender Kompromiß eingegangen: XML-Dokumente können auch losgelöst von einer DTD verarbeitet werden. Man spricht dann von sog. wohlgeformten (well-formed) XML-Dokumenten, die sich bereits von einem Browser verarbeiten lassen, im Gegensatz zu gültigen (valid), die Instanz einer DTD sind und gegen diese geprüft werden. Wohlgeformten Dokumenten wird quasi eine implizite Sprachdefinition unterstellt, da eine explizite in Form einer DTD nicht zur Verfügung steht. Der Verzicht auf eine explizite Sprachdefinition führt zwar nur zu einer abgeschwächten syntaktischen Prüfung, erlaubt aber dennoch die Prüfung der Einhaltung eines gewissen von XML vorgegebenen Regelwerks. Diese Eigenschaft von XML, neben der metasprachlichen Funktion auch unmittelbar der Auszeichnung zu dienen, unterscheidet es von SGML. Die im Vergleich zu XML wesentlich größeren Freiheitsgrade von SGML, die z.B. bei entsprechenden metasprachlichen Festlegungen den Verzicht auf öffnende oder schließende Tags erlauben oder eine besondere Syntax für leere Tags unterstützen, sorgen dafür, daß ein SGML-Parser ein SGML-Dokument immer nur bei Vorliegen einer zugehörigen DTD verarbeiten kann.

Insgesamt muß also zwischen drei Sprachebenen unterschieden werden: die Metasprache XML, die mittels XML definierte Markup-Sprache sowie die Sprache, in der der Dokumentinhalt formuliert wird. Um innerhalb eines Dokuments zwischen Markup und eigentlichem Text differenzieren zu können und um XML in seiner metasprachlichen Verwendung abgrenzen zu können, gibt es syntaktische Konventionen, die in Tabelle 2 dargestellt sind.

Beginn	Ende	Markup-Typ	Sprachebene
<	>	start-tag: kennzeichnet Beginn einer Markup-Anweisung	Anweisungen der in XML definierten Markup-Sprache
</	>	end-tag: kennzeichnet Ende einer Markup-Anweisung	
&	;	entity reference: begrenzt Referenz auf ein Entity	
<!	>	markup declaration: definiert Anweisungen der Markup-Sprache	Anweisungen der Metasprache XML

Tabelle 2: Kennzeichnung von Markup-Anweisungen

Die dargestellte Eigenschaft von XML verdeutlicht, daß ein und dieselbe Sprache in verschiedenen Rollen auftreten kann. In solchen Situationen ist es besonders wichtig, die Existenz dieser verschiedenen Rollen hervorzuheben (Problemkreis 1) sowie Mechanismen zu finden, die die in einem bestimmten Kontext eingenommene Rolle erkennen lassen.

Fazit: Ist die Meta-Eigenschaft keine kontextunabhängige Eigenschaft des betrachteten Gegenstandes, sondern fungiert sie als Rollenzuweisung, so muß die jeweils eingenommene Rolle zweifelsfrei erkennbar sein. Hierzu können z.B. syntaktische Konventionen definiert werden.

3.4 Problemkreis 4: Mangelnde Abgrenzung und Fundierung von Meta-Begriffen

Das Problem einer mangelnden begrifflichen Abgrenzung und Fundierung stellt einen grundlegenden Problembereich von Meta-Begriffen dar. Das Abgrenzungsproblem tritt zudem dann verstärkt auf, wenn der zugrundeliegende Basis-Begriff, im zur Veranschaulichung betrachteten Beispiel der Modellbegriff, sehr breit ist, unklar definiert wurde oder in einer anderen Hinsicht terminologische Unklarheiten in sich birgt. Trotzdem läßt sich in den meisten Fällen eine angemessene begriffliche Fundierung finden, die zu einer klaren Begriffsbildung führt. Möglicherweise kann auch in mehreren Schritten vorgegangen werden, z.B. durch eine anfänglich enge Begriffsbildung, die dann durch Variantenbildung oder Verallgemeinerung etwas weiter gefaßt werden kann. Dies wird am Beispiel des Metamodellbegriffs im folgenden dargestellt. Achtet man nicht auf eine klare Begriffsbildung, ist mit folgender ***Gefahr*** zu rechnen: *Ist der zugrundeliegende Basis-Begriff (hier z.B. Modell) nicht klar definiert, verstärkt sich die Unklarheit in der terminologischen Abgrenzung beim Übergang auf den zugehörigen Meta-Begriff (hier Metamodell). Meta-Begriffe werden meist zu weit gefaßt und können dadurch an Aussagekraft verlieren.*

Beispiel: Begriff des Metamodells

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. Trotz seiner Bedeutung ist er selbst selten Gegenstand einer ausführlichen Betrachtung. Eine Untersuchung von in der Literatur anzutreffenden Metamodellbegriffen zeigt, daß unter Metamodellen ein breites und heterogenes Spektrum an Modellen zu verstehen ist.[8] Viele Arbeiten basieren zudem auf einem eher intuitiven Begriffsverständnis. Es fehlt daher ein Erklärungsversuch, der den Begriff nicht zu weit faßt, aber trotzdem so allgemein ist, daß verschiedene Fälle abgedeckt sind. In einem ersten Ansatz werden Metamodelle meist als Modelle anderer Modelle definiert. 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 in einem zweiten Schritt verallgemeinert, ohne dabei die Eigenschaft der Operationalität zu verlieren.

Der Begriff des Metamodells läßt sich beispielsweise aus der am Anfang dargestellten Sprachstufentheorie herleiten. Ziel der Modellbildung im Rahmen der Softwareentwicklung ist es, einen Gegenstandsbereich in einem Beschreibungsmodell abzubilden. Zur Formulierung dieses Modells wird eine Sprache verwendet, die der Sprachstufentheorie folgend als Objektsprache zu bezeichnen ist. Entsprechend ließe sich das Modell der untersten Ebene auch als Objektmodell bezeichnen. Im folgenden soll jedoch vereinfachend und wegen der zweideutigen Verwendung des Begriffs Objektmodell[9] 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 Beschreibungsmodell derjenigen Sprache definiert werden, in der das Modell der (i-1)-ten Stufe formuliert ist. Abb. 3 veranschaulicht den dargestellten Zusammenhang der identifizierten Modellierungsebenen bei Ableitung des Metamodellbegriffs aus der Stufentheorie der Sprachen.[10]

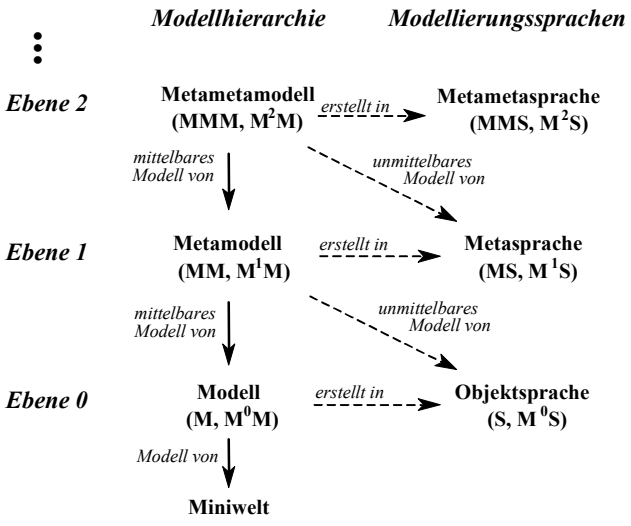


Abb. 3: Sprachbasierter Metamodellbegriff

Metamodelle auf Grundlage des 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. Auch Metamodelle selbst sind nicht per se einer der Modellierungsebenen zuzuordnen, sondern nehmen ihre Rolle als Meta- oder Metametamodell usw. in Relation zu anderen Modellen wahr. Die im vorherigen Abschnitt dargestellte Problematik ist also auch im Umfeld der Metamodellierung anzutreffen und verschärft sich, wenn die Stufenbildung über mehrere Ebenen fortgeführt wird.

Auf den in diesem Abschnitt behandelten Problemkreis der Abgrenzung von Meta-Begriffen zurückkommend bleibt festzuhalten, daß im betrachteten Beispiel des Metamodellbegriffs eine Fundierung durch die Sprachstufentheorie vorgenommen und eine Abgrenzung gewählt wurde, die die meisten Metamodellvarianten, die in der Wirtschaftsinformatik anzutreffen sind, abdeckt, ohne auf eine zu weite Begriffsbildung zurückgreifen zu müssen. Trotzdem ist eine Verallgemeinerung notwendig. Denn neben den berücksichtigten Formen ist noch mindestens eine andere Variante zu finden, die von praktischer Relevanz ist. Sie entstammt dem Themengebiet der Softwareprozeßmodellierung: Dort steht oftmals der Prozeß der Modellbildung im Vordergrund und nicht wie zuvor die Modellierungssprache.[11] Auf der Metamodellebene wird dann entsprechend der Prozeß der Modellbildung modellhaft abgebildet. Auf der darüberliegenden Ebene wird, das Prinzip der Metaebenenbildung fortsetzend, der Prozeß der Prozeßbildung modelliert.[12] Abb. 4 zeigt eine Abb. 3 entsprechende Darstellung dieses Metamodellbegriffs. Der wesentliche Unterschied in den beiden Ansätzen ist in der Verwendung verschiedener Prinzipien, die der Ebenenbildung zugrunde liegen, zu sehen. Dieses als Metaisierungsprinzip bezeichnete Kriterium beschreibt denjenigen Aspekt eines Modells, der in der übergeordneten Modellierungsstufe abgebildet wird.

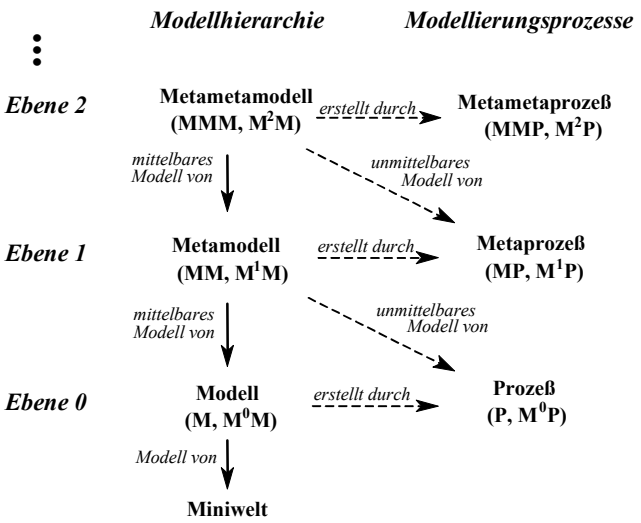


Abb. 4: Prozeßbasierter Metamodellbegriff

Das dargestellte Vorgehen zeigt, daß der Metamodellbegriff nach einem einheitlichen Konzept in verschiedenen Varianten eingegrenzt wurde. Auf Grundlage desselben Konzeptes können prinzipiell weitere Varianten formuliert werden, die einem einheitlichen Begriffsbildungsschema folgen.

Fazit: Meta-Begriffe sollten immer im Hinblick auf den zugrundeliegenden Basis-Begriff und das Kriterium, das der Stufenbildung beim Übergang auf die Metaebene zugrundeliegt, näher untersucht und abgegrenzt werden.

3.5 Problemkreis 5: Inkonsistente Fortführung von Meta-Begriffen über mehrere Abstraktionsstufen

Im zuvor betrachteten Abschnitt wurde gezeigt, daß sich durchaus verschiedene Kriterien identifizieren lassen, nach denen einen Abstraktionsstufenbildung vorgenommen werden kann. Bildet man nun Metaebenen über mehrere Stufen hinweg, dann kann in einer solchen Metahierarchie dieses Kriterium prinzipiell auch variiert werden. Dabei droht jedoch folgende ***Gefahr***, die wiederum am Beispiel der Metamodellierung veranschaulicht wird: *Mischt man in Modellhierarchien, die mehrere Ebenen umfassen, das Kriterium bzw. Prinzip, nach dem die Stufenbildung erfolgte, kann es zu konzeptionellen Widersprüchen kommen.*

Beispiel: Mischung von Metaisierungsprinzipien in Metamodellhierarchien

Grundsätzlich ist es möglich, beide der zuvor betrachteten Prinzipien der Abstraktionsstufenbildung (Sprache vs. Prozeß) in einer Modellhierarchie zu kombinieren.[13] Die Offenlegung des Metaisierungsprinzips ist dann allerdings sehr hilfreich und vermeidet konzeptionelle Widersprüche bei der Ebenenbildung. 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 Modellierungsbeispiel erläutert werden.

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. 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,

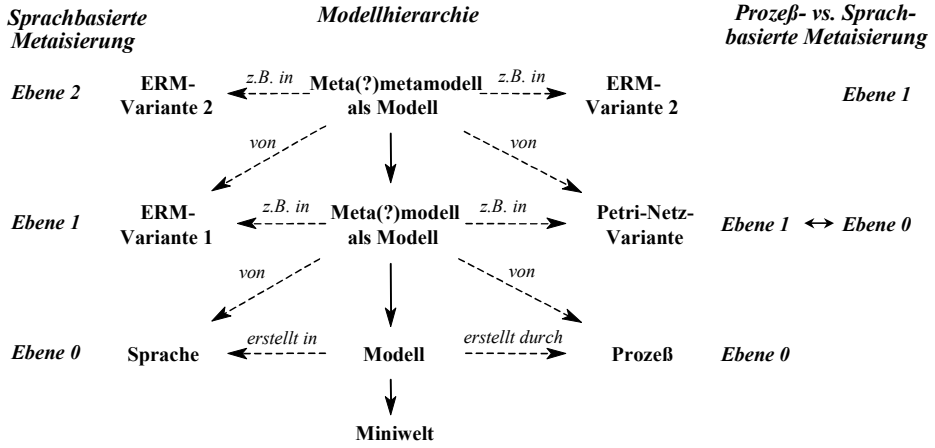


Abb. 5: Widerspruch bei gleichzeitiger Verwendung verschiedener Metaisierungsprinzipien

z.B. unter Verwendung einer weiteren ERM-Variante, beschrieben werden. Abb. 5 veranschaulicht den dargestellten Zusammenhang.

In Abb. 6 ist eine stark vereinfachte Modellhierarchie für ein Datenflußdiagramm dargestellt. Es handelt sich um ein konkretes Beispiel der in Abb. 5 schematisch abgebildeten Struktur. Im Gegensatz zu Abb. 5 wurde im konkreten Beispiel für die Prozeßmodellierung eine einfachere Modellierungssprache als ein Petri-Netz-Ansatz gewählt. Abgebildet ist auf Ebene 0 ein Datenflußmodell eines außersprachlichen Gegenstandsbereichs.[14] 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 ER-Diagramm dargestellt.

Das Bilden von Modellhierarchien in der Form, wie es in den beiden 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. 5 und 6 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 die mittlere Modellebene, also 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 und nicht um ein Metamodell. Wäre es ein Metamodell, müßte sich nach dem gewählten Metaisierungsprinzip ein zugehöriges Modell finden lassen. Das Modell auf der mittleren Ebene ist also in Relation zur darunterliegenden Ebene ein Metamodell, im Verhältnis zur übergeordneten Ebene aber ein Modell. Die Ebenennumerierung auf der rechten Seite von Abb. 5 verdeutlicht die korrekte Ebenenbildung. Das aus prozeßbasierter Sicht als Metamodell einzustufende 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 darüber, ob es sich bei einem Modell um ein Metamodell handelt, davon abhängt, von welchem Metaisierungsprinzip man ausgeht.

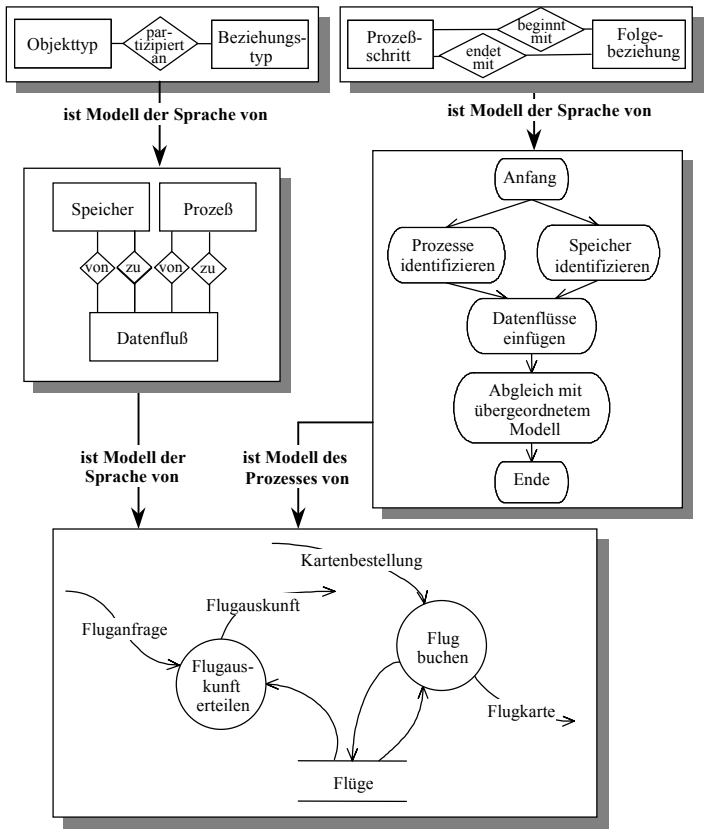


Abb. 6: Modellhierarchie mit Datenflußdiagramm auf Ebene 0

Fazit: Bildet man Metahierarchien, so sollte man diejenigen Einheiten zu Teilhierarchien zusammenfassen, innerhalb derer sich jede Ebene zu der ihr untergeordneten Ebene so verhält wie die ihr übergeordnete zu ihr selbst.

3.6 Problemkreis 6: Unklare Verankerung von Meta-Begriffen

Im Bereich von Metahierarchien, die sich über mehrere Ebenen erstrecken, ist neben der Transparenz hinsichtlich des Kriteriums, nach dem die Ebenenbildung erfolgt, auch relevant, wie man Systeme, die man mit Blick auf die unterstützten Metaebenen zu charakterisieren sind, in eine solche Hierarchie einordnet. Dabei kommt es nicht nur darauf an, wieviele Ebenen, sondern auch welche Ebenen unterstützt werden. Da Metahierarchien jedoch keine fest vorgegebenen Verankerungspunkte haben, muß eine Ebene 0 definitorisch gesetzt werden. Dieser Problemkreis wird im folgenden am Beispiel von Softwareentwicklungswerkzeugen diskutiert. Dort ist bei unklarer Verankerung bzw. Einordnung mit folgender ***Gefahr*** zu rechnen: *Werkzeuge, die verschiedene Modellierungsebenen unterstützen, lassen sich nicht alleine durch die Zahl dieser Ebenen charakterisieren. Fehlt z.B. eine klare Verankerung der Ebenen, so läßt sich ihr Einsatzgebiet nicht angemessen beschreiben.*

Beispiel: Einsatzgebiete von Meta-CASE-Tools

Herkömmlichen CASE-Tools zugrundeliegende Metamodelle sind als statisch zu betrachten, d.h. sie sind vom Benutzer des CASE-Tools nicht veränderbar. Das Schema des CASE-Tools kann in Analogie zum Schema einer Fachanwendung vom Benutzer der jeweiligen Software nicht verändert werden, sondern lediglich von ihrem Entwickler, also dem CASE-Tool-Hersteller. Dies gilt nicht für sogenannte *Meta-CASE-Tools*, die in Analogie zu XPS-Schalen auch als *CASE-Shells* bezeichnet werden.[15] In solchen generischen Entwicklungsumgebungen kann das Metamodell vom Benutzer der Umgebung selbst eingerichtet, verändert oder ergänzt werden. Die benutzerseitige Pflegbarkeit des Metamodells setzt das Vorhandensein einer Metametaebene voraus, die die vom Benutzer bei der Einrichtung oder Pflege des Metamodells verwendbaren Konzepte bestimmt.[16]

In der Literatur lassen sich bereits Ansätze finden, die das gleiche Prinzip auf eine noch höhere Stufe übertragen. Der bislang verwendeten Terminologie zufolge müßte man in diesem Fall von sog. *Meta-Meta-CASE-Tools* sprechen. Ein Beispiel hierfür ist das Werkzeug MetaEdit[17], das "multilingual specification functionality for methodology specifications"[18] bietet. Das Einsatzgebiet eines solchen Werkzeuges sehen seine Entwickler (Smolander et al.) darin, aus ein und derselben Umgebung heraus die Metamodelle verschiedener CASE-Shells pflegen und eine zentrale "metamodel base" aufbauen zu können. Voraussetzung dafür ist, daß die Metametamodelle dieser CASE-Shells in MetaEdit definiert werden. Bezüglich der außersprachlichen Welt als Gegenstandsbereich bedeutet dies das Vorliegen einer M^3-Ebene. Dennoch unterscheidet sich das Werkzeug in seiner Funktionalität nicht wesentlich von einem herkömmlichen Meta-CASE-Tool, da es wie dieses drei und nicht etwa vier Modellierungsebenen umfaßt. Das Einsatzgebiet des Werkzeuges, das auch als herkömmliches Meta-CASE-Tool verwendet werden könnte, ist jedoch ein anderes, denn der betrachtete Gegenstandsbereich sind in der Softwareentwicklung verwendete Objektsprachen. Smolander et al. sprechen in diesem Zusammenhang von CAME (Computer Aided Methodology Engineering) und charakterisieren ihr Werkzeug dadurch, daß es eine Ebene höher angesiedelt ist als eine CASE-Shell.

Abb. 7 veranschaulicht den Zusammenhang zwischen MetaEdit, einer CASE-Shell und den Modellierungsebenen. Der Vorgang der Einrichtung einer CASE-Shell unter Verwendung von MetaEdit stellt sich wie folgt dar. Das Metametamodell der CASE-Shell wird auf Metaebene in MetaEdit definiert. Auf dieser Grundlage wird in MetaEdit ein entsprechendes Modell entwickelt, das dem Metamodell der CASE-Shell entspricht und über eine Export-Routine an diese übergeben werden kann.

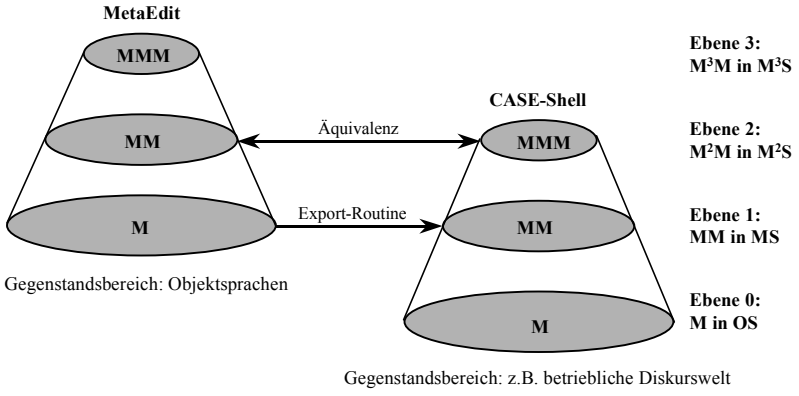


Abb. 7: Beziehung zwischen MetaEdit, CASE-Shell und Modellierungsebenen

Zusammenfassend läßt sich MetaEdit technisch gesehen als Meta-CASE-Tool verstehen, da es ebenso wie andere Werkzeuge dieser Kategorie drei Modellierungsebenen umfaßt, nicht etwa vier. Es unterstützt also nicht mehr Ebenen als vergleichbare Systeme, ist aber mit Blick auf das Einsatzgebiet in einer "gedachten" Ebenenhierarchie eine Ebene höher verankert.

Fazit: Zur Charakterisierung von Metahierarchien ist neben der Zahl der unterstützten Ebenen auch deren Verankerung wichtig, da Meta-Begriffe immer nur in Relation zum Verankerungspunkt korrekt interpretiert werden können.

4 Zusammenfassung

Im Beitrag wurde versucht, einige Problemkreise im Umfeld von Meta-Begriffen anhand von Beispielen zu veranschaulichen und die sich daraus ergebenden Gefahren kurz zu charakterisieren. Wie am Anfang des Beitrags in Tab. 1 dargestellt, läßt sich ein Teil der Problemkreise auf die grundlegende Fragestellung der expliziten Bildung und Differenzierung von Abstraktionsstufen zurückführen. Existieren Abstraktionsstufen und sollen diese explizit zum Ausdruck gebracht werden, dann stellt sich in einem zweiten Schritt die Frage nach der Begriffsbildung und definitorischen Abgrenzung. Des weiteren ergeben oder verschärfen sich einige der Probleme durch die wiederholte Bildung von Abstraktionsstufen, die zu mehrstufigen Metahierarchien führen.

Bei der Behandlung der Beispiele aus dem Umfeld von Metasprachen und Metamodellen wurde gezeigt, daß die Sprachstufentheorie der Logik einen zweckmäßigen Ausgangspunkt bildet für entsprechende Betrachtungen in der Informatik. Es wurde ausgehend von der Sprachstufentheorie ein Metamodellbegriff geprägt, der sich über die Einführung eines Metaisierungsprinzips verallgemeinern läßt. Die Offenlegung eines solchen Prinzips und eine ihm folgende Ebenenbildung sind insofern von Bedeutung, als daß nur auf diese Weise das Verhältnis zwischen Modell und Metamodell innerhalb von Modellhierarchien zum Ausdruck gebracht werden kann. Die Rolle, die ein Modell innerhalb einer Modellhierarchie einnimmt, ist immer nur in bezug auf ein Metaisierungsprinzip definiert. Es beschreibt somit zum einen die Absicht der Metamodellbildung und verhindert zum anderen, daß es bei Kombination verschiedener solcher Metamodellarten zu semantischen Unklarheiten kommt.

Insgesamt kann die exemplarische Auswahl der diskutierten Problemkreise letztlich nur eine Sensibilisierung im Hinblick auf potentielle Problemkreise in anderen Teilgebieten bewirken. Dies ist aber insofern notwendig, als daß in vielen Bereichen der Informatik eine Neigung zur "Überabstraktion" zu beobachten ist. Um so mehr ist folgendes zu beachten: Je weiter man sich vom Konkreten entfernt, um so deutlicher sind die durch Meta-Begriffe angedeuteten Abstraktionen zu konkretisieren.

Literaturverzeichnis

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.

Behme, H., Mintert, S., XML in der Praxis, Bonn 1998.

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

Ebert, J., Süttenbach, R., Uhe, I., Meta-CASE in Practice: a Case for KOGGE, in: Olive, A., Pastor, J. A., Hrsg., CAiSE ´97, LNCS 1250, Berlin und Heidelberg 1991, S. 203-216.

Essler, W. K., Die Sprache der Logik, in: Speck, J., Hrsg., Philosophie der Gegenwart I, Göttingen 1972, S. 300-339.

Flatscher, R. G., Meta-Modellierung in EIA/CDIF, Wien 1998.

Fowler, M., Analysemuster: Wiederverwendbare Objektmodelle, Bonn usw. 1999.

Habermann, H.-J., Leymann, F., Repository, München und Wien 1993.

Jarke, M., Metamodellierung: Werkzeuge für das Engineering von Unternehmensprozessen, in: Hansmann, K.-W., Scheer, A.-W., Hrsg., Praxis und Theorie der Unternehmung, Wiesbaden 1992, S. 157-175.

Kleinknecht, R., Wüst, E., Lehrbuch der elementaren Logik, Band 1: Aussagenlogik, München 1976.

Knuth, E., Halasz, F., Rado, P., SDLA: System Descriptor and Logical Analyzer, in: Olle, T.W., Sol, H. G., Verrijn-Stuart, A. A., Hrsg., Information Systems Design Methodologies: A Comparative Review, IFIP WG 8.1 Working Conference, Amsterdam 1982, S. 143-171.

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

Loos, P., Geschäftsprozeßadäquate Informationssystemadaption durch generische Strukturen, in: Vossen, G., Becker, J., Hrsg., Geschäftsprozeßmodellierung und Workflow-Management, Bonn usw. 1996, S. 163-175.

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, K., Stichworte "Metasprache", "Objektsprache", in: Mittelstraß, J., Hrsg., Enzyklopädie Philosophie und Wissenschaftstheorie, 2 Bde., Mannheim usw. 1980, S. 875, 1054-1055.

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

Martin, J., Odell, J. J., Object-Oriented Analysis and Design, Englewood Cliffs 1992.

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

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

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

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

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., CAiSE ´92, LNCS 593, Berlin und Heidelberg 1992, S. 546-567.

Runggaldier, E., Analytische Sprachphilosophie, Stuttgart usw. 1990.

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., CAiSE ´91, LNCS 498, Berlin und Heidelberg 1991, S. 168-193.

Stachowiak, H., Allgemeine Modelltheorie, Wien und New York 1974.

Stegmüller, W., Wissenschaftliche Erklärung und Begründung, Berlin usw. 1974.

Strahringer, S., Metamodellierung als Instrument des Methodenvergleichs: Eine Evaluierung am Beispiel objektorientierter Analysemethoden, Aachen 1996.

Verhoef, T. F., Hofstede, A. H. M. ter, Wijers, G. M., Structuring modelling knowledge for CASE shells, in: Andersen, R. et al., Hrsg., CAiSE ´91, LNCS 498, Berlin und Heidelberg 1991, S. 502-524.

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.

---

[1] Zur Sprachstufentheorie vgl. Bochenski (1965, S. 59 f.), Essler (1972, S. 302 ff.), Kleinknecht/Wüst (1976, S. 40 ff.), Lorenz (1980, S. 875, S. 1054), Runggaldier (1990, S. 63), Stachowiak (1973, S. 217), Stegmüller (1974, S. 30-32).

[2] Vgl. zum Thema XML z.B. Behme/Mintert (1998) oder http://www.xml.com und http://www.w3.org/xml.

[3] DTD steht für Document Type Definition und ist die Bezeichnung für eine in XML oder SGML formulierte Definition einer Auszeichnungssprache.

[4] Beispiele finden sich bei Loos (1995, S. 57, 1996, S. 168), Loos/Scheer (1995, S. 192), Odell (1992, S. 47) oder Fowler (1999).

[5] Vgl. Loos (1995, S. 57), Loos/Scheer (1995, S. 191), Loos (1996).

[6] Vgl. Fowler (1999, S. 30).

[7] Vgl. zu dieser Thematik vertiefend Strahringer (1996, S. 35-36), Martin/Odell (1992, S. 489-483), Odell (1992), Partridge (1994).

[8] Siehe zu einer solchen Untersuchung z.B. Strahringer (1996, S. 11-16).

[9] Dieser terminologische Konflikt tritt im Zusammenhang mit der objektorientierten Modellierung auf.

[10] Man beachte, daß die Numerierung der Ebenen in Analogie zur Numerierung der Sprachstufen erfolgt. Andere Autoren, z.B. Flatscher (1998, S. 32), bezeichnen die Modellebene als 1. nicht wie hier 0. Ebene. Sie sprechen dann von M1, M2, M3 etc., wobei die Ziffer nach dem "M" die Anzahl an "M"s angibt, mit denen die einzelnen Wörter des zusammengesetzten Begriffs beginnen. Folglich steht M3 für **M**eta-**M**eta-**M**odell. In der hier vorgeschlagenen Nomenklatur läßt sich hierfür M^2M oder MMM schreiben. Nur diese Konvention läßt sich unmittelbar auf Metasprachen, Metaprozesse etc. übertragen.

[11] 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)". Vornehmlich erstere sind im vorliegenden Kontext relevant, eine Erweiterung auf die anderen Formen ist grundsätzlich möglich.

[12] 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).

[13] Nur in wenigen Arbeiten ist ein ausgewogenes Verhältnis zwischen beiden Aspekten (Sprache/Prozeß) anzutreffen. Verhoef/Hofstede/Wijers (1991) beispielsweise betonen ausdrücklich die Sprach/Prozeß-Dualität bei Betrachtung einer Methode - in ihren Worten als "the way of modelling" vs. "the way of working" bezeichnet. Sie wenden Metamodellierung auch konsequenterweise auf beide Aspekte an.

[14] Es handelt sich um ein stark vereinfachtes Modell des Flugkartenverkaufs einer Fluggesellschaft, das an ein Beispiel aus Raasch (1993, S. 91) angelehnt ist.

[15] Vgl. zu Meta-CASE-Tools z.B. Ebert/Süttenbach/Uhe (1997), Smolander et al. (1991), Rossi et al. (1992), Verhoef/Hofstede/Wijers (1991). Alderson (1991, S. 81-82) faßt den Begriff Meta-CASE etwas weiter und zählt auch Sprachen der 4. Generation und Compiler-Compiler dazu. Der hier gewählte engere Meta-CASE-Begriff entspricht in seiner Terminologie einem "methods meta CASE" (S. 90). Knuth/Halazs/Rado (1982) ist eine der früheren Arbeiten zu Meta-CASE-Systemen, in der bereits sehr deutlich die grundlegende Idee der "so called *meta systems*" dargestellt wird. "It is possible to construct a set of computer aided tools *independently* of the application language and then these can be supplemented with those mechanisms facilitating language definitions." (Knuth/Halazs/Rado (1982, S. 143)).

[16] Das Prinzip der Metamodellpflegbarkeit bei Meta-CASE-Tools entspricht dem ANSI-IRDS-Konzept aus dem Bereich der Repositories. Siehe hierzu auch Strahringer (1996, S. 29-31) bzw. zum Thema im allg. Habermann/Leymann (1993).

[17] Das Werkzeug wird in Smolander et al. (1991) sowie Rossi et al. (1992) vorgestellt.

[18] Smolander et al. (1991, S. 174).