Zur Standardisierung objektorientierter Modellierungssprachen: Eine kritische Betrachtung des State of the Art am Beispiel der Unified Modeling Language

Prof. Dr. Ulrich Frank
Dipl. Inf. Michael Prasse
Institut für Wirtschaftsinformatik, Universität Koblenz-Landau
Rheinau 1, 56075 Koblenz

Zusammenfassung

Der vorliegende Beitrag ist durch die zur Zeit laufenden Bemühungen der Object Management Group (OMG) motiviert, objektorientierte Modellierungssprachen zu standardisieren. Eine solche Standardisierung verspricht eine Reihe wirtschaftlicher Vorteile. Angesichts der nach wie vor wenig einheitlichen Begriffe und Konzepte stellt sich allerdings die Frage, ob der gegenwärtige Stand der Kunst eine Reife erreicht hat, die eine Standardisierung rechtfertigt. Um dieser Frage nachzugehen, wird zunächst das aktuelle Standardisierungsprocedere der OMG skizziert und kritisch hinterfragt. Anschließend werden Anforderungen an Modellierungssprachen formuliert. Dies geschieht einerseits, um offene Fragen der einschlägigen Forschung zu verdeutlichen, andererseits, um eine neuere Modellierungssprache zu beurteilen, die einen Eindruck vom herrschenden State of the Art vermittelt und zudem allem Anschein nach den zu erwartenden Standard nachhaltig prägen wird.

1. Motivation

Die rege Forschung im Bereich der objektorientierten Modellierung hat zu einer kaum überschaubaren Vielfalt von Ansätzen geführt. Auch wenn sich die zögerliche Verbreitung der objektorientierten Modellierung in der betrieblichen Praxis (vgl. als Indikator dafür etwa [Schn97]) im Vergleich dazu auf wenige Ansätze (wie etwa [Boo94], [Rum91], [Jac92], [CoYo91]) beschränkt, so stellt doch auch diese verbleibende Vielfalt für Unternehmen, die ihre Software-Entwicklung zukünftig objektorientiert betreiben möchten, eine kaum tragbare Gefährdung von Investitionen in die Schulung von Mitarbeitern und begleitende Werkzeugumgebungen dar. Vor diesem Hintergrund erscheinen die aktuellen Bemühungen der Object Management Group (OMG), eine Standardisierung objektorientierter Modellierungssprachen zu erreichen, ausgesprochen sinnvoll.

Bei allen Vorzügen, die standardisierte Modellierungssprachen in Aussicht stellen - hier ist gewiß nicht nur an die Interoperabilität von CASE-Werkzeugen zu denken, sondern auch an die Spezifikation von Referenzmodellen für betriebliche Informationssysteme als Grundlage sog. "Business Objects", bleibt dennoch die Frage, ob die entsprechenden Konzepte einen Reifegrad erreicht haben, der eine Standardisierung zum gegenwärtigen Zeitpunkt rechtfertigt.

2. Die Standardisierungsinitiative der OMG

Die Vision der OMG umfaßt nicht nur transparente, plattformunabhängige Verteilung, sondern auch die Wiederverwendbarkeit anwendungsnaher, besonders anschaulich dargestellter Konzepte. Es liegt auf der Hand, daß objektorientierte Programmiersprachen oder die von der OMG spezfizierte Interface Defini-

tion Language (IDL) für entsprechende Darstellungen kaum geeignet sind. Man hat denn auch bei der OMG schon früh die Bedeutung der objektorientierten Modellierung erkannt: 1992 wurde die "Object Analysis and Design Special Interest Group" [KaCh92] gegründet mit dem Ziel, "... to formalise the definitions of the concepts used for analysis and design" ([KaCh92], S. 12). Gleichzeitig wurde die Verfügbarkeit standardisierter Modellierungssprachen immer dringlicher; einerseits um die Austauschbarkeit von Modellen zwischen verschiedenen CASE-Werkzeugen zu ermöglichen, andererseits um die Voraussetzung für die Standardisierung sog. "Business Objects" zu schaffen, die mit Hilfe geeigneter Objektmodelle dargestellt werden ([OMG96a], S. 5). Da die erwähnte Arbeitsgruppe der OMG ihr Ziel nicht erreichte, erfolgte im Juni 1996 ein "Request for Proposals" ([OMG96b]), in dem Vorschläge für die Standardisierung objektorientierter *Modellierungssprachen* erbeten wurden. In der Ausschreibung wird relativ offen gelassen, welche Art von Modellen durch die vorzuschlagenden Sprachen erstellt werden sollen. So ist die Rede von "static models", "dynamic models" und "usage models", aber auch von "... other models of value during analysis and design." ([OMG96b], S. 3) Als Hauptanforderung wird die Unterstützung der Interoperabilität von CASE-Werkzeugen unterstrichen. Gleichzeitig wird allerdings auch auf die Bedeutung von Modellen als Medium zwischenmenschlicher Kommunikation verwiesen ([OMG96b], S. 4). Dabei fällt auf, daß die Aufforderung der OMG eine Reihe von Fragen offenläßt. So präsentiert sich schon die Zielsetzung in mehrdeutiger Weise: Es macht einen Unterschied, ob man Modellierungssprachen zum Zweck der Abbildung in Werkzeugen formalisiert, oder ob man den Fokus auf die Brauchbarkeit einer Sprache für die konzeptuelle Modellierung richtet - auch wenn beide Ziele nicht im Konflikt liegen müssen. Darüber hinaus wird nicht näher darauf eingegangen, wie - also z.B. in welchem Formalisierungsgrad - die Sprachen beschrieben werden sollen. Desweiteren wird nicht deutlich, wie sich die Vorschläge zu existierenden oder entstehenden Standards verhalten sollen. Hier ist beispielsweise an das CASE Date Interchange Format (CDIF, [CDI96]) oder auch an die "Workflow Process Definition Language" der "Workflow Management Coalition" (WFMC, [WFM96]) zu denken.

3. Anforderungen an objektorientierte Modellierungssprachen

In dem hier bedeutsamen Kontext ist eine Modellierungssprache eine Sprache, die zum Zweck der konzeptuellen Modellierung eingeführt wird. Sie beinhaltet zumeist, aber nicht notwendigerweise, eine graphische Notation. Syntax und Semantik einer Modellierungssprache können mehr oder weniger formal sein. Sie werden unter Rückgriff auf eine Metasprache beschrieben, die selbst wieder als Modellierungssprache ausgelegt sein kann. In diesem Fall spricht man von einem Metamodell. Da es ggfs. hilfreich ist, ein komplexes System mit unterschiedlichen Abstraktionen zu beschreiben, kann eine Modellierungssprache mehrere Modell- bzw. Diagrammarten (etwa Objektmodelle oder Zustandsdiagramme) unterstützen. Wir formulieren im folgenden Anforderungen an Sprachen für die (objektorientierte) konzeptuelle Modellierung, die im Unterschied zu dem vordergründigen Ziel der gegenwärtigen Initiative der

OMG weniger an der Austauschbarkeit von Modellen zwischen Werkzeugen orientiert sind, als vielmehr an den Erfordernissen der konzeptuellen Modellierung. Dabei werden wir nicht allein allgemeine Aspekte berücksichtigen, sondern auch solche, die aus den Besonderheiten des Entwurfs betrieblicher Informationssysteme entstehen.

Die konzeptuelle Modellierung ist eine intellektuelle Tätigkeit im Rahmen der Software-Entwicklung, die vor allem darauf zielt, eine natürliche Abbildung der für relevant erachteten Bestandteile einer Anwendungsdomäne zu schaffen. Aus diesem Grund sollte weitgehend von Randbedingungen abstrahiert werden, die durch die spätere Implementierungsumgebung gesetzt werden. Gleichzeitig aber liefern konzeptuelle Modelle Vorgaben für die Implementierung, so daß daraus erwachsende Besonderheiten auch berücksichtigt werden müssen. Vor dem Hintergrund dieses Spannungsfelds sind die Anforderungen an Modellierungssprachen zu betrachten.

Anschaulichkeit/Verständlichkeit

Ein Modell ist anschaulich, wenn es möglichst direkt mit Wahrnehmungsmustern und Konzeptualisierungen des Betrachters korrespondiert. Während die Anschaulichkeit eines Modells auch von der Verwendung der Modellierungssprache abhängt (also z.B. der Wahl von Bezeichnern), hat die Modellierungssprache selbst insofern einen erheblichen Einfluß, als die jeweils verfügbaren Sprachelemente sowie die jeweils zulässigen Verknüpfungen die Darstellung wesentlich prägen. Ergänzend ist dabei zu berücksichtigen, daß neben der - mehr oder weniger präzise definierten - Syntax und Semantik einer Sprache die Verständlichkeit eines Modells auch von der graphischen Notation abhängt. Allgemein läßt sich vermuten, daß Syntax und Semantik einer Modellierungssprache tendenziell dann als anschaulich empfunden werden, wenn sie eine deutliche Nähe zu Sprachmitteln aufweisen, die der jeweilige Betrachter aus ihm vertrauten Sprachen kennt - also beispielsweise das generelle Muster "Subjekt, Prädikat, (Objekt)". Zudem kann man davon ausgehen, daß die Anschaulichkeit von Modellen mit zunehmender Komplexität von Syntax und Semantik abnimmt. Während dieser letzte Punkt für die Anschaulichkeit des Objekt/Beziehung-Paradigmas, das ja auch der objektorientierten Modellierung in wesentlichen Teilen zugrundeliegt, spricht, zeigen empirische Untersuchungen, daß Entity Relationship Modelle nicht von allen Menschen als anschaulich empfunden werden ([Hit95], [GoSt90]). Vor dem Hintergrund von in subtiler Form streuenden Wahrnehmungsfähigkeiten der Betrachter läßt sich der Beitrag einer Modellierungssprache zur Anschaulichkeit von Modellen kaum in objektiver Weise beurteilen. Im Hinblick auf generell verwendbare Modellierungssprachen kann man lediglich fordern, daß der genannten Varianz dadurch begegnet wird, daß unterschiedliche Detaillierungs- und Formalisierungsstufen von Modellen unterstützt werden. Die graphische Notation sollte eine deutliche Abbildung von Sprachkonzepten auf graphische Symbole bieten, zudem sollten die Symbole semantisch nicht überladen sein (vgl. dazu

auch [Rum96a]). Im Hinblick auf die Modellierung betrieblicher Informationssysteme könnte die Anschaulichkeit durch Sprachkonzepte gefördert werden, die vertraute Darstellungen spezieller Sichten - wie etwa von Geschäftsprozessen oder Organisationsstrukturen - ermöglichen.

Angemessenheit/Mächtigkeit

Eine Modellierungssprache unterstützt die Erstellung angemessener Modelle, wenn sie die für den jeweiligen Modellierungszweck erforderlichen Aspekte des abzubildenden Realitätsausschnitts in einer sinnvollen Detaillierung und Formalisierung abzubilden gestattet. Konzepte, die für die Modellierung häufig benötigt werden (z.B. Beziehungsarten), sollten von der Modellierungssprache bereitgestellt werden. Wenn man fordert, daß objektorientierte Modellierungssprachen den Lebenszyklus möglichst umfassend abdecken sollten, bedeutet dies, daß möglichst alle - zwischen den Phasen bekanntlich variierenden Modellierungsziele - berücksichtigt werden. Für die späte Entwurfsphase heißt das u.a., daß dem Modellierer die Möglichkeit gegeben wird, Integritätsbedingungen in einem ihm sinnvoll erscheinenden Maß formal zu spezifizieren.

Wiederverwendbarkeit

Da sich Wiederverwendbarkeit gewiß nicht auf einzelne Klassen beschränkt, ist hier an die Möglichkeit zu denken, Ausschnitte von Modellen, die für bestimmte Verwendungskontexte eine Einheit bilden, zusammenzufassen. Dabei sollten mögliche Seiteneffekte auf andere Modellteile deutlich darstellbar sein. In diesem Zusammenhang sind auch abstrakte Entwurfsartefakte zu nennen, wie sie etwa unter dem Schlagwort "Design Pattern" diskutiert werden. Wiederverwendbarkeit wird zudem grundsätzlich gefördert, wenn die zur Modellierung verwendbaren Konzepte Generalisierungs-, bzw. Spezialisierungsmöglichkeiten bieten.

Integration

Im Rahmen der Software-Entwicklung entsteht gewöhnlich eine Reihe von Teilmodellen, die unterschiedliche Sichten auf das zu erstellende System beschreiben. Darüber hinaus werden diese Modelle im Laufe des Lebenszyklus schrittweise verfeinert. Eine Modellierungssprache sollte eine möglichst enge Integration von Teilmodellen, also beispielsweise von Objektmodellen und Nachrichtenfluß-Diagrammen, ermöglichen. Dazu ist es erforderlich, daß die Beziehungen zwischen den für die Teilmodelle verfügbaren Sprachkonzepten eindeutig beschrieben werden. Das gleiche gilt für die Beziehung zwischen den für die einzelnen Phasen des Lebenszyklus verwendeten Detaillierungs- bzw. Formalisierungsstufen.

Flexibilität, Erweiterbarkeit, Anpaßbarkeit

Modellierungssprachen reflektieren notwendigerweise bestimmte Vorstellungen über die in den verschiedenen Tätigkeiten, wie etwa Analyse und Entwurf, zu erhebenden bzw. festzulegenden Angaben.

Da sich solche Vorstellungen im Zeitverlauf wie auch mit der je betrachteten Domäne ändern mögen, sollte eine Modellierungssprache erweiterbar sein. Im Hinblick auf die Austauschbarkeit von Modellen sollte eine solche Erweiterung die Semantik ursprünglicher Sprachmittel nicht berühren. Dazu ist es wesentlich, daß zulässige Erweiterungen im Metamodell festgelegt sind.

Dokumentation

Die Dokumentation sollte die Sprachelemente in einer verständlichen Form beschreiben. Angesichts der mangelnden Einheitlichkeit objektorientierter Terminologie gehört dazu einerseits die Definition zentraler Begriffe wie Klasse, Objekt, etc., andererseits eine Erläuterung des intendierten Sprachgebrauchs (also z.B. der Verwendung von Attributen - etwa im Unterschied zu Beziehungen). Die ingenieurmäßige Entwicklung von Software erfordert zudem eine (semi-) formale Definition der Semantik konzeptueller Modelle mittels geeigneter Metasprachen bzw. Metamodelle. Ein Metamodell sollte ein korrektes, vollständiges und möglichst einfaches Modell einer Modellierungssprache sein. Im Idealfall bedeutet dies u.a., daß aus dem Metamodell alle zulässigen Modelle generiert werden können und daß für jedes Modell entschieden werden kann, ob es zulässig ist (eine ausführliche Darstellung dieser Anforderungen findet sich in [SüEb97], S. 2 f.).

4. Charakterisierung des State of the Art am Beispiel der UML

Die "Unified Modeling Language" (UML) der Firma Rational gehört zu den sechs bis zum Einsendeschluß am 17. Januar 1997 bei der OMG eingegangen Vorschlägen, unter denen sie eine besondere Stellung einnimmt. So stellt sie die Vereinheitlichung und Weiterentwicklung dreier bereits bekannter Modellierungsansätze ([Boo94], [Jac92], [Rum91] dar. Darüber hinaus wird sie von einem beeindruckenden Industrie-Konsortium (u.a. sind Microsoft, Hewlett-Packard, Oracle und Texas Instruments beteiligt) vorgeschlagen. Zudem fällt auf, daß sich andere Vorschläge zum Teil nur als Erweiterungen bzw. Modifikationen der UML verstehen.

Hintergrund

Nachdem Rational eine vereinheitlichte Modellierungsmethode unter dem Namen "Unified Method" ([Rum96b]) angekündigt hatte, entschloß man sich später, wohl auch weil die Vereinheitlichung von Modellierungsprozessen grundsätzlich fragwürdig ist, zunächst eine objektorientierte Modellierungssprache unter dem Etikett "Unified Modeling Language" zu entwerfen. Die ersten Darstellungen erschienen im Herbst 1996, die uns vorliegende Version 1.0, zugleich der an die OMG eingereichte Vorschlag, datiert vom 13.2.1997. Die Version ist in insgesamt 13 Dokumenten beschrieben, die zusammen mehr als 550 Seiten umfassen. Zwei dieser Dokumente sind dabei von wesentlicher Bedeutung ([Rat97a], [Rat97b]). Im Unterschied zu den bekannten Lehrbüchern der drei Autoren fällt die Beschreibung der

Sprache nicht nur wesentlich ausführlicher aus, sie erfolgt zudem durchgängig in Form objektorientierter Metamodelle - also semi-formal - unter Verwendung der für die UML selbst vorgesehenen Notation. Es ist bemerkenswert, daß es bereits zum gegenwärtigen Zeitpunkt eine Reihe von Lehrbüchern außenstehender Autoren gibt, in denen die UML bzw. die Unified Method als der zukünftige Standard für objektorientierte Modellierungssprachen gefeiert wird ([Bur97], [Neu97], [Oes97]).

Anschaulichkeit

Die Sprachmittel der UML sind im Vergleich zu ihren direkten Vorläufern sehr umfangreich. Zunächst fällt auf, daß man die konventionellen Konzepte, die in OMT enthalten sind (wie etwa Datenflußdiagramme), aufgegeben hat. Die UML unterstützt neun Diagrammarten. Zur Darstellung statischer Zusammenhänge können einerseits Klassen- und Objektdiagramme verwendet werden, andererseits werden, speziell für den späten Entwurf und die Implementierung zwei weitere Diagrammarten angeboten. "Component Diagrams" dient der Abbildung existierender Software-Kompenenten und der zwischen ihnen bestehenden Abhängigkeiten. Im Unterschied dazu sind "Deployment Diagrams" auf die Abhängigkeiten zwischen Laufzeitsystemen über Plattformgrenzen hinweg gerichtet. Nachrichtenflüsse zwischen Objekten können mit Hilfe von "Collaboration Diagrams" dargestellt werden. Der Abbildung dynamischer Aspekte dienen "Sequence Diagrams", "State Diagrams" und "Activity Diagrams". Darüber hinaus sind "Use Case Diagrams" vorgesehen, wie sie u.a. von Jacobson [Jac92] vorgeschlagen wurden.

Gleichwohl eine solche Vielfalt von Diagrammarten die Aussichten auf Anschaulichkeit erhöhen mag, ist zu bemängeln, daß sich deren Einsatzbereiche teilweise überschneiden (z.B. "Sequence Diagram", "State Diagram" und "Use Case Diagram"). Trotz dieser Vielfalt bleibt anzumerken, daß spezielle Diagrammtechniken, wie sie für den Entwurf betrieblicher Informationssysteme sinnvoll sind, nicht angeboten werden. Die Anschaulichkeit der graphischen Notation soll hier nicht kommentiert werden.

Angemessenheit

Die erwähnten Diagrammarten scheinen prinzipiell geeignet, wesentliche Zusammenhänge in allen Phasen des Lebenszyklus abzudecken. Es ist allerdings fraglich, ob die angebotenen Sprachmittel immer eine angemessene Detaillierung bzw. Formalisierung erlauben. So wird beispielsweise eine detaillierte Spezifikation von Attributen, wie sie etwa von Booch ([Boo94]) vorgesehen ist, mit der schwer nachvollziehbaren Begründung abgelehnt, es handele sich hierbei um Angaben, die abhängig von der jeweiligen Programmiersprache sind. Explizite Integritätsbedingungen können nur natürlichsprachlich angegeben werden. Grundsätzlich bleibt anzumerken, daß ein erhebliches Maß an Erfahrung nötig ist, um die für eine bestimmte Abstraktion angemessenen Sprachmittel zu identifizieren. Hier kann man bei einigen

Diagrammarten (wie etwa "Activitiy Diagrams", "Deployment Diagrams") Zweifel haben.

Wiederverwendbarkeit

Elemente aller Diagramme können zu sog. "Packages" zusammengefaßt werden. Da die Verwendung von Packages im Hinblick auf die Erzeugung wiederverwendbarer Teile nicht weiter diskutiert wird, handelt es sich hierbei vor allem um einen generellen Mechanismus zur Unterstützung der Komposition/ Dekomposition von Modellen. Zur Darstellung von Entwurfsmustern wird - abweichend von gängigen Dokumentationsformen ([Gam95]) - vorgeschlagen, Schablonen für Objektinteraktionen ("Collaboration Diagrams") zu verwenden ([Rat97a], S. 66). Im Hinblick auf Spezialisierungsmöglichkeiten der vorgeschlagenen Konzepte ist zu erwähnen, daß eine Generalisierung/Spezialisierung für Use Cases skizziert wird, allerdings ohne deren Semantik zu beschreiben ([Rat97b], S. 63 ff.).

Flexibilität

Es sind drei Mechanismen zur Erweiterung der UML vorgesehen. "Stereotypes" erlauben die Einführung neuer Sprachelemente durch die Auszeichnung existierender Sprachmittel, die hinsichtlich bestimmter Eigenschaften gleichartig sind ([Rat97a], S. 19). Es werden also letztlich neue Klassifikationen von Sprachelementen eingeführt. Es handelt sich allerdings nicht um eine Spezialisierung. Stereotypen werden zur Spezifikation der UML selbst verwendet - beispielsweise zur Klassifikation von Beziehungen zwischen Modellelementen und "Packages" in "private", "friend" und "public" ([Rat97a], S. 12). Stereotypes sind insofern mächtig, daß sie nahezu beliebige Erweiterungen zulassen. Es ist allerdings als wesentliche Einschränkung anzusehen, daß keine hinreichenden Hilfsmittel zur Formalisierung dieser Erweiterungen angeboten werden. Ihre Semantik ergibt sich also im wesentlichen aus der Interpretation durch den Betrachter und/oder durch spezielle Werkzeuge. "Tagged Values" sind durch möglichst selbstsprechende Bezeichner zu kennzeichnende Sprachelemente, deren Semantik i.d.R. nicht spezifiziert ist. Auch sie werden zur Beschreibung der UML selbst verwendet. So werden beispielsweise Vor- und Nachbedingungen als "Tagged Values" zur Beschreibung von Operationen eingeführt. Die dritte Erweiterung wird durch frei definierbare "Constraints" möglich, die selbst als ein Stereotyp textueller Annotationen eingeführt werden. Auch für Constraints ist also, wie bereits erwähnt, keine Formalisierung vorgesehen. Aus unserer Sicht sind noch die sog. "Business Modeling Stereotypes" ([Rat97c], S. 8) erwähnenswert, wie etwa "work unit", "worker" usw. Es fehlt allerdings jegliche Beschreibung der Semantik dieser Klassifikationen.

Integration

Der Rückgriff auf ein gemeinsames Metamodell ist grundsätzlich gut geeignet, die Integration der verschiedenen Diagrammarten zu beschreiben. Diese Chance wird allerdings nur in eingeschränkter Weise

genutzt, da die verwendeten Konzepte teilweise nur oberflächlich modelliert sind. Ein besonders krasses Beispiel dafür sind Use Cases.

Dokumentation/Formalisierung

Durch die Verwendung eines Metamodells ist die Beschreibung der UML wesentlich präziser als es für ihre Vorgänger der Fall war. Auch die Trennung der Sprachbeschreibung von der Darstellung der graphischen Notation ist positiv zu werten. Die Beschreibung erfolgt allerdings nicht vollständig durch das Metamodell, sondern wird durch natürlichsprachliche Erläuterungen ergänzt. Die Verständlichkeit der Dokumentation ist sicher abhängig vom Betrachter, wir wagen allerdings die Behauptung, daß die Darstellungen auch für sachkundige Betrachter mitunter schwer nachvollziehbar sind - was das folgende Zitat veranschaulichen mag: "Instance of is an association between an Instance instance and its Type instance. The responsibility of instance of is to specify that the Instance instance is a concrete manifestation of the Type instance. Every Type instance may have zero or more Instance instances, and every Instance instance is the instance of not more than one Type instance." ([Rat96a], S. 32) Auch die zentrale Unterscheidung von "Type", "Class" und "Interface" wird u.E. nicht überzeugend begründet. Danach ist ein Typ die Festlegung einer Domäne zusammen mit den Operationen, die auf diese Domäne anwendbar sind. Eine Klasse beschreibt eine Realisation eines Typs. "Class is a subtype of Type, and therefore instances of Class have the same property as instances of Type. The fundamental difference being that Type instances specify interfaces, whereas Class instances specify the realization of these interfaces." ([Rat96a], S. 57) In [Mey97] werden Klassen als Modelle abstrakter Datentypen beschrieben, wobei auf die Bedeutung der Axiome zur Beschreibung der Semantik hingewiesen wird. Werden Typen nun auf Signaturen oder Schnittstellen reduziert, wie dies in UML geschieht, so fehlt jegliche semantische Beschreibung der Funktionalität. Die konkrete Implementierung einer Klasse kann ja gerade als operationale Semantik aufgefaßt werden. Es kann durchaus sinnvoll sein, eine Klasse in ihr Interface und ihre Implementierung zu trennen. Die Verwendung des Begriffs Typ für die Schnittstelle ohne Berücksichtigung der dynamischen Semantik führt u.E. jedoch zu einer Begriffsverwirrung. Der Verwendungszweck der angebotenen Sprachelemente wird in der Dokumentation allenfalls am Rande behandelt.

5. Abschließende Bemerkungen

Die im vorliegenden Beitrag nur in Grundzügen dargestellte Charakterisierung der UML sollte deutlich machen, daß gegenüber ihren Vorläufern erhebliche Fortschritte gemacht wurden. Dessen ungeachtet weist auch die UML einige konzeptionelle Lücken und Schwächen auf. Dabei ist allerdings zu berücksichtigen, daß einige der aufgezeigten Unzulänglichkeiten durch den Rückgriff auf andere bei der OMG eingereichte Vorschläge kompensiert werden könnten (ein Überblick findet sich in [Fra97]). Besonders bemerkenswert ist u.E. ein Vorschlag ([IBM97]), in dem eine sog. "Object Constraint Language" spezifi-

ziert wird. Sie dient der Beschreibung des Metamodells der Modellierungssprache und erlaubt dadurch präzise definierte Spracherweiterungen. Ein weiterer wichtiger Sprachentwurf ist die "OPEN Modeling Language" (OML, [FiHe96]). Im Unterschied zur UML ist die OML Ausdruck einer vor allem wissenschaftlich motivierten Initiative - was auch der Grund dafür ist, daß sie nicht bei der OMG eingereicht werden konnte, da die OMG u.a. innerhalb eines Jahres eine kommerzielle Verfügbarkeit fordert ([OMG96b], p. 24) - was immer das bei einer Modellierungssprache heißen mag. Gegenüber der UML weist die OML eine Reihe attraktiver Eigenschaften auf (ein Vergleich findet sich in [FrPr97]). Neben der OML gibt es eine Fülle wissenschaftlicher Beiträge zu einzelnen Aspekten der objektorientierten Modellierung, die von den bei der OMG eingereichten Vorschlägen nicht reflektiert werden. Hier ist beispielsweise an die Einführung korrespondierender Modellierungssprachen, wie sie etwa für die Beschreibung von Workflows eingesetzt werden können, zu denken. In diesem Zusammenhang ist auch anzumerken, daß die OMG selbst das Verhältnis der zu standardisierenden Modellierungssprache zur "Workflow Process Definition Language" der "Workflow Management Coalition" bisher nicht geklärt hat. Ein weiterer Grund, der gegen eine Standardisierung zum gegenwärtigen Zeitpunkt spricht, ist darin zu sehen, daß es bisher keine hinreichenden Untersuchungen ergonomischer Aspekte objektorientierter Modellierungssprachen gibt. Das gilt einerseits für die Akzeptanz der Sprachkonzepte selbst, andererseits für die gewiß nicht triviale Festlegung der graphischen Notation.

Aus der Sicht der Wirtschaftsinformatik wirft das gegenwärtige Standardisierungsprocedere zudem einige grundsätzliche Frage auf. So ist durchaus nicht überzeugend geklärt, ob objektorientierte Modellierung das leistungsfähigste Paradigma für die Beschreibung fachlicher Konzepte darstellt. Bei aller "Natürlichkeit" ist es auch durch Randbedingungen geprägt, die im Programmiersprachenentwurf begründet sind. Es mag also durchaus Sprachen geben, die für diesen Zweck besser geeignet sind (vgl. dazu [Ort97]). Daneben stellt sich die Frage, welche Rolle eine wissenschaftliche Disziplin, zu deren Kerngebieten die Modellierung gehört, bei einschlägigen Standardisierungsvorhaben, die von der Industrie dominiert werden, spielen könnte - und sollte.

Literatur

[Boo94] Booch, G.: Object-oriented Analysis and Design with applications. 2. Aufl., Benjamin Cummings, Redwood/CA., 1994

[Bur97] Die Unified Modeling Language. Objektorientierte Modellierung für die Praxis. Bonn u.a.: Addison Wesley 1997

[CDI96] CDIF: Harmonization of CDIF with other Standards Bodies, 96-07-26, 1996. Obtained via http://www.cdif.org/intro.html

[CoYo91] Coad, P.; Yourdon, E.: Object Oriented Design. Englewood Cliffs, NJ: Prentice Hall 1991

[FiHe96] Firesmith, D.; Henderson-Sellers, B.; Graham, I.; Page-Jones, M.: Specification of the Common Object Modeling Notation (COMN). Version 1.0. 13.11.96. OPEN Consortium 1996, (http://www.csse.swin.edu.au/OPEN/comn.html)

[Fra97] Frank, U.: Towards a Standardisation of Object-Oriented Modelling Languages? Arbeit-

spapiere des Instituts für Wirtschaftsinformatik. Nr. 3, Universität Koblenz-Landau 1997

[FrPr97] Frank, U.; Prasse, M.: A Comparison of the Unified Modelling Language (UML) and the Open Modelling Language (OML). Arbeitspapiere des Instituts für Wirtschaftsinformatik. Nr. 5, Universität Koblenz-Landau 1997

[Gam95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Reading/Mass. etc.: Addison-Wesley 1995

[GoSt90] Goldstein, R.C.; Storey, V.C.: Some Findings on the Intuitiveness of Entity Relationship Constructs. In: Lochovsky, F.H. (Hrsg.): Entity Relationship Approach to Database Design and Query. Amsterdam: Elsevier 1990

[Hit95] Hitchman, S.: Practitioner Perceptions on the Use of some Semantic Concepts in the Entity Relationship Model In: European Journal of Information Systems. Vol. 4, 1995, S. 31-40

[IBM97] IBM; ObjecTime Limited: OMG OA&D RFP Response Version 1.0, 1997 (ftp://ftp.omg.org/pub/docs/ad/97-01-18.pdf)

[Jac92] Jacobson, I. et al.: Object-Oriented Software Engineering. A Use Case Driven Approach. Reading/Mass.: Addison Wesley 1992

[KaCh92] Kain, J.B.; Christopherson, M. et al.: Object Analysis and Design. OMG Document 92-10-01.PDF, draft 7.0, 1992. Obtained via http://www.omg.org/library/public-doclist.html

[Mey97] Meyer, B.: Object-Oriented Software Construction. 2nd. ed., Englewood Cliffs: Prentice Hall 1997

[Neu97] Neumann, H.: Objektorientierte Entwicklung mit der Unified Method. München: Hanser 1997

[Oes97] Oesterreich, B.: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language. München: Oldenbourg 1997

[OMG96a] Object Management Group: Common Facilities RFP-4: Common Business Objects and Business Object Facility. OMG, TC 13CF/96-01-04, 1996. Obtained via http://www.omg.org/library/public-doclist.html

[OMG96b]OMG: Object Analysis & Design RFP-1, 6/6/1996 (http://www.omg.com)

[Ort97] Ortner, E.: Methodenneutraler Fachentwurf. Wiesbaden: Teubner 1997

[Rat96a] Rational: UML Semantics. Version 1.0. 13.02.1997 1997 (http://www.rational.com)

[Rat96b] Rational: UML Notation Guide. Version 1.0. 13.02.1997 1997, (http://www.rational.com)

[Rat96b] Rational: UML Process Specific Extensions. Version 1.0. 13.02.1997 1997, (http://www.rational.com)

[Rum91] Rumbaugh, J. et al.: Object-oriented Modelling and Design. Englewood Cliffs: Prentice Hall, N.J. 1991

[Rum96a] Rumbaugh, J.: Notation notes: Principles for choosing notation. In: Journal of Object-Oriented Programming. Mai 1996, S. 11-14

[Sch97] Schnur, B.: Objektorientierung in Versicherungsunternehmen. Die Branche gibt sich bislang noch zurückhaltend In: Informatik Spektrum, 20. Jg., 1997, Heft 1, S. 52-53

[SüEb97] Süttenbach, R.; Ebert, J.: A Booch Metamodel. Fachberichte Informatik, 5/97, Universität Koblenz-Landau 1997

[WFM96] WfMC (Workgroup 1): Interface 1: Process Definition Interchange WfMC TC-1016, Version 1.0 Beta, May 29, 1996. Obtained via http://www.aiai.ed.ac.uk/WfMC/ 1996