Fachbeiträge

Konzeptuelle Objektsysteme zur Modellierung von Informations- und Steuerungssystemen

Tibor Németh

Institut für Angewandte Informatik und
Formale Beschreibungsverfahren
der Universität Karlsruhe (TH)
7500 Karlsruhe 1

Zusammenfassung

Konzeptuelle Modelle sind ein geeignetes Hilfsmittel, wenn für die Entwicklung komplexer rechnerunterstützter Anwendungen methodische Konzepte und Beschreibungsmittel benötigt werden, die eine weitgehend implementationsunabhängige Betrachtung des relevanten Systems bzw. Realweltausschnitts erlauben. Wenn auch zum Teil in der Literatur zu objektorientierter Programmierung die Notwendigkeit einer sorgfältigen Betrachtung des relevanten Realweltausschnitts in einer implementationsunabhängigen Form in Frage gestellt wird, so setzt sich doch vermehrt die Erkenntnis durch, daß dies nur bis zu einem gewissen Komplexitätsgrad der zukünftigen Anwendung möglich ist. In diesem Beitrag werden die Ideen eines konstruktiven Ansatzes für den Entwurf von Objektklassensystemen kurz skizziert. Dieser Ansatz basiert auf Ideen der konzeptuellen Modellierung mit Petri-Netzen.

1 Einleitung

Systeme zur Steuerung technischer Prozesse oder Informationssysteme zur Überwachung und Steuerung administrativer betrieblicher Abläufe machen einen Großteil der Anwendungen aus, die im allgemeinen nicht durch Standardlösungen abgedeckt werden können. Die Entwicklung solcher komplexen rechnerunterstützen Anwendungen kann nur dann in einer sinnvollen Weise erfolgen, wenn adäquate Entwicklungsmethoden und Vorgehensmodelle eingesetzt werden.

1.1 Aktuelles Entwurfsparadigma

In [Das91] wird ein Klassifikationsschema vorgeschlagen, das die im gesamten Bereich der Informatik bisher entstandenen Entwicklungsmethoden[1] und Vorgehensmodelle fünf Paradigmen zuordnet:

- Analysis-Synthesis-Evaluation Paradigm (ASE),
- Artificial Intelligence Paradigm,
- Algorithmic Paradigm,
- Formal Design Paradigm (FD),
- Theory of Plausible Designs (TPD).

Die Mehrzahl der für die Entwicklung von Informations- und Steuerungssystemen vorgeschlagenen Methoden und Vorgehensmodelle – in entsprechender Weise auch die dazugehörigen

[1] In [Das91] wird der Begriff „design" so verwendet, daß damit nicht nur Aspekte der Tätigkeiten, die, z.B. im Wasserfallmodell, mit Design – im Sinne von Enwurf – bezeichnet werden verbunden sind. Mit „design" wird eher der gesamte Entwicklungsprozeß referenziert.

Werkzeuge – lassen sich dem „Analysis-Synthesis-Evaluation Paradigm (ASE)" zuordnen. Methoden, die diesem Paradigma zugeordnet werden können, gehen von einer Menge von Anforderungen aus, die zunächst einer Analyse unterzogen werden. Daran schließen sich ein oder mehrere Syntheseschritte an. Abschließend folgt die Überprüfung des Designs an den Anforderungen. Damit ist nichts darüber ausgesagt, wie die einzelnen Schritte auszuführen sind oder welche Beziehungen jeweils paarweise zwischen Analyse, Synthese bzw. Evaluation bestehen.

Wenn auch der überwiegende Teil der in der Software- bzw Systementwicklung eingesetzten Methoden und Vorgehensmodellen diesem Parardigma zugeordnet werden können, so werden in [Das91] dennoch einige Kritikpunkte angeführt, die dieses Paradigma bzw. die darauf basierenden konkreten Methoden und Modelle zumindest von einer wissenschaftstheoretischen Betrachtung her grundsätzlich in Frage stellen. Das grundlegende Prinzip des ASE Paradigmas beruht auf dem Konzept der Induktion als Methode des wissenschaftlichen Erkenntnisprozesses. Durch entsprechende Anpassung dieser Methode läßt sich der Entwurfsprozeß als wissenschaftlicher Ansatz zur Systementwicklung darstellen. In der aktuellen wissenschaftstheoretischen Diskussion wird die Induktion jedoch als logisch nicht akzeptables Modell verworfen. Andererseits entspricht es jedoch einem psychologischen Bedürfnis, was sich in der weitverbreiteten Anwendung diese Prinzips in der Wissenschaft und im täglichen Leben zeigt.

Aber nicht nur vom wissenschaftstheoretischen Standpunkt läßt sich Kritik am ASE Paradigma anbringen. In [Das91] werden auch einige Schwachstellen beim praktischen Einsatz von Methoden und Modellen, die diesem Paradigma zuzuordnen sind, aufgezeigt. Als Hauptproblem wird die Annahme bezeichnet, daß eine strikte Trennung zwischen der Analyse der Anforderungen und der Synthese des Entwurfs möglich ist. Außerdem ist es in der realen Anwendung solcher Methoden und Modelle im allgemeinen nicht möglich, daß wirklich alle Anforderungen zu Beginn des Entwurfsprozesses identifiziert, gesammelt und analysiert werden können. Analyse und Synthese sind nicht in der strikten Form trennbar, wie es das ASE Paradigma verlangt.

1.2 Konzeptuelle Modelle

Zur Umgehung der beschriebenen Bedenken theoretischer und Einschränkungen praktischer Art wird in [Das91] als pragmatischer Lösungsansatz die Verwendung konzeptueller Modelle vorgeschlagen. Ein konzeptuelles Modell repräsentiert die Anforderungen und deren Wechselbeziehungen. Die Anforderungen werden dabei zunächst nur soweit analysiert, wie es notwendig ist um ein Modell aufbauen zu können. Dieses Modell wird nicht als (End-)Ergebnis der Analyse angesehen sondern gibt auch die ersten Ergebnisse der Synthese wieder. Dieses Modell ist dann die Kommunikationsgrundlage für Diskussionen mit dem Endanwender. In dieser Weise bildet es die Basis zur weiteren Verfeinerung und Ausarbeitung während der Synthese. Dabei werden im allgemeinen noch weitere Anforderungen erkannt oder frühere Anforderungen revidiert. Das Modell wird also laufend geändert und stellt keine starre Größe dar, die nur als Ganzes akzepziert oder vollständig verworfen werden kann.

Hier soll von einem bestimmten Vorgehensmodell oder einer Entwurfsmethode weitgehend abstrahiert werden. Um einen Bezugsrahmen herzustellen soll angenommen werden, daß die Entwicklung von rechnerunterstützten Informations- und Steuerungssystemen, unter anderen, die Phasen oder Tätigkeitsabschnitte *Anforderungsanalyse*, *Entwurf* und *Implementation* umfaßt, unabhängig davon wie diese Phasen im Lebenszyklus eines Software-Produkts angeordnet sind.[2]

Objektorientierte Ansätze für die Lösung komplexer Aufgabenstellungen in der Systementwicklung werden verstärkt erst seit Mitte der achtziger Jahre diskutiert, gewinnen aber zunehmend an Bedeutung. Ausgangspunkt der Arbeiten in diesem Bereich waren die Entwicklungen bei objektorientierten Programmiersprachen. Objektorientierte Ansätze für die Analyse, die Spezifi-

[2] Für eine detaillierte Beschreibung der einzelnen Phasen siehe [Som89, ScN90].

kation und den Entwurf von (Software-) Systemen bieten zunächst eine andere Sichtweise auf das betrachtete System als eher konventionelle funktions-, daten- oder datenfluß-orientierte Ansätze.[3]

Systeme lassen sich im allgemeinen durch die Menge ihrer möglichen Zustände und durch die Menge ihrer möglichen Zustandsübergänge beschreiben. Bei einer Systemrepräsentation durch eine Menge von Objektklassen müssen die globalen Systemzustände und Zustandsübergänge auf die lokalen Zustände und Zustandsübergänge der Objektklassen abgebildet werden. Die Anforderungen, die an solche Objektklassen gestellt werden, machen einen konstruktiven Entwurfsansatz notwendig.

Im folgenden wird ein Ansatz vorgestellt, der bestehende Methoden und Werkzeuge der konzeptuellen Modellierung verwendet, um die Vorteile, die eine objektorientierte Systemsicht bietet, effizient nutzen zu können. INCOME[4] ist ein in die CASE*-Umgebung[5] integriertes Methoden- und Werkzeugpaket für die Analyse und Spezifikation verteilter bzw. eingebetteter Systeme. Die in dieser Umgebung realisierten Methoden zur konzeptuellen Modellierung erlauben die Beschreibung statischer und dynamischer Systemaspekte auf der Grundlage der formalen Konzepte der Petri-Netze. Die von INCOME angebotenen Methoden und Werkzeuge werden benutzt, um eine objektorientierte Sicht auf das zu beschreibende System zu entwickeln. Für den objektorientierten Entwurf lassen sich verschiedene Sprachen verwenden. Besonders interessant für den hier betrachteten Ansatz ist jedoch der Einsatz der Sprache und der zugehörigen Entwicklungsumgebung Eiffel (siehe dazu [Mey88]). Mit Eiffel steht eine objektorientierte Sprache zur Verfügung, die sowohl für den Systementwurf als auch für die Implementierung geeignet ist. Der konstruktive Entwurf von Objektklassen wird durch Werkzeuge, die grafische Notationen verwenden, in effizienter Weise unterstützt. Die Integration der Methoden und Werkzeuge von INCOME, Eiffel und CASE* wird durch eine entsprechend modifizierte Repository-Struktur des CASE*-Dictionaries[6] vervollständigt.

2 Konzeptueller Entwurf von Objektsystemen

2.1 Objektorientierte Systemsicht

In der objektorientierten Sicht wird ein System als eine Menge von Objekten betrachtet. Objektklassen sind Schemata, die die Struktur und das Verhalten von Instanzen dieser Klassen – den Objekten – beschreiben.[7] Als Objekte werden Einheiten bezeichnet, die lokal Zustände in ihren Strukturen repräsentieren und mögliche Übergänge zwischen diesen Zuständen, das heißt beobachtbares Verhalten realisieren. Zwischen den Objekten können Beziehungen bestehen, die beim Entwurf der Klassen festgelegt werden. Mögliche Beziehungen ergeben sich dadurch, daß Objekte Eigenschaften von anderen Objekten erben können, oder daß Objekte bestimmte Eigenschaften anderer Objekte benutzen. Eine weitere wichtige Beziehungsart, die bei der Entwicklung der Objektklassen zu berücksichtigen ist, ist die sogenannte Aggregation, bei der ein Objekt aus mehreren untergeordneten Objekten besteht.

---

[3] Eine solche Einteilung der primären Sichtweisen auf die betrachteten Systeme findet sich z. B. in [CoY90].

[4] Die grundlegenden Konzepte von INCOME wurden am Institut für Angewandte Informatik und Formale Beschreibungsverfahren der Universität Karlsruhe entwickelt. Die weitere Entwicklung dieser Konzepte, die Integration in die CASE*-Umgebung und der Vertrieb des Methoden- und Werkzeugspakets INCOME erfolgen durch die PROMATIS Informatik, Straubenhardt bei Pforzheim.

[5] CASE* ist ein Produkt der ORACLE Corp., Belmont, USA.

[6] CASE*-Dictionary ist ein Produkt der ORACLE Corp., Belmont, USA.

[7] Unter Objektklassen soll die Beschreibung konkreter Objekte verstanden werden. Im folgenden wird keine scharfe Trennung bei der Verwendung der Begriffe Objekt und Klasse vorgenommen, wenn sich aus dem Kontext die genaue Bedeutung ergibt.

2.2 Konzeptuelle Modellierung

Bei der Entwicklung von Informations- und Steuerungssystemen wird der konzeptuellen Modellierung eine zentrale Rolle zugewiesen (siehe dazu z.B. [BMS84, ISO82, ScS83]). Dies gilt insbesondere für solche Systeme, die unter Verwendung von Datenbanksystemen realisiert werden. Ein konzeptuelles Modell eines Systems ist eine erste formale Beschreibung des Systems und der für dieses System relevanten Umgebung. Diese erste formale Darstellung soll jedoch von Aspekten der tatsächlichen Realisierung abstrahieren. Solche Aspekte sollen erst bei Entwurfs- und Implementationsaktivitäten berücksichtigt werden. Ein konzeptuelles Modell beschreibt die relevanten Einheiten des Systems (und teilweise auch seiner Umgebung) sowie deren strukturelle Beziehungen untereinander. Außerdem wird das Verhalten dieser Einheiten und des Systems als Gesamtheit dargestellt. Ein konzeptuelles Modell soll alle statischen und dynamischen Eigenschaften des betrachteten Realweltausschnitts vollständig erfassen (siehe [ISO82]). Es soll aber auch nur die konzeptuell relevanten Aspekte umfassen. Abbildung 1 soll die Einordnung der im konzeptueller Modell erfaßten Aspekte als problemorientierte Beschreibung verdeutlichen.

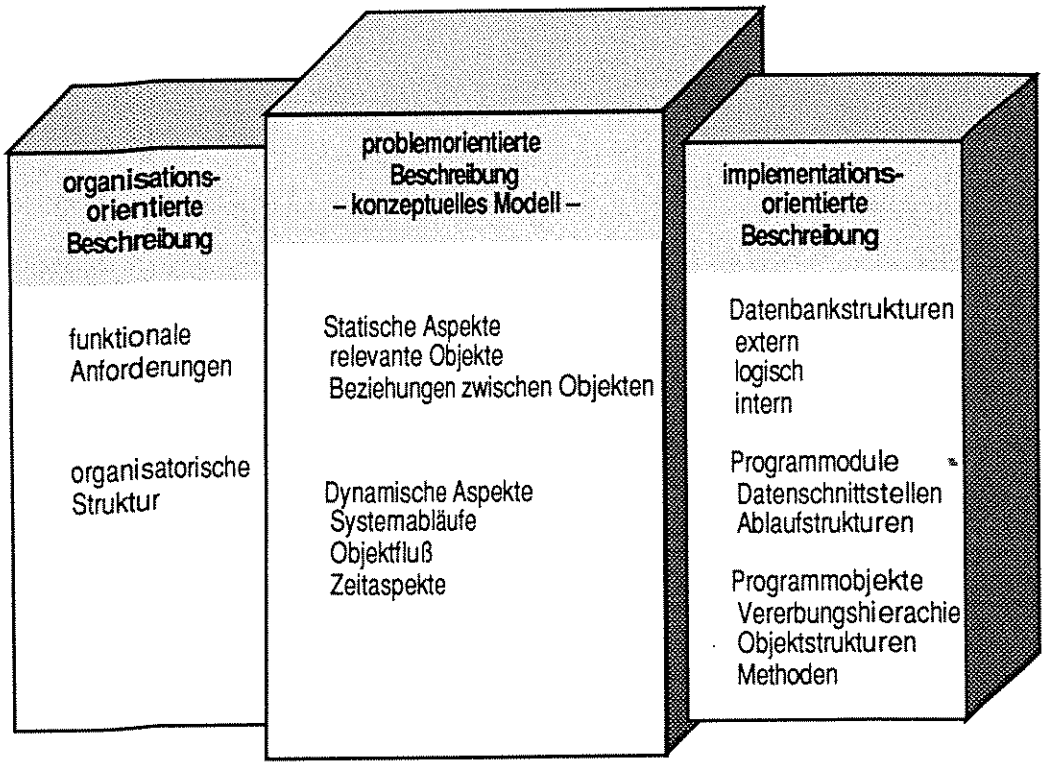

Abb. 1 Einordnung konzeptueller Modelle

2.3 Entwurf von Objektsystemen

Ziel objektorientierter Entwicklungsansätze ist der Aufbau eines Systems von Objekten, das den relevanten Realweltausschnitt für Entwicklungsaufgaben adäquat wiedergibt. Unter einem System soll hier in Anlehnung an [Dae88] eine Zusammenfassung von miteinander in Beziehung stehenden Teilen verstanden werden. Die Gesamtheit der Teile bilden unter bestimmten anzugebenden Aspekten ein in sich abgeschlossenes Ganzes. Ergebnis einer objektorientierten Systementwicklung ist also eine Menge von Objekten, die in bestimmten Beziehungen zueinander stehen. In einem solchen Objektsystem muß der strukturelle Aufbau der Objekte und ihr jeweiliges Verhalten innerhalb des Systems beschrieben werden. Außerdem sind die Beziehungen zwischen den Objekten festzulegen.

Die Entwicklung eines solchen Objektsystems wird in der Literatur zum Teil als trivialer Prozeß angesehen. Exemplarisch ist das folgende Zitat aus [Mey88]: "... object-oriented design is a natural approach: the world being modeled is made of objects ... This is why object-oriented designers usually do not spend their time in academic discussions of methods to find objects: in the physical or abstract reality being modeled, the objects are just there for the picking"! Es hat sich jedoch gezeigt, daß dieses Vorgehen nur für eine sehr eingeschränkte Klasse von Anwendungen möglich ist. So findet sich in [CoY90] folgende Antwort auf obiges Zitat: "Nuts! ..., identifying Objects was never intuitively obvious."

Wenn sich die zur Beschreibung des Realweltausschnitts benötigten Objekte nicht unmittelbar aus Realweltobjekten ableiten lassen, müssen entsprechende zusätzliche Beschreibungsmittel und Verfahren eingesetzt werden. Da, wie oben bereits angedeutet, bei der Betrachtung von Realweltausschnitten, ebenso wie bei Objekten, sowohl statische als auch dynamische Aspekte zu berücksichtigen sind, werden für diese Aspekte adäquate Beschreibungsmittel benötigt. Zunächst sollte eine solche Beschreibung auf einer konzeptuellen Ebene beginnen, d.h. es wird eine erste formale Beschreibung entwickelt, die jedoch noch so weit wie möglich von Aspekten der späteren Implementation abstrahiert. Aufgabe dieser Beschreibung ist die Abstraktion von unwesentlichen Aspekten des betrachteten Realweltausschnitts. Im Zusammenhang mit einer objektorientierten Systementwicklung sollte diese konzeptuelle Beschreibung die Identifikation und den Entwurf der relevanten Objekte, deren Aufbau und Verhalten, sowie der zwischen den Objekten bestehenden Beziehungen unterstützen.

2.4 Modellierung statischer Eigenschaften

Für die Spezifikation statischer Eigenschaften auf einer konzeptuellen Ebene haben sich semantische Datenmodelle bewährt.[8] Besonders interessant in diesem Zusammenhang sind zwei Modelle. Das eine ist das semantisch-hierarchische Objektmodell (SHO), das relationale Konzepte mit vier wichtigen Beziehungen semantischer Netze kombiniert.[9] Als Abstraktionsmechanismen unterstützt das SHO die Klassifikation (instance-of relationship), die Generalisierung (is-a relationship), die Aggregation (part-of relationship) und die Gruppierung (association, member-of relationship). Im Gegensatz zu anderen Datenmodellen werden im SHO Attribute als Objekte behandelt, so daß kein Unterschied zwischen Attributen und Objekten gemacht wird.

Das zweite Datenmodell ist das Entity-Relationship-Modell (ERM), das mit den unterschiedlichsten Erweiterungen heute das wohl am häufigsten eingesetzte konzeptuelle Datenmodell ist.[10] Auf eine ausführliche Darstellung dieser beiden Modelle soll hier verzichtet werden. Nähere Informationen zu den Modellen und den Unterschieden zwischen ihnen sind z.B. in [ScS83, ScN90] zu finden.

2.5 Modellierung von Verhaltensaspekten

Dynamische Aspekte, die das Verhalten von Systemen bzw. Objekten betreffen, lassen sich in geeigneter Weise mit Hilfe von Petri-Netzen spezifizieren. Neben der Möglichkeit der graphischen Darstellung des beschriebenen Sachverhalts, bieten Petri-Netze eine große Ausdrucksmächtigkeit und umfangreiche Möglichkeiten zur Qualitätssicherung. Neben Syntaxanalysen können spezielle formale Analysen durchgeführt werden, mit denen etwa isolierte Prozesse oder Verklemmungen erkannt werden können. Darüber hinaus sind Simulationen des zu entwickelnden Systems möglich, die ein ideales Mittel zur Visualsierung und Überprüfung komplexer dynamischer Zusammenhänge darstellen.[11] Für den hier vorgestellten Ansatz wird eine

---

[8] Für eine Übersicht über solche Datenmodelle siehe z.B. [HuK87]

[9] Grundlagen dieses Modells sind in [SmS77] und [BrR84] dargestellt.

[10] Die grundlegenden Konzepte dieses Modells sind in [Che76] beschrieben.

[11] Übersichten solcher Werkzeuge finden sich z.B. in [Fel90].

höhere Form der Petri-Netze – Prädikat/Transitions-Netze (Pr/T-Netze) – eingesetzt.[12] Diese Netze bieten eine größere Ausdrucksmächtigkeit als einfache Petri-Netz Typen, was zu einer wesentlichen Reduzierung des graphischen Modellierungsaufwands führt.

Abbildung 2 zeigt ein Beispiel eines Pr/T-Netzes. Aktive Einheiten eines Objekts bzw. eines Systems werden als Transitionen (graphisch als Rechtecke) und passive Komponenten als Prädikatstellen (graphisch als Kreise) dargestellt. Transitionen können in Abhängigkeit gewisser Bedingungen schalten, d.h. eine Zustandsänderung durchführen. Zustände werden durch Marken in den Prädikatstellen modelliert. Wenn eine Transition schaltet, dann werden aus ihren Eingangsstellen Marken entnommen und in ihren Ausgangsstellen Marken abgelegt. Eine Transition kann schalten, wenn sie aktiviert ist. Sie ist aktiviert, wenn in den Eingangsstellen genügend Marken vorhanden sind, und die Kapazität ihrer Ausgangsstellen zur Aufnahme von Marken ausreicht. In Pr/T-Netzen werden Marken als konkrete Objekte betrachtet. Zusätzliche Bedingungen an die beim Schalten verwendeten bzw. erzeugten Marken lassen sich über einen logischen Ausdruck als Transitionsinschrift festlegen.

Als Erweiterung der Modellierungsmöglichkeiten von Pr/T-Netzen sind deklarative Beschreibungskonzepte für Ausnahmebedingungen vorgesehen. Ausnahmebedingungen, die sich auf einzelne Zustände oder auf Zustandsübergänge beziehen, können durch sogenannte Fakt-Transitionen bzw. durch ausgeschlossenen Transitionen formuliert werden (siehe dazu [Obe90]). In Abbildung 2 ist als Beispiel einer Zustandsrestriktion die Ausnahmebedingung *unterschreiten Mindestkontostand* als Fakt-Transition modelliert. Über die Transitionsinschrift wird der Zustand ausgeschlossen, daß durch eine Auszahlung der Mindestkontostand unterschritten wird.

2.6 Beipiel

An einem Beispiel sollen im folgenden kurz die wesentlichen Merkmale der Beschreibung des Verhaltens von Objekten auf einer konzeptuellen Ebene erläutert werden. Abbildung 2 stellt die Beschreibung einer Objektklasse *Konto* dar.

Der strukturelle Aufbau der Objekte kann z.B. mit Hilfe des SHO dargestellt. Objekte der Klasse *Konto* sind über eine Aggregationsbeziehung mit den Komponenten *Inhaber*, *MinKntstand* und *Kntstand* verbunden. Komponenten einer Objektklasse können entweder elementar sein, so daß sich für sie ein Basiswertebereich angeben läßt oder es können Objekte sein, für die wieder eigene Beschreibungen existieren. Verbindungen zu diesen Objekten werden über Generalisierungsbeziehungen oder über Aggregationsbeziehungen dargestellt. Im vorliegenden Beispiel ist die Komponenten *Inhaber* über eine Generalisierungsbeziehung mit einer anderen Objektklassen – der Objektklasse *Person* – verbunden, von der sie Eigenschaften erbt.

Für die Beschreibung des Verhalten der Objekte werden die oben angeführten Pr/T-Netze eingesetzt. Für Objekte der Klasse *Konto* sind drei mögliche Operationen vorgesehen. Eröffnen eines Kontos, Einzahlungen auf ein Konto vornehmen, Auszahlung vom Konto vornehmen. Jede dieser Operationen überführt ein Objekt der Klasse Konto in einen neuen Zustand, der sich durch eine geänderte Markierung der Prädikatstellen ausdrückt. Die drei Operationen können nur dann ausgeführt werden, wenn in den die Schnittstelle des Objekts bildenden Prädikatstellen *Person* und *Umsatz* und in der Prädikatstelle *Konto* solche Objekte vorliegen, daß eine der Transitionen aktiviert ist.

---

[12] Eine Einführung in die Konzepte der Pr/T-Netze findet sich in [Gen87]. Zu den im INCOME Werkzeug- und Methodenpaket eingesetzten Pr/T-Netzen siehe [SNS90, Sch90a].

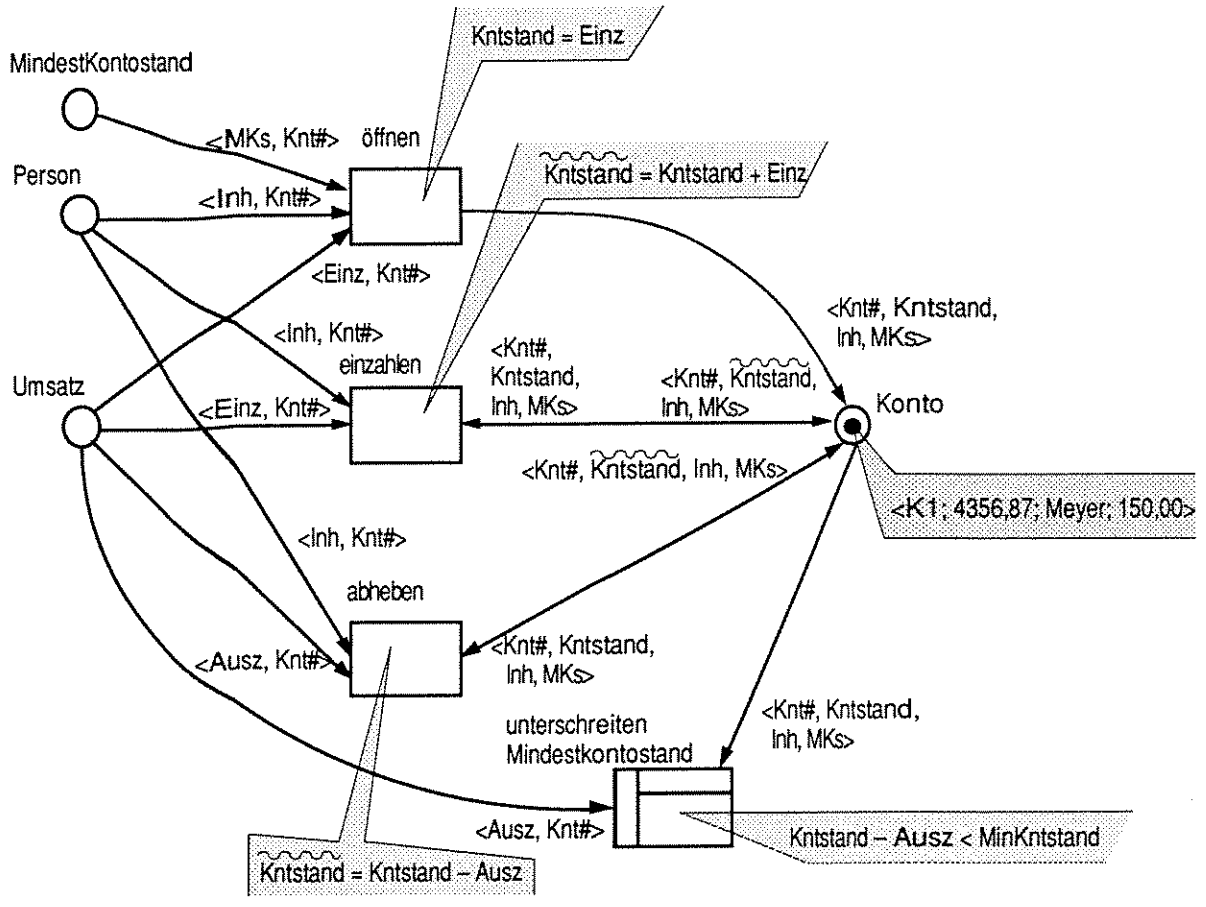

Abb. 2 Darstellung des Zustands eines Objekts der Klasse *Konto*

Für die Spezifikation von Objektsystemen genügt es jedoch nicht, nur die einzelnen Objektklassen zu beschreiben, auch die Beziehungen zwischen den Objekten müssen dargestellt werden. Beziehungen statischer Art können z.B. über Generalisierungen gegeben sein. Ebenso lassen sich auf der dynamischen Seite Beziehungen identifizieren. Dazu zählen etwa Aspekte, die sich aus Verwendungsbeziehungen von Objekten untereinander ergeben. So muß unter anderem berücksichtigt werden, ob bestimmte Aktionen synchron oder asynchron ablaufen können, ob bestimmte zeitliche Restriktionen absoluter oder relativer Art bestehen.

Solche Beziehungen dynamischer Art werden durch spezielle Objekte abgebildet, für deren Darstellung wiederum Pr/T-Netze benutzt werden. Zur Abbildung eines großen Teils dieser Beziehungen können Netze mit einer vorgegebenen Struktur benutzt werden. Die vorgegebenen Strukturen müssen dann nur noch an die jeweilige Semantik der beteiligten Objekte angepaßt werden.

3 Entwicklung von Objektsystemen

Nachdem im vorangegangenen Abschnitt kurz die für die Entwicklung von Objektsystemen einsetzbaren Beschreibungsmittel vorgestellt wurden, soll jetzt auf die Verwendung dieser Beschreibungsmittel im Entwicklungsprozeß eingegangen werden. Wie in der Einleitung angedeu-

tet wurde, soll hier von einem speziellen Vorgehensmodell für die Software-Entwicklung abstrahiert werden. Betrachtet werden muß jedoch, wie für die interessierenden Tätigkeitsabschnitte Anforderungsanalyse und Entwurf ein Objektsystem entwickelt werden kann.

Für die Entwicklung von Objektsystemen lassen sich typische Arbeitsschritte identifizieren, die im allgemeinen zyklisch mehrfach durchlaufen werden. Zunächst werden statische und dynamische Eigenschaften des betrachteten Realweltausschnitts analysiert und im Rahmen der konzeptuellen Modellierung spezifiziert. Mit Hilfe formaler Analysen und der Simulation bzw. Prototyping des entstandenen Modells wird es einer Validierung unterzogen. Dabei wird das Modell auf Korrektheit hin überprüft. Zudem erhält man Hinweise auf noch nicht berücksichtigte Aspekte des soweit entwickelten Anforderungsmodells. Durch die Verwendung formaler Beschreibungsmittel zur Spezifikation dynamischer und statischer Anforderungen – Pr/T-Netze und semantische Datenmodelle – können die einzelnen Schritte effektiv durch Rechnereinsatz unterstützt werden.

Das Arbeiten mit dem Modell im Validierungsschritt ist auch die Grundlage für die Spezifikation des Objektsystems. Zur Entwicklung eines solchen Systems ist es zunächst einmal notwendig, die relevanten Objektklassen aus der Spezifikation der Systemanforderungen abzuleiten. Als nächstes muß der strukturelle Aufbau der Objektklassen und ihr Verhalten spezifiziert werden. Dabei kann sich der Entwickler einer Reihe rechnergestützter Werkzeuge bedienen. So lassen sich aus dem statischen Teil des konzeptuellen Modells Sichten mit unterschiedlich tiefen Hierarchiestufungen ableiten. Diese Sichten auf das konzeptuelle Modell bilden von der statischen Seite her einen Anhaltspunkt für die Identifikation der Klassen des Objektsystems und ihrer Beziehungen untereinander.

Der Aufbau eines Objektsystems kann jedoch nicht nur die strukturellen Gesichtspunkte berücksichtigen. Ebenso muß berücksichtigt werden, wie das globale Systemverhalten, das im dynamischen Teil des konzeptuellen Modells beschrieben ist, auf das lokale Verhalten der einzelnen Objekte abgebildet wird. Hier werden wiederum durch entsprechende Werkzeugunterstützung Ausschnitte des Verhaltensmodells gebildet. Besonders hilfreich hierfür sind die Ergebnisse der im vorhergehenden Validierungsschritt durchgeführten Simulation des Modells.

4 CASE-Umgebung für die Entwicklung von Objektsystemen

Der hier vorgestellte Ansatz zum konstruktiven Entwurf von Objektklassen wird in eine umfassende CASE-Umgebung integriert. INCOME ist eine Erweiterung der CASE*-Umgebung der ORACLE Corp. Sie ergänzt das Vorgehensmodell CASE*Method und die Werkzeuge der Umgebung um eine Methode und Werkzeuge zur Modellierung von Systemverhalten. Mit dieser Erweiterung steht eine Entwicklungsumgebung für ein weites Spektrum an Anwendungen zur Verfügung, die einerseits durch intensiven Datenbankeinsatz anderseits aber auch durch komplexe Ablauforganisationen und Real-Time-Aspekte geprägt sind. Um die Vorteile, die eine objektorientierte Systemsicht bietet und um neue objektorientierte Techniken bei der Implementation (Datenbanksysteme, Programmiersprachen) einsetzen zu können, wird INCOME bzw. CASE* entsprechend erweitert.

INCOME/Dictionary fügt den im CASE*Dictionary (Repository) vordefinierten Datenbankstrukturen Strukturen zur Verwaltung der bei der Arbeit mit den INCOME Werkzeugen anfallenden Informationen hinzu. Diese Informationen sind über eine Vielzahl von Verbindungen mit den Informationen des CASE*Dictionaries verbunden. In analoger Weise wird die Repository-Struktur für die Entwicklung von Objektsystemen aufgebaut und in das bestehende System eingefügt.

4.1 Unterstützung der objekt-orientierten Programmierung

Zur Unterstützung der Tätigkeiten im Rahmen der Entwicklungsphase Implementation bieten CASE* und INCOME eine Reihe von Werkzeugen, wie z.B. Generatoren an. Für eine objekt-

orientierte Erweiterung lassen sich verschiedene objektorientierte Programmiersprachen einsetzen. Grundsätzlich muß dabei unterschieden werden ob eine klassenbasierte oder eine „Actor"-Sprache verwendet wird. Zu den objektorientierten Programmiersprachen, die ein Klassenkonzept zugrundelegen gehören z.B. Smalltalk, Trellis/Owl, Eiffel, C++, Loops und Flavors. Beispiele für Actor-Sprachen, die kein Klassenkonzept besitzen, sind Act1, Delegation und Self. Im Gegensatz zu den klassenbasierten Sprachen werden in Actor-Sprachen Objekte auf der Instanzenebene beschrieben. Jedes Objekt kann als Prototyp angesehen werden, das seinen eigenen Typ definiert. Dieser Unterschied muß bei der Verwendung von Pr/T-Netzen zur Beschreibung des Verhaltens berücksichtigt werden. Ein Netz kann sowohl als Beschreibung auf der Klassenebene als auch auf der Instanzenebene eingesetzt werden.

Hier soll jetzt nur auf die Unterstützung klassenbasierter Sprachen eingegangen werden. Beschreibt ein Netz Objektklassen, so repräsentieren die Marken, die sich im Netz befinden, den Zustand konkreter Objekte. Diese Marken müssen daher konkreten Objekten zuordenbar sein. Der Zustand eines Objekts kann sich auch aus der Markierung mehrerer Prädikatstellen ergeben. Um eine Zuordnung der einzelnen Markierungen zu konkreten Objekten zu erreichen, erhalten die Marken zusätzlich die Identifikation der Objekte. In den hier betrachteten Programmiersprachen wird üblicherweise durch Variablennamen eine Referenz auf Objekte ermöglicht. Dieses Konzept kann auch auf die Wertekombinationen der Netzmarkierungen angewandt werden. Die Markierung in der Stelle *Konto* der Abb. 2 zeigt den Zustand eines konkreten Objekts, das über den Variablennamen *K1* referenziert werden kann.

Eine sinnvolle Anbindung der vorgestellten Konzepte ist mit Eiffel möglich. Besonders solche entwurfsorientierten Sprachkonstrukte wie *deferred classes*, oder Möglichkeiten zur Definition von Pre- und Post-Conditions bzw. Invarianten und die dazugehörigen Mechanismen des Exception handlings erleichtern die Umsetzung konzeptueller Beschreibungen für Implementationszwecke.

Die Eiffel-Entwicklungsumgebung bietet mit dem graphischen Browser *good* eine erste Unterstützung für das Auffinden und Wiederverwenden bereits vorhandener Objekte. Dieses Werkzeug gibt jedoch nur Beziehungen statischer Art zwischen Objekten wieder. Für den Aufbau von komplex strukturierten Objektsystemen läßt sich das Verhalten des entstehenden Gesamtsystems nicht mehr unmittelbar aus dem Verhalten der einzelnen Objekte erkennen. Eine Erweiterung der Eiffel-Entwicklungsumgebung um Werkzeuge zur Spezifikation, Analyse und Simulation des Verhaltens von Objekten bzw. Objektsystemen unterstützt insbesondere die Entwicklung komplexer Aufgabenstellung technischer Anwendungsbereiche. Dazu zählen etwa Automatisierungslösungen in der Fertigungs- und Verfahrenstechnik.[13]

Die Einbindung eines Ansatzes zum konstruktiven Entwurf von Objektklassen in der beschriebenen Art und Weise in eine auf dem Markt verfügbare und direkt einsetzbare Entwicklungsumgebung erlaubt die Anwendung, Überprüfung und Weiterentwicklung der Konzepte an realen Aufgabenstellungen in Zusammenarbeit mit Partnern aus der betrieblichen Praxis.

5 Literatur

[BMS84] M.L. Brodie, J. Mylopoulos und J.W. Schmidt, Eds.: *On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and Programming Languages.* Springer-Verlag, New York, 1984.

[Boe76] B.W. Boehm: Software engineering. *IEEE Trans. Comp. 25*, 12 (Dec. 1976), 1226-1241.

[Boe88] B.W. Boehm: A spiral model of software development and enhancement. *Computer 21*, 5 (May 1988), 61-72.

[13] Siehe dazu auch [Sch90b].

[BrR84] M.L. Brodie und D. Ridjanovic: On the design and specification of database transactions. In *On Conceptual Modelling. Perspectives from Artifical Intelligence, Databases, and Programming Languages*, M.L. Brodie, J. Mylopoulos und J.W. Schmidt, Eds. Springer-Verlag, New York, 1984.

[Che76] P.P. Chen: The entity-relationship model – Toward a unified view of data. *ACM Trans. Database Syst. 1*, 1 (Mar. 1976), 9-36.

[CoY90] P. Coad and E. Yourdan: *Object-Oriented Analysis*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1990.

[Dae88] W.F. Daenzer (Hrsg.): *Systems Engineering. Leitfaden zur methodischen Durchführung umfangreicher Planungsvorhaben. 6. Auflage*. Verlag Industrielle Organisation, Zürich, 1988.

[Das91] S. Dasgupta: *Design Theory and Computer Science*. Cambridge University Press, Cambridge, 1991.

[Fel90] F. Feldbrugge: Petri net tool overview 1989. In *Advances in Petri Nets 1989, Part I. LNCS 424*, G. Rozenberg, Hrsg. Springer-Verlag, Berlin, Heidelberg, 1990.

[Gen87] H.J. Genrich: Predicate/transition nets. In *Petri Nets: Central Models and Their Properties. Advances in Petri Nets 1986, Part I. LNCS 254*, W. Brauer, W. Reisig und G. Rozenberg, Hrsg. Springer-Verlag, Berlin, Heidelberg, 1987.

[HuK87] R. Hull und R. King: Semantic database modelling: survey, applications, and research issues. *ACM Computing Surveys 19*, 3 (Sep. 1987), 201-260.

[ISO82] J.J. Griethuysen, Ed.: *Concepts and Terminology for the Conceptual Schema and the Information Base*, Report of the ISO/TC97/SC5/WG3, Publ. No. ISO/TC97/SC5-N695, 1982.

[Mey88] B. Meyer: *Object-Oriented Software Construction*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[Obe90] A. Oberweis: Petri-Netz-Beschreibungstechniken für Exception-Handling-Mechanismen in der Automatisierungstechnik. Forschungsbericht 207, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe (TH), November 1990 (erscheint in *at – Automatisierungstechnik*)

[Roy70] W.W. Royce: Managing the development of large software systems: concepts and techniques. *Proceedings, WESCON* (Aug. 1970).

[Sch90a] F. Schönthaler: CASE-Tools für verteilte Systeme: Petri-Netze in der Praxis. In *Tool 90. Proceedings. Karlsruhe, November 1990.*, W. Zorn, Hrsg. MESAGO, Messe & Kongreß GmbH, Stuttgart, 1990.

[Sch90b] F. Schönthaler: INCOME und CASE*: Ein Anwendungsentwicklungssystem für die Produktionsautomatisierung. In *DOAG: 3. Deutsche ORACLE-Anwenderkonferenz. Proceedings. Fellbach, November 1990.*

[ScN90] F. Schönthaler und T. Németh: *Software-Entwicklungswerkzeuge: Methodische Grundlagen*. B.G. Teubner, Stuttgart, 1990.

[ScS83] G. Schlageter und W. Stucky: *Datenbanksysteme: Konzepte und Modelle, 2. Auflage*. B.G. Teubner, Stuttgart, 1983.

[SmS77] J.M. Smith und D.C.P. Smith: Database abstractions: aggregation and generalization. *ACM Trans. Database Syst. 2*, 2 (June 1977), 105-133.

[SNS90] W. Stucky, T. Németh und F. Schönthaler: Modellierung und Simulation verteilter Systeme mit INCOME. In *GI – 20. Jahrestagung. Informatik auf dem Weg zum Anwender. Stuttgart, Oktober 1990. Informatik-Fachbericht*, A. Reuter, Hrsg. Springer-Verlag, Berlin, Heidelberg, 1990.

[Som89] I. Sommerville: *Software Engineering, 3rd Edition*. Addison-Wesley Publ. Comp., Wokingham, England, 1989.