Ansatz zur Kombination von objektorientierten Methoden für die Fachkonzeptphase

Barbara Röllnreiter
Softlab GmbH
Zamdorferstraße 120
81677 München
rol@softlab.de

Ralph Penno
Softlab GmbH
Zamdorferstraße 120
81677 München
pen@softlab.de

0 Zusammenfassung

Objektorientierte Projekte werden gemäß Kundenwunsch meist nach einer bestimmten Methode (Booch, Rumbaugh, etc.) durchgeführt. Um zu vermeiden, daß bei jedem Projekt erneut endlose Diskussionen um Vorgehen und Ergebnisse geführt werden, haben wir Vorgaben entwickelt, die die gängigsten OO-Methoden einbeziehen und, da die Methoden für bestimmte Fragestellungen Lücken aufweisen, Bestandteile der Methoden miteinander kombiniert. Vorliegender Aufsatz ist der Ausschnitt der Vorgaben, der sich mit dem ersten Schritt im Projekt, der objektorientierten Fachkonzeptphase, beschäftigt. Die Ergebnisse dieser Phase sind die Grundlagen zum Aufbau einer Projektdatenbank, die über Werkzeuge eine Generierung von Ergebnissen nachfolgender Phasen ermöglicht.

1 Einleitung

Steht ein Analytiker in einem Projekt vor der Aufgabe, ein Fachkonzept zu schreiben, so trifft er bei den Ansprechpartnern auf eine Vielzahl von Methoden und Verfahren. Die Fachseite hat häufig andere Vorstellungen von der Form eines Fachkonzepts und dem Weg, zu einem brauchbaren Ergebnis zu kommen als die DV-Abteilung. Softwarehäuser, die als externe Partner hinzugezogen werden, bevorzugen wieder andere, oft auch eigene Methoden für diese Projektphase.

Bei der Umsetzung des Fachkonzepts in ein DV-technisches Modell werden immer wieder Transformationsfehler begangen. Um einen integrierten Ansatz zu erhalten, ist zu klären, **welche Methoden und Ergebnisse** benötigt werden und **wie** diese zusammenspielen. Es ist eine sinnvolle und praktikable Mischung aus Fachkonzeption, Funktionsmodellierung, Datenmodellierung, objekt-orientierter Analyse, Use Cases, Geschäftsprozeßmodellierung, GUI-Prototyping, Workflow-Prototyping, etc. herauszufiltern.

Sind die Methoden, Ergebnisse und deren Zusammenspiel identifiziert, dann steht ein Softwareersteller vor der Aufgabe, wie die verschiedenen Methoden und Verfahren der Fach- und DV-Abteilung in die eigene Entwicklungsumgebung integriert werden können. Hier gilt es zu beachten, daß die Entwicklungsumgebung nur tragbar ist, wenn sie für verschiedene Kundenprojekte genutzt werden kann. Dies bedeutet, daß solch eine EU offen für die derzeit gängigsten OO-Methoden sein muß.

Dieser Beitrag zeigt einen Ansatz, welche Methoden wie zusammenspielen können und wie diese Komponenten in eine weitgehend unabhängige Entwicklungsumgebung integriert werden können.

2 Einbindung in den SW-Herstellungsprozeß

In der Fachkonzeptphase findet oft der erste Kontakt der DV-Seite mit den Inhalten und Vorgaben eines SW-Projekts statt. Die Sicht des Kunden muß verstanden werden und die **wesentlichen** Punkte müssen im Anforderungsmodell festgehalten werden als Basis für die Transformation in ein DV-technisches Modell.

Das Vorgehen und die Ergebnisse des vorliegenden, integrierten Ansatzes helfen, der Lösung des Problems, nämlich herauszufinden, **was wesentlich ist**, näherzukommen.

Die ersten Ergebnisse auf dem Weg zum Anforderungsmodell sind noch skizzenhaft, werden aber nach und nach ausgebaut und mit einem Analysewerkzeug (z.B. Paradigm Plus, Rational Rose) vervollständigt. Die Ergebnisse des Analysewerkzeugs werden über Reports und Scanner als erste Daten (C++-Klassen) im Projektrepository abgelegt und können u.a. zur Generierung von Dokumenten der Designphase benutzt werden.

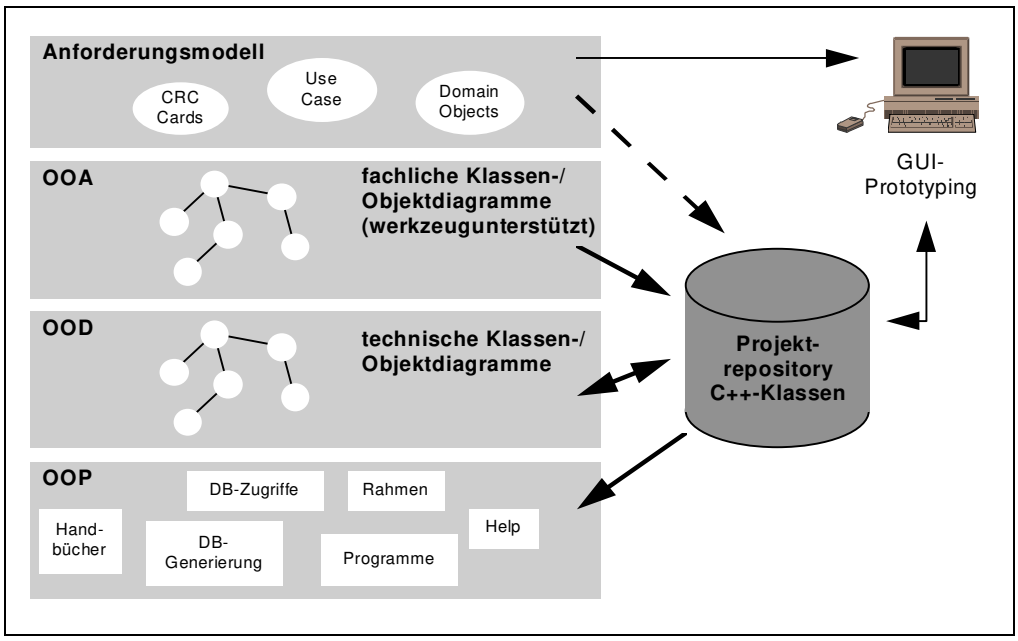

Abbildung 1: Einbindung in den SW-Herstellungsprozeß

3 Vom Anwendertext zum Analysemodell

Idealerweise findet beim Kunden vor Aufsetzen eines Projekts eine Analyse des Ist-Zustandes des zu behandelnden Gesamtprozesses statt. Im Anschluß an die Ist-Analyse wird ein Soll-Konzept erarbeitet, das Grundlage für Geschäftsprozeßmodell und Anforderungsmodell wird.

Ansonsten wird vom Kunden eine Ausschreibung oder ein Pflichtenheft verfasst, das die erste Information über die zu erstellende DV-Lösung verkörpert. Im folgenden werden Soll-Konzept, Pflichtenheft oder Ausschreibung einheitlich als Anwendertext bezeichnet. Der Anwendertext ist laufend durch Interviews mit dem Kunden und durch evtl. zusätzlich notwendige Dokumente zu ergänzen.

Die Abbildung zeigt, wie die Transformation des Anwendertextes in ein DV-technisches Analysemodell methodisch unterstützt werden kann:

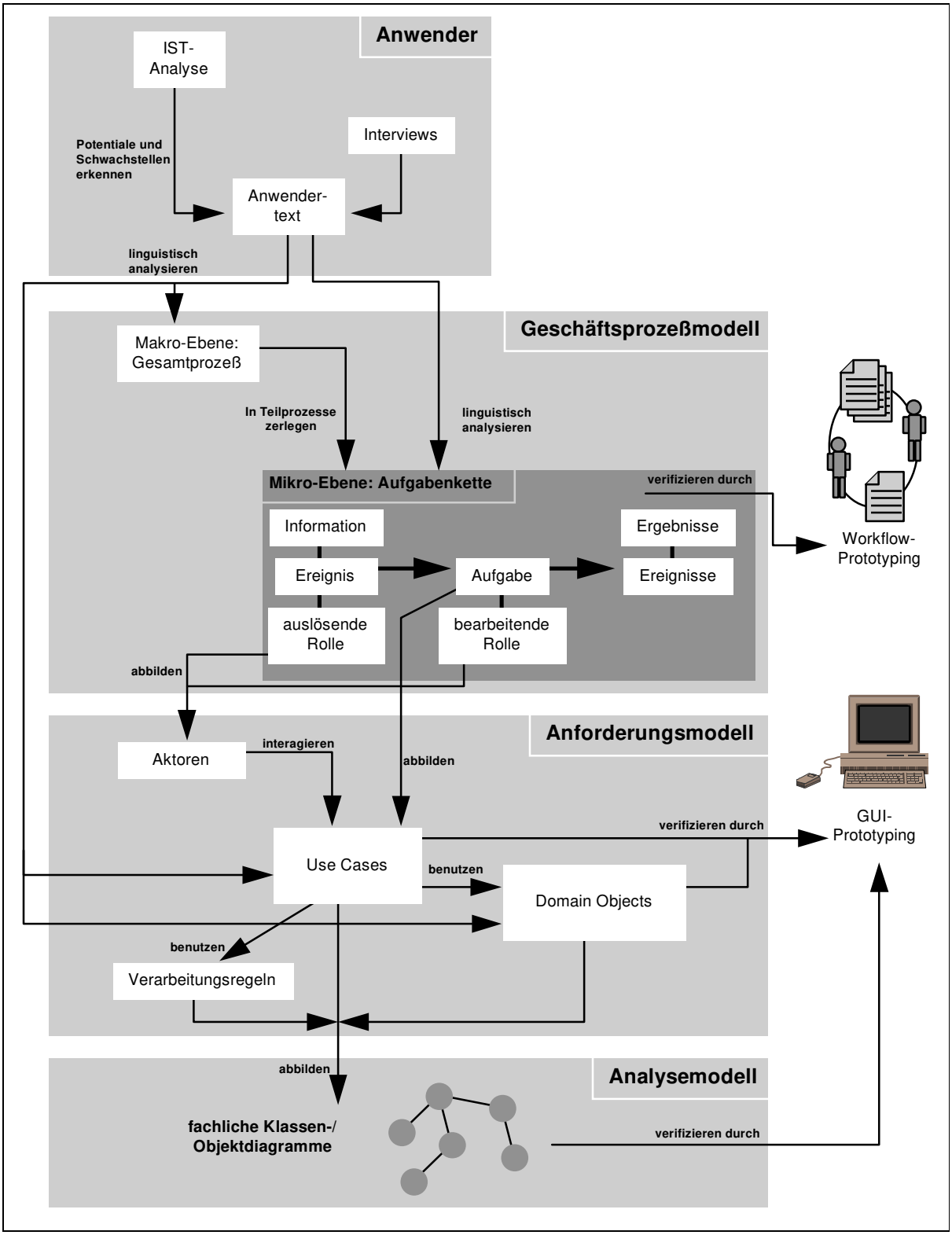

Abbildung 2: Vom Anwendertext zum Analysemodell

3.1 Vom Anwendertext zum Geschäftsprozeßmodell

Der Anwendertext enthält, als Ganzes gesehen, den zu behandelnden Gesamtprozeß. Der Gesamtprozeß läßt sich in Teilprozesse zerlegen, indem der Anwendertext in einzelne Aufgaben zerlegt wird. Einzelne Aufgaben müssen nacheinander ausgeführt werden in einer Aufgabenkette. Die Aufgabenketten lassen sich durch Aussagen über die Verknüpfung bestimmter Textabschnitte oder erkannter Aufgaben aus dem Anwendertext ableiten. Häufig wird im Anwendertext auch beschrieben, wer die jeweilige Aufgabe ausführt oder ausführen darf. Hieraus ergeben sich die Rollen des Geschäftsprozeßmodells.

Durch Workflow-Prototyping verifizierte Aufgabenketten helfen, Mißverständnisse zwischen Anwender und Softwareersteller in Bezug auf die künftige Anwendung zu vermeiden.

3.2 Vom Anwendertext und Geschäftsprozeßmodell zum Anforderungsmodell

Aus den per DV zu lösenden Aufgaben des Geschäftsprozeßmodells lassen sich die Use Cases (nach Jacobson [JAC92]) des Anforderungsmodells entwickeln. Im Normalfall entspricht dann ein Use Case einer Aufgabe. Die Details des Use Cases werden dem Anwendertext entnommen.

Die Rollen des Geschäftsprozeßmodells entsprechen im wesentlichen den Aktoren des Anforderungsmodells.

Die mit linguistischer Analyse aus dem Anwendertext gewonnenen Objekte werden zu einem ersten Objektmodell (Domain Objects) geordnet. Im ersten Analysedurchlauf beschränkt man sich darauf, ca. 60 - 70 % der Gesamtanzahl der Objekte zu erfassen.

Zur Unterstützung dieser Phase ist es empfehlenswert, GUI-Prototyping einzusetzen, um dem Kunden die Möglichkeit zu bieten, die Präsentation der Anwendung möglicht realitätsnah zu überprüfen. Dieser erste GUI-Prototyp läßt sich aus den Use Cases und dem Domain Object Model herstellen.

Das Anforderungsmodell dient dazu, sich mit dem Kunden über den Inhalt der künftigen Anwendung zu unterhalten und zu einigen.

3.3 Vom Anforderungsmodell zum Analysemodell

Aus den Domain Objects des Anforderungsmodells werden Klassendigramme erstellt und vervollständigt.

Die Use Cases und Verarbeitungsregeln sind Ausgangsbasis für Objektdiagramme oder Interaktionsdiagramme.

Zustandsdiagramme ergeben sich aus den Informationen der Use Cases zu einem Objekt.

Das Analysemodell dient der DV-Seite eines Projekts, um sich für die weiteren Phasen ein komplettes Bild der Anwendung machen zu können.

4 Geschäftsprozeßmodell

4.1 Vorgehen Geschäftsprozeßmodell

Der Anwendertext beschreibt die Problemstellung und die Ziele bzgl. den Geschäftsprozessen. Hieraus sind zu jedem Geschäftsprozeß die zur Wertschöpfung notwendigen Aufgaben und deren Umfeld zu ermitteln. Die Ermittlung der Geschäftsprozeß-Objekte (Anforderung, Lösung, Leistungsportfolio) dient einerseits die Problemstellung zu konkretisieren, und andererseits das Problemfeld abzugrenzen und Schnittstellen zu ermitteln.

Der wesentliche Schritt der Geschäftsprozeßmodellierung ist die Festlegung der betrieblichen Abläufe (Aufgabenketten), den Aufgaben, aus denen sie sich zusammensetzen und deren Ergebnissen. Für jeden Ablauf sind die inhaltlich zusammengehörigen Aufgaben und das auslösende Ereignis zu beschreiben. Auch sind die Ablauffolge (sequentiell, parallel, iterativ, nebenläufig) und die Leistungen jeder Aufgabe zu bestimmen.

Um einen Geschäftsprozeß vollständig zu definieren, sind seine Produktionsfaktoren (Personal, IT-Ressourcen) und deren Verwendung im Prozeßablauf festzulegen. Außerdem ist das Prozeßmanagement festzulegen, damit der Geschäftsprozeß gesteuert werden kann. Hierunter sind das Vorgangs-, Qualitäts-, Informations- und Erfolgsmanagement zu verstehen.

4.2 Ergebnisse Geschäftprozeßmodell

Die Makro-Ebene des Geschäftprozeßmodells betrachtet den Gesamtprozeß, der in diesem Ansatz nicht weiter betrachtet wird. Auf der Mikro-Ebene, den Aufgabenketten, sind folgende Ergebnisse relevant:

- Aufgabenkette (Geschäftsprozeß)

  Die Aufgabenkette beschreibt die einzelnen Aufgaben, ihre Ablauffolge und deren Schnittstellen (Leistungen). Um den Ablauf und die inhaltliche Korrektheit der Aufgabenkette zu verifizieren, ist die Überprüfung durch einen Workflow-Prototypen hilfreich. Hierzu können marktgängige Workflow-Systeme oder auch herstellerunabhängige Workflow-Simulatoren, die bei Softlab zum Einsatz kommen [GRT95], herangezogen werden.

- Aufgabe (Geschäftsvorgang)

  Die einzelnen Aufgaben stellen betriebliche Funktionen dar, in der eine oder mehrere Transaktionen auf betriebliche Objekte abgewickelt werden. Eine Aufgabe wird durch ein Ereignis ausgelöst und durch ein Ereignis abgeschlossen. Die Leistungen, die eine Aufgabe mit anderen Aufgaben austauscht sind einerseits Informationen als Input, und andererseits Ergebnisse als Output. Bei der Beschreibung von Aufgaben werden erste grobe Informationen gewonnen, die in die Use Cases und Domain Objects eingehen werden.

- Produktionsfaktoren (Rollen)

  Aufgaben werden von Menschen und/oder Maschinen durchgeführt. Die vorhandenen Produktionsfaktoren, die einer Aufgabe zur Verfügung stehen, werden zum einen durch das Organisationsmodell und zum anderen durch die IT-Unterstützung beschrieben. Jeder Aufgabe und deren Auslösung werden die entsprechenden Produktionsfaktoren zugewiesen. Die dort beteiligten Produktionsfaktoren geben erste Hinweise auf die Aktoren im Use Case Model.

5 Anforderungsmodell

5.1 Vorgehen Anforderungsmodell

Ziel des Anforderungsmodells ist es, die Informationen, die aus dem Anwendertext entnommen wurden, in eine Form zu bringen, die der Kunde versteht, die aber gleichzeitig soweit formalisiert ist, daß sie als Basis für die weiteren Tätigkeiten der DV-Seite dient.

In einem ersten Schritt wird der Anwendertext einer linguistischen Analyse (manuell) unterzogen. Die hierdurch gefundenen, logischen Strukturelemente sind Objekt-Kandidaten für das **Domain Object Model**. Gefundene Objekte werden als Überschriften (**C**lasses) oder Beziehungen (**C**ollaborations) auf CRC-Karten [WIR90] festgehalten. Aktionen sind potentielle Methoden (**R**esponsibilities) und Ausführende von Aktionen werden für das **Use Case Model** als mögliche Aktoren notiert. Bereits dem Anwendertext zu entnehmende Attribute und Hinweise auf Superklassen werden auf den CRC-Karten vermerkt.

Aussagen über Schnittstellen des zu erstellenden Systems zu Fremdsystemen finden Eingang in das **Interface Model**. Aus den Use Cases und den Domain Objects läßt sich eine erste Form des **GUI-Prototyps** entwickeln.

Nicht-funktionale Anforderungen, wie z.B. Performance oder Sicherheitsaspekte werden im vorliegenden Aufsatz nicht behandelt, da sie meist vom Kunden schon sehr detailliert vorgegeben werden und deshalb keiner weiteren Präzisierung bedürfen.

5.2 Linguistische Analyse

Für die linguistische Analyse muß die **syntaktische** Struktur der Sätze in eine **logische** Struktur [BLA73] gebracht werden nach dem Schema:
1. Herausfinden der Aktion
2. Festlegen des Ausführenden der Aktion
3. Festlegen des Objekts der Aktion

Bsp:
Die Katze frißt die Maus.
Die Katze, die die Maus frißt.
Die Maus wurde sofort von der Katze gefressen.
Die kleine Maus wurde gefressen von der Katze.

Alle Sätze führen zur gleichen logischen Struktur:
Aktion = fressen
Ausführender = die Katze
Objekt = die Maus

5.3 Ergebnisse Anforderungsmodell

- Domain Object Model

  Für das Domain Object Model werden die CRC-Karten am besten an einer Pin-Wand zu einem Modell geordnet. Es wird versucht, Klassenhierarchien durch Angaben über Superklassen und Gemeinsamkeiten von Klassen aufzustellen, einzelne Klassen zu vervollständigen oder zu eliminieren und die Beziehungen der Klassen untereinander herauszuarbeiten. Die CRC-Karten lassen sich an der Pin-Wand sehr schnell umstecken, die gewünschte Möglichkeit für eine einfache Änderung des Modells ist dadurch gegeben. Die Beziehungen lassen sich auf dem Papier, das auf die Pin-Wand gespannt ist, mit Bleistift (Änderbarkeit!) zeichnen.

- Use Case Model

  Sämtliche Interaktionen zwischen Aktoren und dem System werden im Use Case Model beschrieben. Ein Use Case wird normalerweise aus einer Aufgabe des Geschäftsprozeßmodells entwickelt. Sollte es notwendig sein, eine einzelne Aktion des Systems genauer zu beschreiben oder mehrfach auszuführen, dann kann hierfür eine Verarbeitungsregel angehängt werden. Use Cases sollten knapp gehalten werden und im wesentlichen die Sicht des Anwenders aufzeigen.

- Interface Model

  Im Interface Model wird die statische Sicht der Schnittstellen des Systems zu den verschiedenen Aktoren (Benutzer, Fremdsysteme) definiert. Die Benutzerschnittstelle wird durch den GUI-Prototypen abgedeckt.

6 Ausblick auf das Analysemodell

6.1 Vorgehen Analysemodell

Mit dem Analysemodell werden die Erkenntnisse des Anforderungsmodells verfeinert und von allen Seiten beleuchtet, damit es als fachliche Ausgangsbasis für die DV-Entwicklung nutzbar ist. Ergebnisse des Analysemodells können mit Hilfe von Analysewerkzeugen festgehalten werden, die das Zeichnen von Diagrammen erlauben und die Inhalte der Diagramme weiterverarbeitbar machen. Weiterverarbeitbar bedeutet vor allem, daß Inhalte für die Generierung von Ergebnissen späterer Phasen genutzt werden.

Die Klassendiagramme stellen eine Weiterentwicklung des Domain Object Models dar. Für Abläufe in Use Cases, an denen mehrere (> 3) Objekte beteiligt sind, empfiehlt es sich, Objektdiagramme oder Interaktionsdiagramme anzufertigen. Bei Booch und Rumbaugh kann man wählen zwischen der Darstellung mittels Objektdiagramm oder Interaktionsdiagramm, Rumbaugh nennt die Objektdiagramme Objektinteraktionsdiagramme und die Interaktionsdiagramme Event Trace Diagramme.

Für das Zustandsdiagramm nimmt man eine einzelne Klasse des Klassendiagramms und zeichnet die Zustände auf, die das Objekt nach Ausführung der möglichen Methoden des Klassendiagramms jeweils annimmt.

6.2 Ergebnisse Analysemodell

- Klassendiagramm

  Das Klassendiagramm enthält alle zum Erstellungszeitpunkt gefundenen Objektklassen. Im ersten Arbeitszyklus wird noch kein Anspruch auf Vollständigkeit des Modells erhoben, man begnügt sich mit ca. 60-70 % der Gesamtmenge. Vermerkt werden im Klassendiagramm der Name des Objekts sowie die zugehörigen Attribute und Methoden.

- Objektdiagramm, Interaktionsdiagramm, Objektinteraktionsdiagramm

  Das Objektdiagramm/Objektinteraktionsdiagramm bzw. Interaktionsdiagramm beschreibt einzelne Ausschnitte aus Use Cases oder Verarbeitungsregeln, im Normalfall dürfte ein Diagramm einem Zweig einer Verarbeitungsregel entsprechen, könnte sich aber auch über mehrere Transaktionen eines Use Cases hinziehen oder aber einen Ausschnitt aus einer Verarbeitunsregel abdecken. Diese Art von Diagramm beschränkt sich auf die Beschreibung komplexerer Sachverhalte, wird also nicht für die gesamte Anwendung angefertigt.

- Zustandsdiagramm

  Das Zustandsdiagramm zeigt alle möglichen Zustände auf, die ein Objekt einer Klasse innerhalb der Anwendung annehmen kann und die Übergänge, die die Zustandsänderungen bewirken. Auslöser für die Übergänge sind Methoden, die auf dem Objekt ausgeführt werden. Auch für Zustandsdiagramme gilt, daß sie nur für komplexere Sachverhalte festzuhalten sind, d.h. wenn ein Objekt mehrere Zustände annehmen kann.

Literaturverzeichnis

[BLA73] G. Blanke. Einführung in die semantische Analyse. Hueber 1973.

[BOO94] G. Booch. Object-Oriented Analysis and Design with Applications, 2. Aufl. Bejamin/Cummings 1994.

[GRT95] Dr. Greiter. Spezifikationsprache und Simulator für Workflow. Softlab 1995.

[HON94] S. Honiden, N. Kotaka. Formalizing Specification Modeling in OAA. Conference Notes. Object World Germany 1994.

[JAC92] I. Jacobson et al. Object-Oriented Software Engineering. A Use Case Driven Approach. Addison-Wesley 1992.

[ÖST95] H. Österle. Business Engineering. Prozeß- und Systementwicklung. Band 1: Entwurfstechniken. Springer 1995.

[RUM91] J. Rumbaugh et al. Object-Oriented Modeling and Design. Prentice Hall 1991.

[SIN94] Prof. Dr. E. Sinz, Dr. M. Amberg. Objektorientierte Modellierung von Gechäftsprozessen. Conference Notes. Object World Germany 1994.

[WIR90] R. Wirfs-Brok et al. Designing Object-Oriented Software. Prentice Hall 1990.

Dateiname: EMISAPEN.DOC
Verzeichnis: N:\TI\OO_EU\OOA
Vorlage: C:\EMISA\EMISA.DOT
Titel:
Thema:
Autor: B. Röllnreiter
Stichwörter:
Kommentar:
Erstelldatum: 13.09.95 10:03
Version: 4
Letztes Speicherdatum: 13.09.95 13:46
Zuletzt gespeichert von: B. Röllnreiter
Letztes Druckdatum: 13.09.95 13:47
Nach letztem vollständigen Druck
Anzahl Seiten: 8
Anzahl Wörter: 2.414 (ca.)
Anzahl Zeichen: 13.762 (ca.)