Objektorientierung in IT-Projekten der Praxis

Joachim Wehler

Ludwig-Maximilians-Universität München

Vorlesung im Sommersemester 2007

1 Einleitung

Diese Vorlesung *Objektorientierung in IT-Projekten der Praxis* behandelt die objektorientierte Methode als einen Weg, um in kommerziellen IT-Projekten Fachkonzepte zu schreiben. Das Fachkonzept ist das Ergebnis der Analysephase. Insofern handelt diese Vorlesung von der objektorientierten Analyse (OOA).

Die Vorlesung erstrebt eine Verbindung von Theorie und Praxis. Alle theoretischen Ergebnisse sollen vor dem Hintergrund der konkreten Erfordernisse kommerzieller Projekte gesehen werden. In solchen Projekten müssen sich die dargestellten Methoden bewährt haben. Die dargestellte Theorie wird also nicht um ihrer selbst willen oder aus allein ästhetischen Gründen entwickelt.

In kommerziellen Projekten analysieren die Berater einer IV-Abteilung die betriebswirtschaftlichen Anforderungen eines Kunden, um ein Konzept für die anschließende Umsetzung in Software zu erstellen. Diese Analyse wird heute objektorientiert durchgeführt. Welche Methoden stellt die Informatik hierfür zur Verfügung?

Kapitel 2 *Was sind kommerzielle Projekte?* nennt die wesentlichen Eigenschaften von IT-Projekten in der Praxis und grenzt sie ab gegen Forschungsprojekte oder private Programmierung in der Freizeit. Jedem kommerziellen Projekt liegt ein Phasenmodell zugrunde, das für die einzelnen Phasen bestimmte Ergebnisse vorschreibt.

Das folgende Kapitel 3 *Prinzipien objektorientierten Denkens* behandelt die theoretischen Grundlagen der Vorlesung. Die Grundsätze der objektorientierten Analyse werden als vier Prinzipien entwickelt. Sie lassen sich in Form eines Metamodells formalisieren.

Objektorientierte Modellierungssprachen sind ein Mittel, diese Prinzipien anzuwenden. Auch Modellierungssprachen, die von Hause nicht für eine objektorientierte Analyse entworfen wurden, können benutzt werden, um auf Basis dieser Prinzipien ein Fachkonzept zu schreiben. Kapitel 4 *Sprachen zur Unternehmensmodellierung* behandelt vier solche Modellierungssprachen.

Kapitel 5 *Fallbeispiel eines kommerziellen Projekts* zeigt an einem Praxisbeispiel die Anwendung des in dieser Vorlesung vermittelten Stoffes.

Am Ende folgen in Kapitel 6 *Ausblick* einige Schlaglichter auf allgemeine Erfahrungen aus dem Berufsalltag der IT-Branche und eine persönliche Einschätzung.

München, August 2007. Joachim Wehler

2 Was sind kommerzielle Projekte?

Programming-in-the-Large folgt anderen Regeln als die Programmierung der eigenen Homepage. Und kommerzielle Industrieprojekte sind large-scale Systeme, keine Fallstudien.

2.1 Abgrenzung und Definition

Wir grenzen kommerzielle Industrieprojekte ab gegen nicht-kommerzielle Projekte wie

- Wissenschaftliche Arbeiten

  Bei wissenschaftlichen Arbeiten zur objektorientierten Modellierung (Diplomarbeiten, Dissertationen, Tagungsbeiträge) steht häufig am Anfang die Theorie. Zu der Theorie wird nachträglich ein Anwendungsfall konstruiert. Dabei ist der Autor frei, den Anwendungsfall so zurechtzuschneiden, daß er auf die Theorie paßt und im Rahmen der für die Arbeit vorgesehenen Zeit bewältigt werden kann. Die Arbeit wird von den Fachkollegen nach wissenschaftlichen Gesichtspunkten geprüft: Ist der Ansatz interessant, löst er ein theoretisches Problem? Eine Prüfung im Umfeld der Wirtschaft findet i. a. nicht statt.

  Ein Erfolg bei diesem Vorgehen spricht nicht gegen die Richtigkeit der Theorie. Er sagt allerdings auch nichts über ihre praktische Brauchbarkeit.

- Staatlich geförderte Forschungsprojekte

  In Deutschland finanziert die Bundesregierung eine Reihe von Projekten der Angewandten Informatik, die gemeinsam von Softwarefirmen, Industrieunternehmen und universitären oder halbstaatlichen Instituten abgewickelt werden. Die Themen dieser Projekte heißen z.B. „Prozeßmanagement bei der industriellen Produktentwicklung“ oder „Prozeßbausteine der Logistik“.

  Diese Forschungsprojekte bieten die Möglichkeit, Konzepte für das Software Engineering zu erproben, ohne daß sich diese sofort im Produktivbetrieb bewähren müssen. Allerdings geht es hierbei nicht um wissenschaftliche Grundlagenforschung. Manchmal werden diesen Projekten schon beim Start Randbedingungen in die Wiege gelegt, die ein wissenschaftliches Vorgehen behindern: Die beteiligten Softwarefirmen setzen ihre eigenen Plattform als Projektstandard durch, ebenso ihr Case-Tool oder andere Standardsoftware. Die beteiligten Industriefirmen wiederum haben kein Interesse daran, ihre neuesten firmeninternen Erkenntnisse offenzulegen und damit auch ihren Konkurrenten zugänglich zu machen. Das Projektergebnis sind häufig schlanke Prototypen.

  Dem Projektmitarbeiter auf Seiten der beteiligten Firmen, der sonst nur die Tagesarbeit seines Softwarehauses kennt, bieten solche Forschungsprojekte den Kontakt mit Kollegen aus der Forschung. Im Alltag kommerzieller Projekte besteht selten Gelegenheit, sich mit theoretischen Fragen zu den angewendeten Konzepten auseinanderzusetzen.

2.1.1 Definition

Ein *kommerzielles IT-Projekt* ist eine Dienstleistung, die ein Lieferant für einen Kunden gegen Entgelt erbringt. Gegenstand dieser Dienstleistung ist ein Konzept und seine Realisierung in Form von Soft- und Hardware.

Kommerzielle Projekte werden nicht durchgeführt, um neue Software-Methoden zu erproben. Ein Unternehmen setzt ein Software Projekt auf, um seine betriebswirtschaftlichen oder technischen Prozesse besser zu unterstützen. Die IV-Abteilung ist Dienstleister für das Kerngeschäft eines Unternehmens. Und das heißt Absatz von Produkten oder Dienstleistungen auf dem Markt.

Bei Auftragserteilung wird ein Zeit- und Kostenrahmen vereinbart. Bei Festpreisprojekten ist der Rahmen relativ starr, er darf vom Auftragnehmer nur in engen Grenzen und in begründeten Fällen überschritten werden. Aufwandsprojekte sind in dieser Beziehung flexibler, aber auch hier werden Obergrenzen festgelegt.

2.2 Branchenorientierung

IT-Projekte lassen sich klassifizieren nach

- Branche: Industrie, Banken, Versicherung, Verkehr, Öffentliche Verwaltung, etc.
- Art der Software: Standardsoftware, Individualsoftware.

Ein kommerzielles Projekt sollte nicht mit Fragen der Modellierungsmethode beginnen, sondern mit der Einordnung des Projekts in die betriebswirtschaftlichen Abläufe des Unternehmens. Da diese Abläufe branchenspezifisch sein können, erwartet der Kunde von den Projektmitarbeitern in der Rolle von Beratern (Consultant) neben dem IT-spezifischen auch branchenspezifisches Wissen.

Die folgenden Screenshots zeigen eine Reihe von Selbstdarstellungen deutscher IT-Firmen aus dem Internet und betonen die Rolle des Branchenwissens.

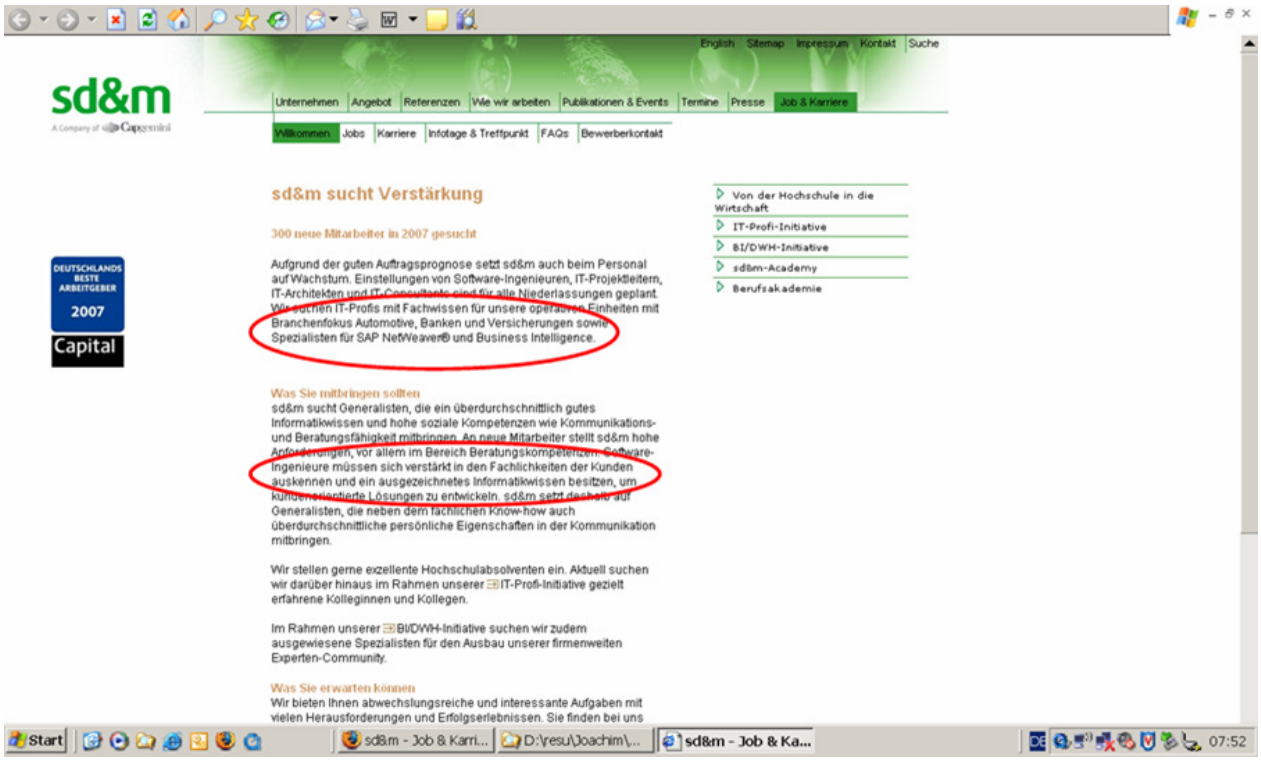


Abbildung 1: sd&m, Branchenorientierung

Abbildung 2: softlab GmbH, Branchenorientierung

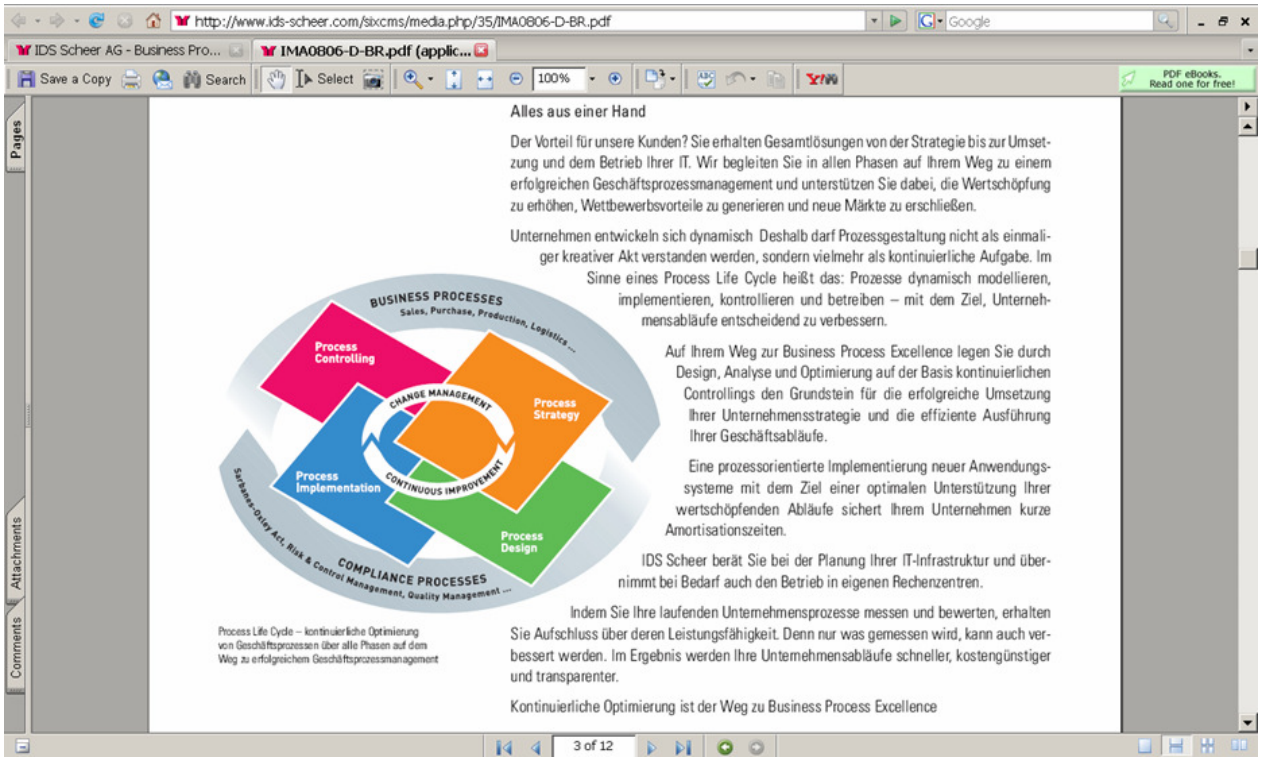


Abbildung 3: IDS Scheer AG, Firmendarstellung

Für einen Industriebetrieb haben sich dabei zwei Prozeßketten als grundlegend herausgestellt (Abbildung 4):

- Time-to-Market: Entwicklung eines Produktes zur Serienreife
- Time-to-Customer: Auftragsabwicklung eines Kundenauftrags.

Ein Berater in der Analysephase eines kommerziellen Projektes bei einem Industrieunternehmen muß diese beiden Prozeßketten im Kopf haben. Er sollte sie als oberstes Gliederungsprinzip aller betriebswirtschaftlicher Anforderungen heranziehen.

Abbildung 4 ist ein Ausschnitt aus der obersten Ebene eines unternehmensweiten Geschäftsprozeßmodells. Es gilt für einen Industriebetrieb mit auftragsgesteuerter Fertigung.

Abbildung 4: Prozeßketten eines Industriebetriebes (Quelle: Softlab GmbH)

Beide Prozeßketten sind als Wertschöpfungsketten dargestellt. Jeder einzelne Abschnitt beider Wertschöpfungsketten stellt wiederum ein Prozeß dar.

Im einzelnen bedeuten in der Prozeßkette Time-to-Market:

- Produktmarketing: Welche Produkte oder Produktvarianten lassen sich auf welchen Märkten verkaufen?
- Produktentwicklung: Konzeption und Entwicklung des Produktes zur Serienreife
- Produktionsvorbereitung: Konzeption und Realisierung der zur Herstellung des Produktes notwendigen Fertigungsverfahren und Produktionsmittel.

Bei der Prozeßkette Time-to-Market handelt es sich um die Teilprozesse:

- Absatzplanung: Unternehmerische Entscheidung, welche Mengen an Produkten und Produktvarianten in den kommenden Zeitperioden am Markt abgesetzt werden sollen.
- Vertrieb: Akquisition von Kundenaufträgen für Produkte
- Produktionsplanung: Festlegung, wann und in welchen Werken die geplanten Kundenaufträge gefertigt werden und Planung des Bedarfes an Komponenten und Teilen.
- Beschaffung: Einkauf von Komponenten und Teilen bei Zulieferern und Bereitstellung für die Produktion.
- Produktion: Bau oder Montage der Produkte

- Distribution: Auslieferung des Produktes an den Kunden
- Faktura: Erstellung und Versand der Rechnung an den Kunden
- Wartung, Service: Reparatur des Produktes beim Kunden

Zu den Unterstützungsprozessen gehören z.B. Prozesse im Controlling, in der Personalwirtschaft und in der Datenverarbeitung.

Es ist nun leicht, den in Abbildung 4 dargestellten Sachverhalt objektorientiert zu formulieren. Dazu identifizieren wir in den beiden Prozeßketten als erstes die folgenden Geschäftsklassen, die in den Prozessen entweder selbst bearbeitet oder als Hilfsmittel zur Bearbeitung gebraucht werden:

- Markt
- Fertigungsmittel
- Produkttyp
- Produkt
- Kundenauftrag
- Teil
- und Zeit.

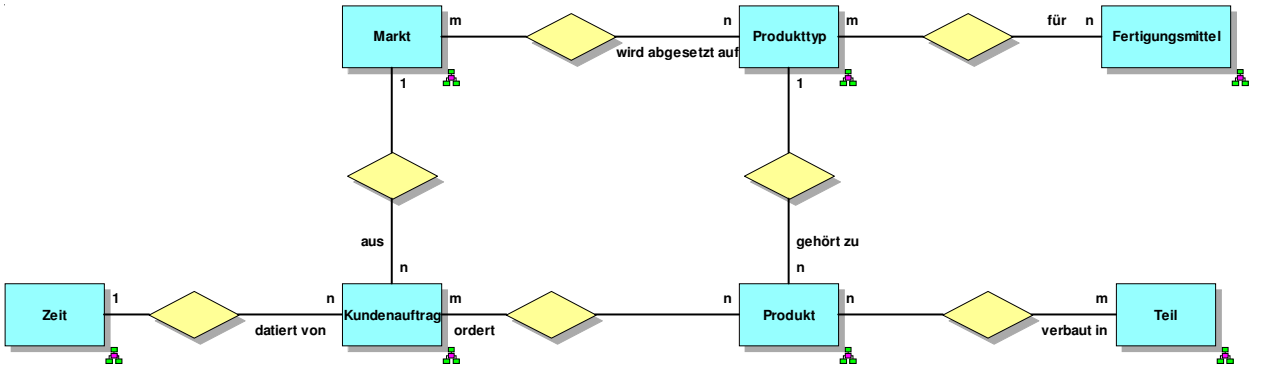


Abbildung 5: Geschäftsklassen

Produkttypen werden auf Märkten abgesetzt, wobei derselbe Produkttyp auch auf mehreren Märkten angeboten wird. Es gibt auch Märkte, die erst in der Zukunft eröffnet werden sollen. Auf diesen werden noch keine Produkttypen abgesetzt. Zu jedem Produkttyp werden viele Produkte hergestellt, allerdings erst nach der Serienreife, davor gibt es noch kein Produkt. Um die Produkte eines bestimmten Typs zu fertigen, sind eigene Fertigungsmittel nötig. Mit manchen Fertigungsmitteln können auch unterschiedliche Produkttypen hergestellt werden. Ein Produkt wird bei seiner Fertigung aus Teilen zusammengebaut. Produkte werden nur hergestellt, wenn ein Kundenauftrag dazu vorliegt. Ein Kundenauftrag bestellt immer mindestens ein Produkt, kann aber auch mehrere Produkte umfassen.

Unter Verwendung dieser Geschäftsklassen lassen sich die beiden Prozeßketten von Abbildung 6 objektorientiert darstellen, indem jede Aktivität einer Prozeßkette als Operation einer dieser Geschäftsklassen aufgefaßt wird (siehe Abbildung 6)

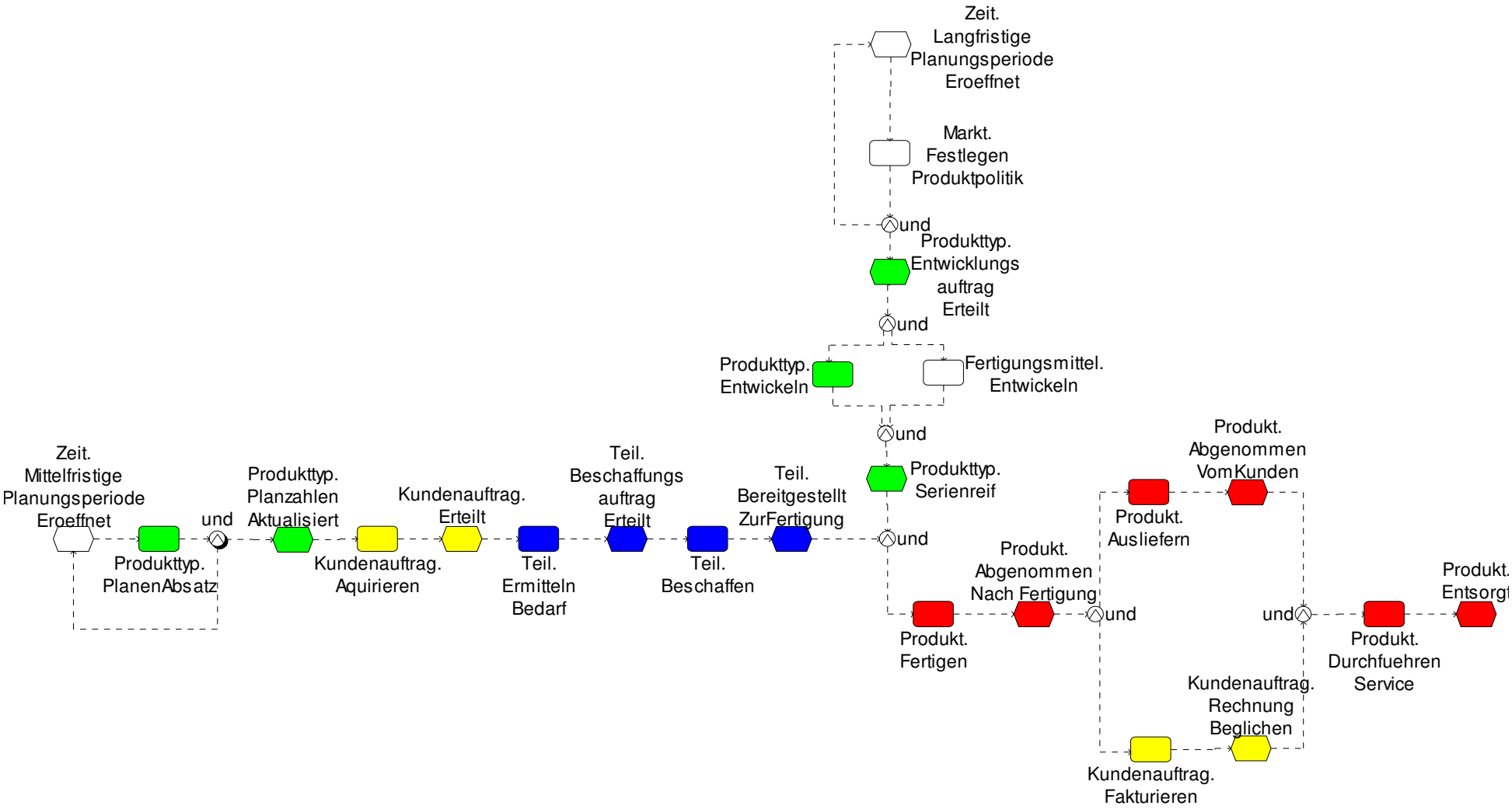


Abbildung 6: Prozeßketten eines Industriebetriebes in objektorientierter Darstellung

2.3 Phasenmodell der Projektabwicklung

Kommerzielle Projekte werden nach einem Phasenmodell abgewickelt. Es strukturiert den Zeitplan, legt die Ergebnisse fest und beendet jede Phase mit einem Review der abgelieferten Ergebnisse.

2.3.1 Phasen und Ergebnisse im Überblick

Projekt-phase	**Name**	**Ergebnis**	**Beschreibung**
1	Problemanalyse = Grobkonzep-tion	Grobkonzept	Für die im Projektauftrag festgehaltene Problem-stellung werden zunächst Lösungsalternativen er-arbeitet und deren Machbarkeit untersucht. Da-nach erfolgt die Auswahl einer Lösungsalternative und die Genehmigung durch den Auftraggeber. Auf der Basis der gewählten Lösungsalternative wird das Grobkonzept entworfen. Zu dem ausge-arbeiteten Grobkonzept wird ein Leistungsstufen-plan erstellt. Im Leistungsstufenplan werden Teilsysteme und Leistungsstufen definiert und terminiert. Dabei werden insbesondere die Abhängigkeiten zwi-schen den einzelnen Leistungsstufen und Teilsy-stemen berücksichtigt.
2	Fachkonzeption (OOA)	Fachkonzept, Pflichtenheft	Die fachlichen Aufgaben und Abläufe dieser Lei-stungsstufe werden verfeinert und aus Benutzer-sicht beschrieben, d.h. IV-spezifische Gesichts-punkte und Betrachtungsweisen werden in den Hintergrund gestellt. Das Fachkonzept baut auf den Ergebnissen der Vorphasen bzw. fertiggestell-ten Leistungsstufen auf, schreibt diese bei Bedarf fort und richtet sich an zwei Zielgruppen: • Benutzer aus der Fachabteilung • IV-Entwickler Mit dem Fachkonzept wird grundsätzlich beab-sichtigt, die gewonnenen fachlichen Erkenntnisse für beide Zielgruppen zu nutzen, d.h.: • fachliche Aspekte in die Benutzerdokumenta-tion zu übernehmen, • Basis für die IV-technische Ausarbeitung zu

Projekt-phase	**Name**	**Ergebnis**	**Beschreibung**
			sein.
3	IT-Konzeption (OOD)	IT-Konzept	In dieser Phase wird das IV-System entworfen. Ziel ist die Umsetzung des Fachkonzepts in einen implementierungsorientierten Entwurf. Die Systemarchitektur berücksichtigt das geplante Gesamtsystem und nicht nur die aktuelle Leistungsstufe. Prototypen für komplexe, kritische oder technisch neue Teile werden erstellt, um die technische Machbarkeit nachzuweisen.
4	Realisierung (OOP)	IT-System	In dieser Phase erfolgt die Realisierung der in Fach- und IV-Konzept entworfenen Leistungsstufe: von Konstruktion, Programmierung und Test der einzelnen Klassen über die schrittweise Integration der (Sub-)Systeme bis zu Test und Integration des gesamten IV-Systems der Leistungsstufe. Ergebnis dieser Phase ist das getestete IV-System der Leistungsstufe.

Abbildung 8: Phasenmodell für IT-Projekte zur Einführung von Individualsoftware

2.3.2 Phasen und Ergebnisse im Detail

IT Process Quality Management

ITPM ITPM

Projektmodell - Objekt Orientierung

Project Model - Object Orientation

Projektergebnisse für OO IV-Projekte

Version 2.1
Status: freigegeben
Stand: 21.03.2003

Version 2.1
Status: released
Date: 21.03.2003

0 Projektdefinition / Project Initiation

Projektauftrag / Project Brief

Vereinbarung - bevor das Projekt startet

Die Projektdefinition ist die Vorphase der eigentlichen Projektabwicklung. Vor Auslösung nennenswerter Aufwände soll der Rahmen für das Projekt geschaffen und zwischen Auftraggeber und künftigem Projektleiter vereinbart werden.

Agreement - before the project starts

Project Initiation represents the pre-phase of project development. The Project Brief sets the framework for the project and is agreed by the Process Owner and the Project Leader before any significant expenditure is made on the project.

▶ *Projektauftrag* / *Project Brief*

A	**Projekthintergrund**	**Project background**
	– Historie, Stärken, Schwächen	– History, Strengths, Weaknesses
	– Gründe für das Projekt	– Reasons for project
A	**Ziele**	**Objectives**
A	**Anforderungen** (grob) an das Projektergebnis	**Requirements** (outline) regarding the project solution
A	**Abgrenzung**	**Scope**
	– Betrachtete Umfänge (fachlich, technisch)	– Business / technical / problem areas to be addressed
	– Explizite Ausschlüsse	– Specific exclusions
A	**Betroffene Schnittstellen**	**Interfaces affected**
	– Organisationseinheiten, Werke, ...	– Organisation units / plants involved
P	– IV-Systeme	– Existing systems
P	**Grobe Vorgehensweise** (geplante Projektaktivitäten und Ergebnisse)	**Approach** (planned project activities and deliverables)
P	**Rahmenbedingungen / Voraussetzungen**	**Constraints and assumptions**
A	**Risiken, Schutzaspekte**	**Risks, protection aspects**
P	**Lösungsalternativen** (grob, sofern schon bekannt)	**Alternative solutions** (outline form, where known)
A	**Wirtschaftlichkeit / Projektbegründung**	**Profitability / project justification**
	– Kosten / Nutzen (grobe Abschätzung)	– Costs / benefits (initial soft estimate)
	– Projektbegründung (qualitativer Nutzen, Strukturprojekt)	– Project justification (qualitative benefits, strategic reasons)
P	**Projektorganisation**	**Project organisation**
	– Auftraggeber, Lenkungskreis, Eskalationsweg	– Process Owner, Steering Committee, escalation path
	– Projektleiter, Team, QSP	– Project Leader, project team, QAP
	– Interne, externe Partner	– Potential suppliers
	– Rollen / Verantwortlichkeiten	– Definition of roles / responsibilities
A	**Offene Punkte**	**Open issues**
P	**Projektplanung**	**Project and quality planning**
	– Aufwand, Ressourcen, Kosten, Meilensteine, Termine *für die Problemanalyse (Phase 1)*	– Effort, resources, costs, milestones, schedules *for the Requirements Analysis (phase 1)*
	– Ausblick für das gesamte Projekt	– Outlook for the whole project
	– Tailoring	– Tailoring

✓ *Gateway* / *Gateway*

A	**Abnahme Projektauftrag** und Mittelfreigabe für *die Folgephase* oder Abbruch	**Acceptance of Project Brief** and release of resources for *next phase* or project cancellation

1 Problemanalyse / Requirements Analysis

Grobkonzept / Business Proposal

Lösungsalternativen / Skizze des vorgeschlagenen Lösungswegs

Das Grobkonzept skizziert die angestrebte Lösung, um sicherzustellen, daß sich damit die Projektziele realistisch erreichen lassen.

Alternative Solutions / Outline of recommended solution

The Business Proposal presents an outline of the solution to be implemented to ensure that the solution is realistic and will achieve the project's objectives

Im ersten Schritt werden Lösungsalternativen untersucht. Im zweiten Schritt wird auf Basis der genehmigten Lösungsalternative das eigentliche Grobkonzept erstellt.

Zum Grobkonzept wird ein Leistungsstufenplan erstellt.

Im Leistungsstufenplan werden Teilsysteme und Leistungsstufen definiert und terminiert. Dabei werden insbesondere Abhängigkeiten zwischen einzelnen Leistungsstufen, Teilsystemen und anderen Projekten berücksichtigt.

Der Leistungsstufenplan ist das zentrale Steuerungsinstrument für die Phase X, Iterationen

ject's objectives.

As a first step alternative solutions are identified and evaluated. In the second step the approved solution forms the basis for the development of the Business Proposal.

A Release Package Plan is included in the Business Proposal.

Sub-systems and Release Packages are defined and scheduled in the Release Package Plan. During this the inter-dependencies of the Release Packages, the sub-systems and other projects are considered.

The Release Package Plan is the central control instrument for Phase X, Iterations.

	▶ ***Alternativenauswahl***	***Decide solution***
P	**Ist-Analyse**	**Analysis of current situation**
A	– Geschäftsprozesse (fachliche Abläufe aus Anwendersicht)	– Business processes (processes from the user viewpoint)
A	– Aufbauorganisation und Standorte	– Organisation structure and locations
	– Schnittstellen	– Interfaces
	– Altsysteme	– Existing IT system
	– *Klassenmodell*/Grobdatenmodell	– *Class model*/Soft data model
A	**Ziele** (meßbar, inklusive Abnahmekriterien)	**Objectives** (measurable, including acceptance criteria)
A	**Anforderungen** (präzisiert)	**Requirements** (refined)
A	**Sollkonzept für Kernprozesse / Kernbereiche**	**Target concept for core processes / areas**
	– Tragende Geschäftsprozesse (fachliche Abläufe aus Anwendersicht) *mit Geschäftsobjekten (Aktivitätsdiagramm)*	– Core business processes (processes from the users viewpoint) *with business objects (Activity diagram)*
P	*– Wichtige Schnittstellen (Kontextdiagramm)*	*– Important interfaces (context diagram)*
P	**Lösungsalternativen**	**Alternative solutions**
	– Beschreibung	– Description
	– Liste der Alternativen der engeren Wahl	– Short list of suitable solutions
	Zu jeder Alternative der engeren Wahl:	For each short listed solution:
A	– Vorschlag Geschäftsprozesse	– Proposal for business processes
	– Schnittstellen (Kontextdiagramm)	– Interfaces (context diagram)
	– Technische Umsetzung	– Technical implementation
	– Machbarkeitsüberlegungen [inklusive Prototypen]	– Feasibility studies [including prototypes]
	– Migrationsüberlegungen	– Migration considerations
A	– Wirtschaftlichkeit	– Profitability
P	**Lösungsvorschlag**	**Recommended solution**
	– Eindeutige Empfehlung für **eine** Lösungsalternative	– Recommendation of **one** of the alternative solutions
A	**Genehmigung des Lösungsvorschlags**	**Approval of the recommended solution**

	▶ ***Grobkonzept* und Leistungsstufenplan**	***Business Proposal* and Release Package Plan**
A	**Sollkonzept (fachlich) für die gewählte Lösung**	**Target concept for the approved solution**
	– Geschäftsprozesse (fachliche Abläufe aus Anwendersicht) inklusive Zuständigkeiten *(Aktivitätsdiagramm)*	– Business processes (from the user viewpoint) including responsibilities *(Activity diagram)*
	– Anwendungsfälle (Use Cases)	– Use Cases
	– Aufbauorganisation und Standorte (Änderungen an den Rollen und Zuständigkeiten aller Beteiligten)	– Organisation structure and locations (changes to roles and responsibilities of all parties)
P	– Schnittstellenübersicht: Kontextdiagramm und Informationsflüsse (auf *Klassen*/Entitätsebene)	– Interfaces overview: context diagram and information flows (at the *class*/entity level)
P	**Sollkonzept Infrastruktur**	**Target concept for infrastructure**
	– Eckpunkte für Systeminfrastruktur (Hardware, Netz, Basissoftware)	– Cornerstones for system infrastructure (hardware, network, supporting)
	Verteilungsdiagramme	· *Deployment diagrams*
	· *Musterarchitektur*	· *Standard architecture*
	– Allgemeine Klassenbibliotheken, Frameworks	– General class libraries, frameworks
	– Technische Ausrüstung für Geschäftsprozesse (neben IV)	– Technical equipment required for business processes (in addition to IT)
	– Betrieb	– Production acceptance
P	**Leistungsstufen*plan* / Teilsysteme**	**Release Package *Plan* / sub-systems**
	– Leistungsstufen für die Teilsysteme und über Teilsysteme hinweg	– Release Packages for the sub-systems and across sub-systems
	Migrationsüberlegungen	· Migration considerations
	Überlegungen zur technischen Umsetzung	· Considerations for technical implementation
P	**Rahmenbedingungen / Voraussetzungen** (fortgeschrieben)	**Constraints and assumptions** (continued)
A	**Wirtschaftlichkeit** (fortgeschrieben)	**Profitability** (continued)
A	**Risikoanalyse**	**Risk analysis**
A	**Offene Punkte** (aktualisiert)	**Open issues** (updated)
P	**Projektplanung** (aktualisiert)	**Project planning** (updated)

	✓ ***Gateway***	***Gateway***
A	**Abnahme Grobkonzept *und Leistungsstufenplan***	**Acceptance of Business Proposal *and Release Package Plan***
	und Mittelfreigabe für die erste	*and release of resources*

Leistungsstufe (eine Iteration der Phase X)

for the first Release Pakkage (one iteration of Phase X)

oder Abbruch

or project cancellation

X Iterationen — *Iterations*

Die folgenden Ergebnisse werden iterativ erarbeitet und mit jedem Iterationsschritt verfeinert. Jede Leistungsstufe wird in den Phasen X.2 bis X.5 analysiert, entworfen, realisiert, getestet sowie üblicherweise in Produktion gebracht und in die Wartung überführt (je nach Festlegung im Leistungsstufenplan).

Für jede Produktionsaufnahme ist ein Integrationstest durchzuführen. Wichtig dabei ist die Integration dieser Leistungsstufe in die bisher erarbeiteten Leistungsstufen.

Nach Abschluß einer Iteration wird der Leistungsstufenplan überarbeitet und erneut abgenommen.

The following deliverables are worked out iteratively and are refined step by step. In the phases X.2 to X.5 each Release Package is analysed, designed, implemented, tested and usually brought into production and transferred into maintenance (according to the Release Package Plan).

An integration test is to be carried out for each Release Package that goes into production. In the process it is important to integrate this Release Package into those that have been worked out up to now.

Following completion of an iteration, the Release Package Plan is revised and accepted once more.

X2 Aufgabendefinition — Requirem. Definition
Fachkonzept — System Proposal

Fachliche Beschreibung der Lösung („Was“)

Das Fachkonzept beschreibt die künftige Lösung *der Leistungsstufe* aus fachlicher Sicht. Es dient als Vereinbarung, was die Lösung beinhaltet und was nicht.

Es baut auf den Ergebnissen der Vorphasen bzw. bereits fertiggestellter Leistungsstufen auf.

Anhand des Fachkonzepts sollen einerseits die Benutzer aus der Fachabteilung das geplante System beurteilen können, andererseits bildet es die Basis für IV-Konzept und Implementierung.

Das Ergebnis OO-Analysemodell wird (inkl. der vom System unterstützten Geschäftsprozesse) mit UML modelliert.

Business and functional description of the solution („What“)

The System Proposal defines the planned solution *of the Release Package* from a business viewpoint. It provides an agreement what the planned solution will include and exclude.

It builds on the deliverables of the pre-phases or completed Release Packages.

The System Proposal should enable the users / user departments to assess the planned system and on the other hand forms the basis for the System Design to be undertaken in the next phase.

▶ *Fachkonzept* — *System Proposal*

A **Geschäftsprozesse** (fortgeschr.) — **Business processes** (continued)

- *Aktivitätsdiagramme* — *Activity diagrams*
- Abläufe inkl. Zuständigkeiten — Processes, including responsibilities
- Aufbauorganisation und Standorte — Organisational structure and locations

P ***OO-Analysemodell*** — ***OO Analysis Model***

- *Systemarchitektur (Verteilungsdiagramm)* — *System architecture (Deployment diagram)*

A
- *Anwendungsfalldiagramm (fortgeschr.)* — *Use Case diagram (continued)*
- *Statisches Modell* — *Static model*
  - *Klassendiagramm (fachlich)* — *Class diagram (business)*
  - *[Objektdiagramm]* — *[Object diagram]*
- *Dynamisches Modell* — *Dynamic model*
  - *Interaktionsdiagramm (Sequenz- / Kollaborationsdiagramm)* — *Interaction diagram (Sequence / Collaboration diagram)*
  - *Aktivitätsdiagramm* — *Activity diagram*
  - *[Zustandsdiagramm]* — *[State diagram]*
- *Softwarearchitektur (Komponentendiagramm)* — *Software architecture (Component diagram)*

P ***Modell für persistente Datenhaltung (logisch)*** — ***Model for persistent data storage***

- *Relationales* Datenmodell — *Relational* data model
  - Entity-Relationship-Diagramm — Entity relationship diagram
  - Entitäten mit Schlüsseln und Mengengerüst — Entities with identifiers and quantities
  - Beziehungen, Attribute, Regeln (Constraints) — Relationships, attributes, constraints

P **Schnittstellen** — **Interfaces**

- Schnittstellenübersicht: Matrix — Interfaces overview: matrix
- Schnittstellenkontrakte — Interface contracts

P **Benutzermodell** — **User interface model**

- Berechtigungskonzept — Access authorisation / control concepts
- Oberflächengestaltung und Dialogabläufe — Design of user interface and dialogue processes
- Hilfesystem — Help system
- Prototyp für die Oberfläche — Prototype for user interface

A **Übergreifende Anforderungen** — **System implementation requirements**

- Einführung und Schulung — Introduction and training
- Migration — Migration
- Notfall / Fallback für den Fachprozess — Contingency and fall back for business continuance
- Informationsschutz / -sicherheit (VIVA) — Information protection / security (CIA^2)
  - **V**ertraulichkeit — **C**onfidentiality
  - **I**ntegrität — **I**ntegrity
  - **V**erfügbarkeit (Backup, Restart, Recovery, Notbetrieb) — **A**vailability (back-up, restart, recovery, emergency operation)
  - **A**uthentizität — **A**uthenticity
- Performance — System performance
- Archivierung — Archiving
- Betrieb — Production acceptance (continued)

A **Testspezifikation** — **Test specification**

- Fachliche Testfälle (mit erwarteten Ergebnissen) — Business process test cases (and expected results)
- Abnahmeverfahren und -kriterien — Acceptance process and criteria

A **Benutzerdokumentation** — **User documentation**

- Fachliche Beschreibung — System description from user viewpoint
- Bedienung mit Fehlerliste — Handling, list of error messages
- Fachglossar — Glossary

P **Sollkonzept Infrastruktur** (fortgeschrieben) — **Target concept for infrastructure** (continued)

A **Wirtschaftlichkeit** (fortgeschr.) — **Profitability** (continued)

A	**Risikoanalyse** (aktualisiert)	**Risk analysis** (updated)
A	**Offene Punkte** (aktualisiert)	**Open issues** (updated)
P	**Projektplanung** (aktualisiert)	**Project planning** (updated)

✓ ***Gateway***	***Gateway***
Abnahme Fachkonzept	**Acceptance of System Proposal**
und Freigabe der Folgephase[n] der Leistungsstufe (Iteration)	*and release of next phase[s] of the Release Pakkage (Iteration)*
oder Abbruch	or project cancellation

3 Prinzipien objektorientierten Denkens

Das Fachkonzept hat unterschiedliche Leserkreise, siehe Abbildung 10. Sie stellen unterschiedliche Anforderungen an das Fachkonzept.

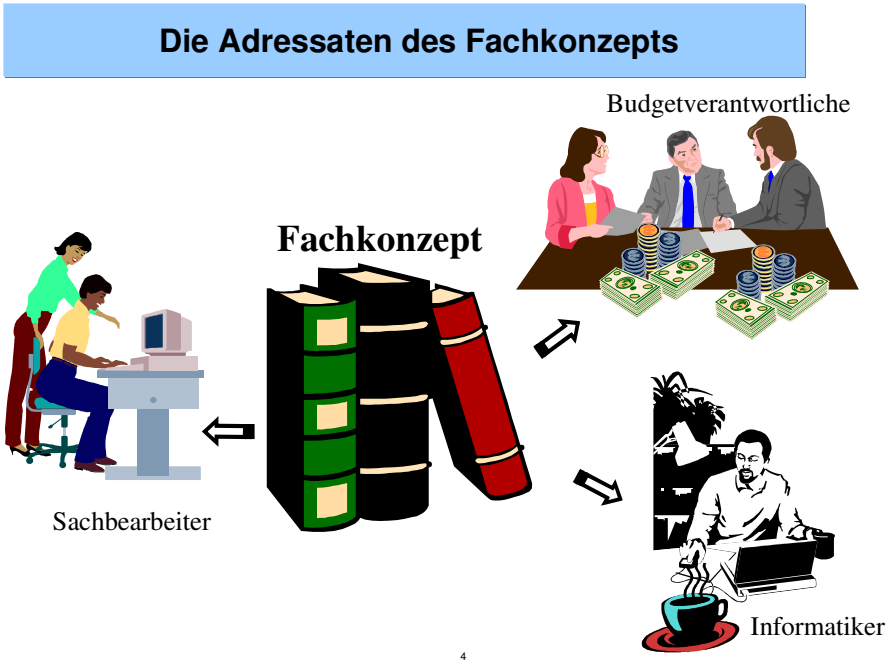


Abbildung 10: Verschiedene Adressaten des Fachkonzepts

Die gegenwärtige Propagierung der UML als Modellierungssprache auch für objektorientierte Analyse (OOA) verschärft das Problem noch, das sich aus den unterschiedlichen Leserkreisen des Fachkonzepts ergibt.

Kein Anwender in der Fachabteilung will Sequenzdiagramme oder korrekte Aktivitätsdiagramme lesen. Bestenfalls noch ein paar Usecasediagramme mit ihren netten Strichmännchen als Aktoren. Auch Klassendiagramme sind nicht die Methode der Wahl. Schon die Entity-Relationship Diagramme der letzten 20 Jahre galten als eine Geheimsprache von IT-Spezialisten.

Die Lösung kann also nicht heißen: Mehr UML in's Fachkonzept.

Noch eine zweite Schwierigkeit kommt in jüngster Zeit hinzu. Und das ist die Diskussion über die „richtige" Sprache zur Modellierung von Geschäftsprozessen. Diese Diskussion entzündet sich zwischen den Anhängern von UML-Aktivitätsdiagrammen, den Anhängern der „Ereignisgesteuerten Prozessketten" (EPK) und den Anhängern von Sprachen wie BPEL oder BPMN. Die erste Gruppe nennt ihr Vorgehen „Objektorientierte Prozeßmodellierung". Die Anhänger von EPKs tun sich schwerer, den Objektbezug ihrer Modellierungssprache herzustellen. BPEL oder BPMN sind in der Analysephase überhaupt nicht objektorientiert, sondern BPEL erst bei der Implementierung.

Unbestritten kann jede Seite auf Erfolge ihrer Sprache verweisen. UML ist als Sprache für das objektorientierte Design (OOD) unumstritten. Aber ebenso erfolgreich sind EPKs als Sprache zur Modellierung von Geschäftsprozessen in einem betriebswirtschaftlichen Umfeld. Beide Sprachen prallen dann aufeinander, wenn der Umfang eines Projektes vom Grobkonzept bis zur Realisierung eines IT-Systems reicht und zu Beginn die Modellierung von Geschäftsprozessen enthält. Das Ergebnis ist ein Methodenbruch. Er findet i. a. am Ende der Fachkonzeptphase statt, vor Beginn der Phase des IV-Konzepts.

BPEL und BPMN sind noch in der Erprobung.

Diese Diskussion über den Einsatz der UML ist nötig. Aber sie diskutiert nicht den Kern der Objektorientierung. Objektorientierung heißt nicht: In welcher Sprache *rede* ich?

Objektorientierung bedeutet in der frühen Phase: *Denke* ich objektorientiert?

Unabhängig von den unterschiedlichen Anforderungen der beiden Leserkreise eines Fachkonzepts gilt: Das Fachkonzept muß

- verständlich
- widerspruchsfrei
- vollständig
- und in der IT umsetzbar sein.

Verständlichkeit erreicht man, wenn man die fachlichen Begriffe der Anwender benutzt und ihnen das neue System anhand eines Prototypen vorstellt. Ansonsten ist der beste Rat an dieser Stelle, ein einfaches und präzises Deutsch oder Englisch zu schreiben. Kurze Sätze, direkte Formulierungen, keine Aneinanderreihung von Substantiven und kein Herausstellen von OO-Termini.

Um Widerspruchsfreiheit zu gewährleisten, muß man die Begriffe schärfen, Redundanz vermeiden und unterschiedliche Ebenen der Detaillierung abgrenzen.

Für die Vollständigkeit sollte man in Gedanken einmal die wesentlichen Geschäftsvorfälle durchspielen: Von der Benutzeroberfläche über die Schnittstellen der Anwendung zu Nachbarsystemen bis zur Datenspeicherung und wieder zurück.

Und die technische Umsetzbarkeit eines Fachkonzepts beurteilt am besten ein Kollege mit Erfahrung bei der Realisierung, z.B. der für die späteren Phasen vorgesehene technische Projektleiter.

Viele dieser Ratschläge kann man umsetzen, wenn man die objektorientierte Methode anwendet - so früh wie möglich und möglichst konsequent. Wir entwickeln Methode und Vorgehen bei der objektorientierten Analyse als Antwort auf vier Fragen, die jeweils durch ein Prinzip objektorientierten Denkens beantwortet werden:

- Welche Komponenten hat das System? Diese Frage beantwortet die *Klassenbildung* (Abschnitt 3.1)

- Welche Eigenschaften der Komponenten lassen sich verallgemeinern? Hierauf antwortet das Prinzip der *Vererbung* (Abschnitt 3.2).
- In welche Beziehungen treten die Komponenten zueinander? Das legt die Spezifikation der *Wechselwirkung* fest (Abschnitt 3.3).
- Auf welchen Ebenen der Detaillierung kann man das System und seine Komponenten studieren? Hierauf antwortet das Verfahren der *Multi-Skalen-Analyse* (Abschnitt 3.4).

Frage	Prinzip
Welche Komponenten gibt es?	Klassenbildung
Welche Eigenschaften von Komponenten lassen sich verallgemeinern?	Vererbung
Welche Aufrufbeziehungen bestehen zwischen Komponenten?	Wechselwirkung
Wie vergleicht man Komponenten unterschiedlicher Detaillierung?	Multi-Skalen-Analyse

Abbildung 11: Prinzipien einer objektorientierten Analysemethode

3.1 Klassenbildung

Klassen fassen Objekte mit gleichen Eigenschaften zusammen. Es gibt drei Arten von Eigenschaften:

- Daten
- Operationen (= Funktionen)
- Lebenszyklus.

Aus Sicht der Klasse sind *Objekte* die Instanzen von Klassen, also ihre konkreten Ausprägungen. Aus Sicht der Objekte ist die Klasse eine Schablone zum Erzeugen und Verwalten gleichartiger Objekte.

Der Begriff der Klasse verallgemeinert den Begriff der Entität (genauer: Entitätstyp).

- Eine Klasse besitzt Daten genau wie eine Entität.
- Zusätzlich enthält eine Klasse aber auch alle Funktionen, die für die Bearbeitung dieser Daten nötig sind. Diese Funktionen heißen *Operationen* der Klasse.

  Daten eines Objektes sollen nur durch die Operationen der zugehörigen Klasse, aber nicht durch Operationen anderer Klassen verändert werden.
- Der *Lebenszyklus* einer Klasse besteht aus den möglichen Zuständen und Zustandsübergängen ihrer Objekte.

  Der richtige Zustand des Objektes ist in vielen Fällen die Vorbedingung, damit bestimmte Operationen auf diesem Objekt ausgeführt werden können. Und häufig verändert die Ausführung einer Operation die Datenwerte des Objektes, so daß das Objekt seinen Zustand ändert. Man sagt: Das Objekt hat einen neuen „Zustand in seinem Lebenszyklus“ erreicht.

Daten sind i. a. den Objekten zugeordnet, weil jedes Objekt für ein gegebenes Attribut seinen eigenen Wert hat. Daneben kann es weitere Daten geben, die für ein einzelnes Objekt keinen Sinn ergeben, sondern nur für die Klasse als ganze (z.B. die Anzahl aller Objekte einer Klasse). Sie heißen Klassenattribute. *Operationen* sind stets der Klasse zugeordnet. Sofern sie nicht die Klassenattribute verwalten, stehen sie aber allen Objekten der Klasse zur Verfügung. Der *Lebenszyklus* einer Klasse ist das Template für die Lebenszyklen ihrer Objekte. Der Lebenszyklus der Klasse spezifiziert die möglichen Lebenszyklen aller ihrer Objekte, die Klasse selbst durchläuft dagegen keinen Lebenszyklus.

Ein grundlegendes Prinzip, welches das Klassenkonzept vom klassischen Modulkonzept übernommen hat, ist die Trennung von „*public*“ (öffentlichen) und „*private*“ (privaten) Eigenschaften. Klassen dienen dazu, für andere Klassen im Sinne eines Client/Server Prinzips einen Dienst zu erbringen. Der Auftraggeber (Client) kennt dabei nur die public Eigenschaften seines Servers. Das sind diejenigen Eigenschaften, die der Server explizit in seiner Schnittstelle bekannt gemacht hat. Im strengen Sinne sind Daten immer private. Der Zugriff auf die Daten geschieht ausschließlich über eine Operation, und diese kann in der Schnittstelle als public angeboten werden. Im Fachkonzept wird im wesentlichen nur der öffentliche Teil der Klassen festgelegt, d.h. spezifiziert.

Die unlösbare Verbindung von Daten und Funktionen im Begriff der Klasse ist ein wesentlicher Unterschied zum konventionellen Vorgehen mit getrennten Daten- und Funktionsmodellen. Aus Sicht der Objektorientierung gehören die Daten und die Funktionen zu ihrer Bearbeitung untrennbar zusammen, es sind zwei Seiten desselben Konzepts.

> Was von Natur aus zusammengehört, das soll der Mensch nicht scheiden.

Durch die Zusammenfassung in Klassen wird die Komplexität des Gesamtsystems reduziert: Die Anzahl der Klassen ist um eine Größenordnung kleiner als die Summe aus Daten und Funktionen beim konventionellen Vorgehen.

Klassen stehen in *strukturellen Beziehungen* zueinander. Diese sind statisch und gelten für alle Objekte der betreffenden Klassen, jeweils während ihrer gesamten Lebensdauer. Die folgenden beiden strukturellen Beziehungen spielen in unserer Sicht der Objektorientierung eine besondere Rolle:

- Generalisierung und Spezialisierung („is a“, Kapitel 3.2).

Eine Klasse ist ein Spezialfall einer anderen Klasse: Die beiden Klassen *Lieferant* und *Kunde* sind Spezialfälle der Klasse *Geschäftspartner*: *Lieferant* „is a“ *Geschäftspartner*.

- Komposition („is part of“, Kapitel 3.4).

Eine Klasse ist Bestandteil einer anderen Klasse: Die beiden Klassen *Auftragskopf* und *Auftragsposition* sind Bestandteil der Klasse *Auftrag*: *Auftragskopf* „is part of“ *Auftrag* und *Auftragsposition* „is part of“ *Auftrag*. Eine Kompositionsbeziehung erzeugt eine Baumstruktur ihrer Bestandteile, jeder Bestandteil ist Teil genau eines Ganzen.

- Aggregation

Eine abgeschwächte „is part of“-Beziehung im Sinne von „wird gruppiert durch“: Eine Klasse kann als Teil mehreren Klassen zugeordnet sein, die in der Rolle des Ganzen auftreten.

Daneben gibt es die üblichen (1:n)-Beziehungen und (n:m)-Beziehungen, die aus der Datenmodellierung bekannt sind. Diese beiden Arten von Beziehungen sowie Komposition und Aggregation werden zusammenfassend auch *Assoziation* genannt.

Obige Beziehungen zwischen Klassen heißen strukturelle Beziehungen im Unterschied zu Aufrufbeziehungen. Mit Aufrufbeziehungen werden die Client/Server-Beziehung zwischen Klassen und ihren Operationen ausgedrückt.

Das Konzept der Klasse ist wie andere Prinzipien der Objektorientierung in der Programmierung entstanden. Es ist aber viel allgemeiner und gilt genau so für Klassen, die keine Implementierungsklassen sind. Sie heißen manchmal auch Typen.

Klassen, die im Rahmen des Grobkonzepts oder des Fachkonzepts spezifiziert werden, sind nie Implementierungsklassen, sondern es sind *Geschäftsklassen* mit einem betriebswirtschaftlichen Inhalt.

Man findet die *Geschäftsklassen* eines Projekts in den ersten drei Wochen des Grobkonzepts, indem man den Kunden bittet, zehn Schlüsselbegriffe zu nennen, mit denen er den gesamten fachlichen Inhalt seines Projekts beschreiben würde. Als Berater sind einem diese immer wiederkehrenden Begriffe schon mehrfach aufgefallen. Zum anderen kennt ein Berater betriebswirtschaftliche Standardklassen wie „Kunde“, „Auftrag“, „Produkt“, „Rechnung“ des Projektumfeldes und erwartet, daß sie auch in dem vorliegenden Projekt unter einem ähnlichen Namen vorkommen. Allerdings denkt der Kunde i. a. von den Attributen aus, wenn er seine Kandidaten für Geschäftsklassen aufzählt. Der Berater muß sich dann vergewissern, daß sich auch die fachliche Funktionalität des Projekts genau diesen Geschäftsklassen als ihre Verwaltungsoperationen zuordnen läßt.

3.2 Vererbung

Um verschiedene Spezialfälle zu generalisieren, bildet man für den allgemeinen Fall eine *Superklasse* und für die Spezialfälle zugehörige *Subklassen*. Die resultierende Beziehung zwischen den Klassen heißt Vererbungsbeziehung, und stellt in der einen Richtung eine Generalisierung und in der anderen eine Spezialisierung (Kapitel 3.1) dar.

Alle Eigenschaften einer Superklasse können auch ihren Subklassen zur Verfügung gestellt werden: Sie werden von der Superklasse auf die Subklassen vererbt. Beispielsweise verfügt die Superklasse *Geschäftspartner* über eine Operation "Beurteilen", die auf die Subklassen *Lieferant* und *Kunde* vererbt wird und beim Kunden die Bonitätsprüfung darstellt. Subklassen können allerdings zusätzlich weitere Operationen, Attribute und einen reichhaltigeren Lebenszyklus haben.

Das Prinzip der Vererbung entfaltet seine volle Kraft erst bei der Programmierung. Hier kann die Implementierung einer ererbten Operation von der Subklasse geändert werden, während die Außenansicht der Operation, ihre Signatur, gleich bleibt.

Vererbung ist eine transitive Eigenschaft, d.h. eine Subklasse kann selbst wiederum Superklasse für weitere Subklassen sein.

3.3 Wechselwirkung

Die Klassen des Unternehmensmodells stehen nicht isoliert zueinander, sondern ihre Objekte rufen sich wechselseitig über Operationen auf. Prozesse entstehen, wenn die Objekte temporär in Client/Server-Beziehungen miteinander treten. Der Client ruft dabei Operationen aus der Schnittstelle seines Servers auf. Diese Wechselwirkung von Klassen sind die Prozesse, und ihre Modelle sind die dynamische Komponente des Fachkonzepts. Die entstehenden Abläufe gehören zu den Geschäftsprozessen des Unternehmens (siehe Abbildung 4). Mit ihrer Modellierung erhält das Unternehmensmodell neben seinen statischen Anteilen auch eine dynamische Komponente.

Die Wechselwirkung der Klassen und die zugehörige Multi-Skalen-Analyse (Kapitel 3.4) stellt wesentlich höhere Anforderungen an die Modellierung als die statische Komponente des Unternehmensmodells. Im Zuge der Multi-Skalen-Analyse (siehe Abschnitt 3.4) wird auch die Wechselwirkung verfeinert. Dabei wird der Lebenszyklus der Klasse einer höheren Ebene als Interaktion der Lebensläufe der Komponentenklassen dargestellt. Die entscheidenden Charakteristika der Wechselwirkung sind

- Nebenläufigkeit
- und Verteiltheit

Nebenläufigkeit bedeutet, daß mehrere Aktionen unabhängig voneinander ausgeführt werden können. *Verteiltheit* besagt, daß der globale Zustand des Systems aus vielen lokalen Zuständen einzelner Objekte besteht und daß globale Zustandsänderungen eine Summe lokaler Aktionen sind.

3.4 Multi-Skalen-Analyse

Multi-Skalen-Analyse ist ein Hierarchisierungskonzept.

Bei einem Top-Down Vorgehen geschieht die Analyse des Projektinhaltes auf Ebenen sukzessiver Detaillierung. Jede Ebene hat einen anderen Maßstab der Detaillierung. Daher nennen wir dieses Vorgehen „Multi-Skalen-Analyse“.

Eine *Multi-Skalen-Analyse* hält also die Anforderungen an ein System auf mehreren Abstraktionsniveaus in unterschiedlichen Graden der Detaillierung fest. Gleichzeitig wird festgelegt, wie der Übergang zwischen den verschiedenen Ebenen der Detaillierung aussieht. In der einen Richtung heißt dieser Übergang Verfeinerung, in der anderen Vergröberung. Sukzessive Verfeinerung ist das wichtigste Verfahren, um ein System schrittweise mit wachsender Detaillierung zu verstehen.

Zwei Ebenen mit benachbarten Skalenfaktoren, die gröbere und die feinere Ebene, stehen zu einander im Verhältnis von Spezifikation (Was?) und Konstruktion (Wie?). Aus Sicht der gröberen Ebene ist die Verfeinerung „private“, nur die Klassen der gröberen Ebene mit ihren Eigenschaften und Beziehungen sind „public“.

Jede Ebene hat denselben Grad der formalen Strenge und besteht aus statischen und dynamischen Modellen. Der Übergang zwischen Modellen benachbarter Ebenen ist ebenfalls formalisiert: Im statischen Modell dient zur Verfeinerung zwischen benachbarten Ebenen die Kompositionsbeziehung zwischen Klassen (Kapitel 3.1). Die gröbere Ebene ist das Ganze, die Klassen der Verfeinerung bilden die Teile. Bei der Verfeinerung werden gleichrangig mit den Klassen auch die strukturellen Beziehungen zwischen ihnen verfeinert.

Beispiel: Die Klasse *Auftrag* wird durch das Zusammenspiel der Klassen *Auftragskopf*, *Auftragsposition*, *Auftragstext*, etc. verfeinert. Die Klassen *Auftragskopf*, *Auftragsposition* und *Auftragstext* erlauben das System auf einer feineren Ebene zu betrachten als die Vergröberung, bei der man nur den *Auftrag* als Black Box kennt.

In einem Klassen-Beziehungs-Diagramm, das die strukturellen Beziehungen zwischen Klassen zeigt, sollten nicht Klassen mit verschiedenem Detaillierungsgrad gemischt werden. Vielmehr legt man für jede gegebene Ebene eines oder mehrere separate Klassen-Beziehungs-Diagramme an. Dabei soll ein Klassen-Beziehungs-Diagramm der tieferen Ebene jeweils genau eine Klasse der nächsthöheren Ebene verfeinern. Da Kompositionsbeziehungen eine Detaillierung bedeuten, sollten sie möglichst nicht zwischen Klassen derselben Ebene angelegt werden. Vielmehr stellen Kompositionsbeziehungen gerade die Verbindung zwischen benachbarten Ebenen dar.

Bei einer Multi-Skalen-Analyse hat man als Leser die Möglichkeit, das Ergebnis an frei gewählten Stellen in einer größeren Detaillierung zu betrachten. Je nach gewähltem Skalenfaktor sieht man Details verschiedener Größe. Dabei ist das System auf dem jeweils gewählten Niveau verständlich - ohne daß man seine Details in der Verfeinerung zur Kenntnis nehmen muß.

3.5 Metamodell objektorientierten Denkens

Abbildung 12 zeigt die objektorientierten Prinzipien noch einmal in einer formalen Zusammenstellung, einem Metamodell: Das Prinzip der „Klassenbildung“ (Abschnitt 3.1) wird formalisiert durch die *Geschäftsklasse*, ihre definierenden Bestandteilen *Attribut*, *Operation* und *Zustand* sowie die *Horizontale Beziehung* zur strukturellen Verbindung von Klassen.

Das Prinzip „Vererbung“ (Abschnitt 3.2) spielt in den frühen Phasen von IT-Projekten nur eine geringe Rolle und wurde daher nicht in das Metamodell aufgenommen.

Das Prinzip „Wechselwirkung“ (Abschnitt 3.3) gehört zum dynamischen Teil des Metamodells. Das Prinzip wird durch *Geschäftsprozess* und seine definierenden Beziehungen zu *Operation* und *Zustand* widergegeben. Dabei wird die Wechselwirkung als ein Prozess formalisiert, in dem Objekte der Geschäftsklassen untereinander ihre Operationen aufrufen und dadurch einen Zustandsübergang erfahren.

Das Prinzip „Multi-Skalen-Analyse“ (Abschnitt 3.4) schließlich wird durch *Abstraktion* mit den drei Spezialisierungen *Prozessabstraktion*, *Klassenabstraktion* und *Beziehungsabstraktion* formalisiert. Eine Abstraktion ist eine Abbildung, welche jeweils alle *Teile* einer Äquivalenzrelation auf ihre Äquivalenzklasse als *Ganzes* abbildet.

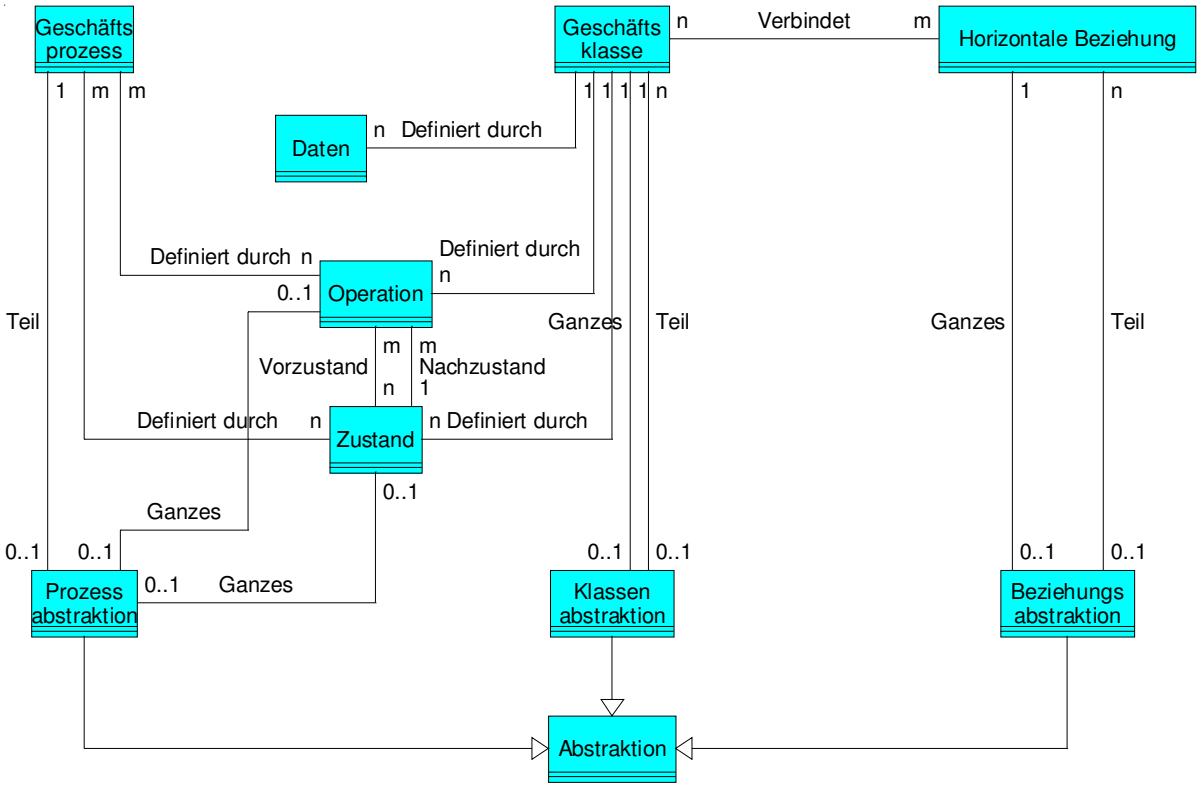


Abbildung 12: Metamodell Objektorientierung

4 Sprachen zur Unternehmensmodellierung

In diesem Kapitel stellen wir eine Reihe von Sprachen vor, die in der Praxis zur Unternehmensmodellierung eingesetzt werden können. Die Unified Modeling Language ist die einzige unter ihnen, die sich selbst als objektorientierte Sprache bezeichnet. Alle anderen Sprachen unterstützen jedoch die in Kapitel 3 eingeführten Prinzipien objektorientierten Denkens in vergleichbarem Maße.

Während die UML ihren Ausgangspunkt in der Modellierung der statischen Aspekte hat, haben Ereignisgesteuerte Prozeßketten, Petri-Netze und die Business Process Modeling Language in der Modellierung der dynamischen Aspekte ihren Ursprung.

4.1 Ereignisgesteuerte Prozeßkette (EPK)

Wertschöpfungsketten wie in Abbildung 4 werden gern als Übersichtsdarstellungen verwendet. Als solche stellen sie natürlich nur ein grobes Modell der Geschäftsprozesse dar. Eine genauere Darstellung modelliert einen Prozeß mit den beiden dualen Konzepten

- lokalen Aktionen
- und lokalen Zuständen.

Eine weitverbreitete Modellierungssprache für Geschäftsprozesse sind *Ereignisgesteuerte Prozeßketten* (EPK).

Diese Sprache beruht auf drei Sprachbestandteilen:

- Funktionen,
- Ereignissen
- und logischen Operatoren.

Die Funktionen stellen die Aktionen des Prozesses dar. Die Ereignisse sind sowohl die Zielzustände der Funktionen als auch ihre auslösenden Trigger. Mit den logischen Operatoren lassen sich Prozeßverzweigungen wie Alternativen oder nebenläufig ablaufende Teilprozesse modellieren.

Die Besonderheit der EPK ist die explizite Modellierung von Alternativen und Nebenläufigkeiten durch die logischen Operatoren. Eine Alternative wird durch einen xor-Operator geöffnet und wieder geschlossen, die Nebenläufigkeit zweier Teilprozesse durch ein Paar von und-Operatoren. Die oder-Alternative kann nach der Formel

$$x \text{ oder } y \Leftrightarrow (x \text{ und } y) \text{ xor } (x \text{ xor } y)$$

auf die xor-Alternative und die Nebenläufigkeit zurückgeführt werden (Abbildung 13).

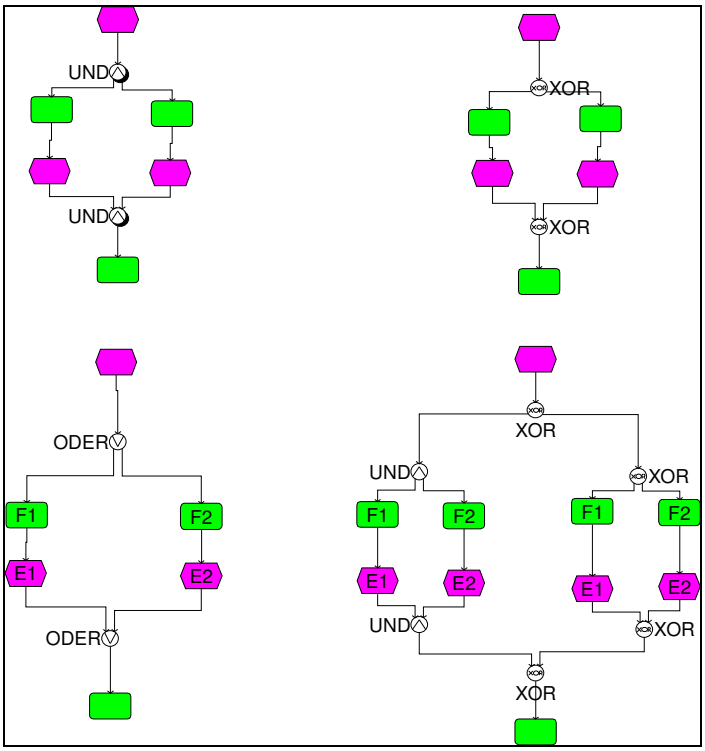


Abbildung 13: Nebenläufigkeit und Alternativen

Neben den in Abbildung 13 gezeigten in Paaren auftretenden Alternativen gibt es auch Situationen mit nicht-paarigen Alternativen. In solchen Fällen kann es schwierig sein, eine wohlgeformte EPK wie in Abbildung 14 von einer nicht wohlgeformten EPK wie in Abbildung 15 zu unterscheiden.

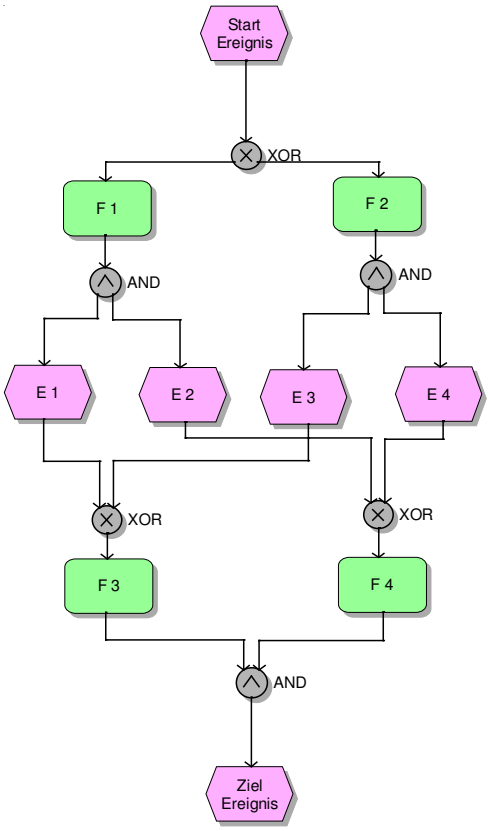


Abbildung 14: Wohlgeformte EPK

Durch Vertauschung von XOR- und AND-Konnektoren entsteht eine EPK, die nicht wohlgeformt ist: Z.B. erzeugt das gemeinsame Eintreten der Ereignisse E1 und E4 einen Deadlock, in dem keine der beiden Funktionen F3 oder F4 aktiviert ist.

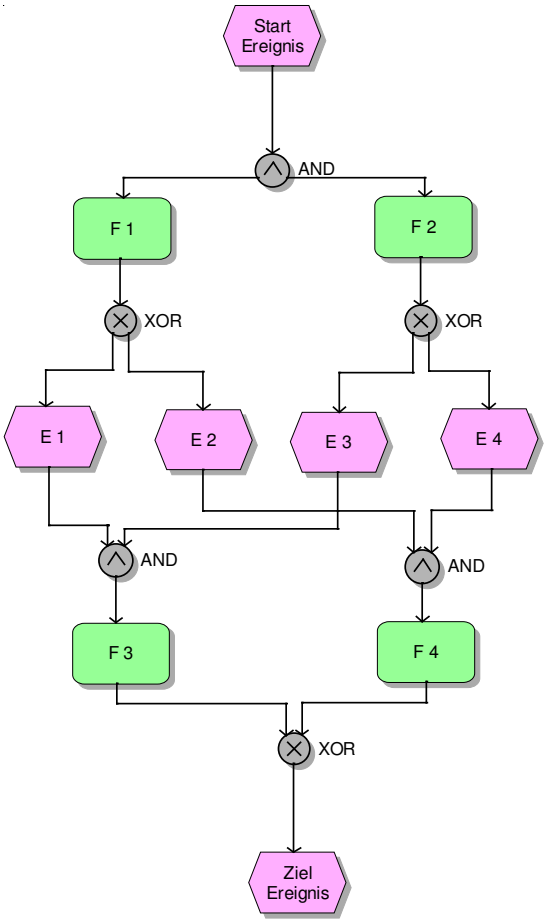


Abbildung 15: Nicht wohlgeformte EPK

EPKs sind Bestandteil der Methode ARIS (Architektur integrierter Informationssysteme) zur Unternehmensmodellierung. Man kann EPKs als eine objektorientierte Modellierungssprache einsetzen und damit das Prinzip der Klassenbildung (Abschnitt 3.1) unterstützen: Sobald man sich über die am Prozeß beteiligten Geschäftsklassen klar geworden ist, werden sie einem Klassendiagramm zusammen mit ihren strukturellen Beziehungen festgehalten, siehe Abbildung 5.

Außerdem wird jede Funktion der EPK einer dieser Geschäftsklasse als Operation zugeordnet. Ebenso werden die Ereignisse der EPK jeweils einer Geschäftsklasse zugeordnet, als Zustand im Lebenszyklus dieser Geschäftsklasse. Jede Geschäftsklasse einer objektorientierten EPK wird durch eine Klassentabelle spezifiziert. Sie sammelt aus den Prozeßmodellen alle Operationen dieser Klasse und alle Ereignisse. Dabei stellen die Ereignisse die möglichen Werte des Attributes *Status* dar. Das Ergebnis sind objektorientierte EPKs, siehe Abbildung 6, gemäß dem Prinzip der Wechselwirkung (Abschnitt 3.3).

Abbildung 16 zeigt die Klassentabelle der Klasse *Produkt*, Abbildung 17 diejenige der Geschäftsklasse *Teil*. Die Klassentabellen können in einem späteren Schritt durch weitere Attribute der Klasse erweitert.

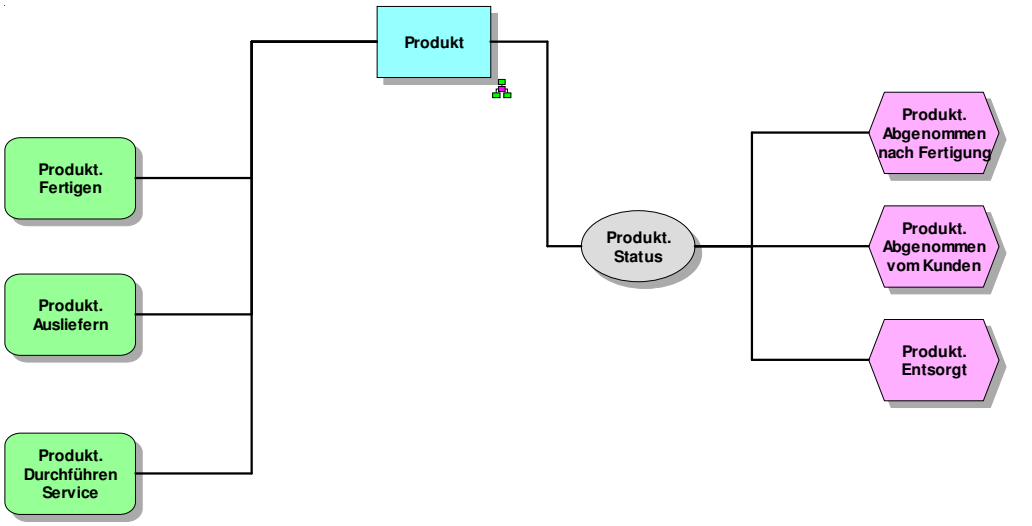


Abbildung 16: Klassentabelle der Geschäftsklasse Produkt

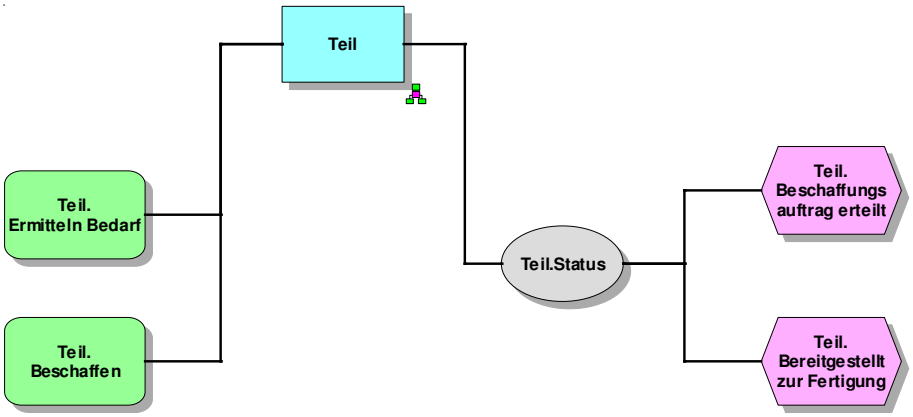


Abbildung 17: Klassentabelle der Geschäftsklasse Teil

EKPs unterstützen zu Teilen auch das Prinzip der Multi-Skalen-Analyse (Abschnitt 3.4): Dabei wird eine Funktion der gröberen Ebene durch einen ganzen Prozeß auf der feineren Ebene verfeinert. Die vor- und nachgelagerten Ereignisse der Funktion werden auf die untere Ebene übernommen, oder durch eine Kombination von Ereignissen verfeinert. Eine vollständige Ereignisverfeinerung, bei der auch ein Ereignis im Sinne eines Zustands durch einen ganzen Prozeß verfeinert wird, ist nicht allerdings nicht möglich.

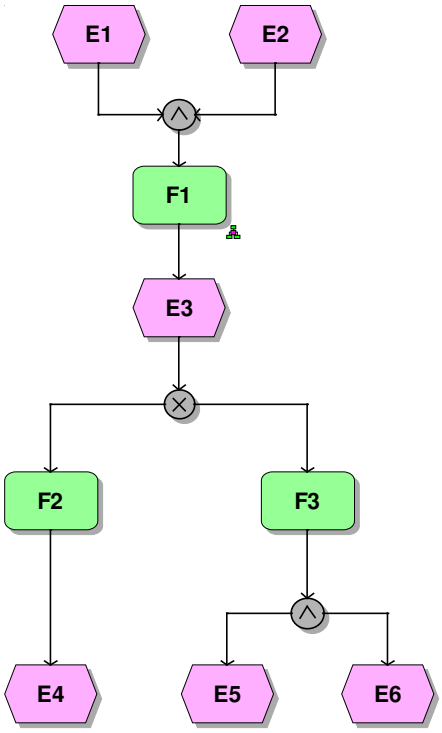


Abbildung 18: Prozeß der gröberen Ebene

Die Verfeinerung der Funktion „F1“ aus Abbildung 18 zeigt die folgende Abbildung 19.

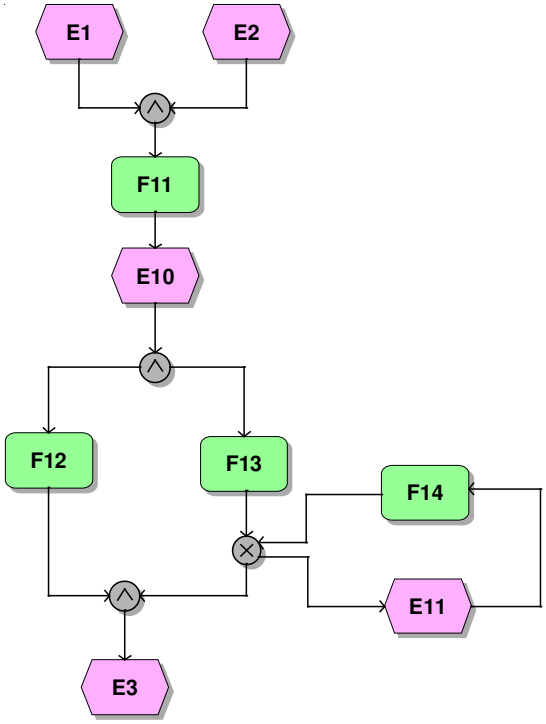


Abbildung 19: Verfeinerung der Funktion F1 aus Abbildung 18 auf der feineren Ebene

Gemäß ARIS kann die Unternehmensmodellierung mit der Modellierung der Ziele beginnen. Als graphisches Element stellt die Methode dafür das „Zieldiagramm“ zur Verfügung. Ziele können quantifiziert und ebenfalls hierarchisiert werden. Anschließend lassen sich Prozesse und ihre Funktionen sich den Zielen zuordnen, welchen sie dienen.

Die i. a. hierarchische Aufbauorganisation des Unternehmens läßt sich ebenfalls in Form von „Organigrammen“ modellieren. Die Rollen der Organisationseinheiten lassen sich den Funktionen der EPKs zuordnen, die sie verantworten oder ausführen.

Außerdem läßt sich jeder Funktion das Anwendungssystem zuordnen, das diese Funktion unterstützt. Auf diese Weise läßt sich auch der IT-Bebauungsplan des Unternehmens mit den Mitteln von ARIS abbilden.

4.2 Unified Modeling Language (UML)

Die überwiegende Zahl objektorientierter Fachkonzepte verwendet in ihren formalisierten Abschnitten die Spezifikationssprache UML (Unified Modeling Language). Diese Sprache wurde von der OMG (Object Management Group) als Standard anerkannt, gegenwärtig (April 2007) in der Version 2.0.

Die wesentlichen Sprachkonstrukte der UML sind eine Reihe von Diagrammtypen, welche entweder die statische Struktur oder das Verhalten des Modells abbilden. Für die Phase Fachkonzept bietet sich eine Auswahl aus den Diagrammtypen von Abbildung 27 an. Wir erläutern daher einige UML-Diagrammtypen an einem Praxisprojekt im Umfeld des Geschäftsprozeß „Faktura“.

Die UML wurde von den drei Autoren Booch, Jacobson und Rumbaugh entwickelt. Jeder brachte einen Teil seiner bevorzugten Modellierungskonzepte ein. Von Anfang geschah die Entwicklung der UML auch unter kommerziellen Gesichtspunkten mit dem Ziel der Marktführerschaft. So gründeten die drei Autoren die Firma Rational Inc., die auch ein entsprechendes UML-Tool vertreibt.

Verbreitete Modellierungstools, welche die UML unterstützen, sind Rose (Rational), Together (Borland), StP (Aonix) oder ARIS UML-Designer (IDS Scheer AG).

4.2.1 UML Klassendiagramm

Ein Klassendiagramm zeigt die Klassen des Systems und ihre strukturellen Beziehungen, siehe Abschnitt 3.1. Klassendiagramme gehören zur statischen Sicht des Modells. Sie zeigen die an der Wechselwirkung beteiligten Klassen, aber nicht den Ablauf der Wechselwirkung.

Jede Klasse eines Klassendiagramms kann mit einer Klassentabelle hinterlegt werden, welche die Attribute und Operationen der Klasse definiert. Die Klassentabelle kann in das Klassendiagramm eingeblendet werden.

UML erlaubt die verschiedenen Arten struktureller Beziehungen zwischen Klassen zu kennzeichnen:

- Assoziation
- Generalisierung
- Aggregation
- Komposition

Alle Beziehungen zwischen zwei Klassen tragen eine Kardinalität. Sie legt fest, wie viele Objekte aus beiden Klassen an der Beziehung teilnehmen.

Ein Vorgehen gemäß der Multi-Skalen-Analyse von Abschnitt 3.4 zeigt Abbildung 20 bei der Detaillierung einer Preisliste, die als Geschäftsklasse „Preisblatt“ aufgefaßt wird. Alle gezeigten Klassen mit dem Präfix „Preisblatt“ verfeinern die gegebene Klasse „Preisblatt“. Alle Klassen mit einem anderen Präfix gehören nicht zum Preisblatt, sondern verfeinern eine benachbarte Geschäftsklasse. Sie werden in diesem Bild gezeigt, um die Beziehungen vom Preisblatt zu dieser Nachbarklasse zu detaillieren.

Bei der Fortsetzung der Modellierung in den späteren Phasen und bei der Realisierung wird die Klasse „Preisblatt“ zu einem Package, das in Form eines Teilbaums die Detailklassen und ihre Implementierungsklassen enthält.

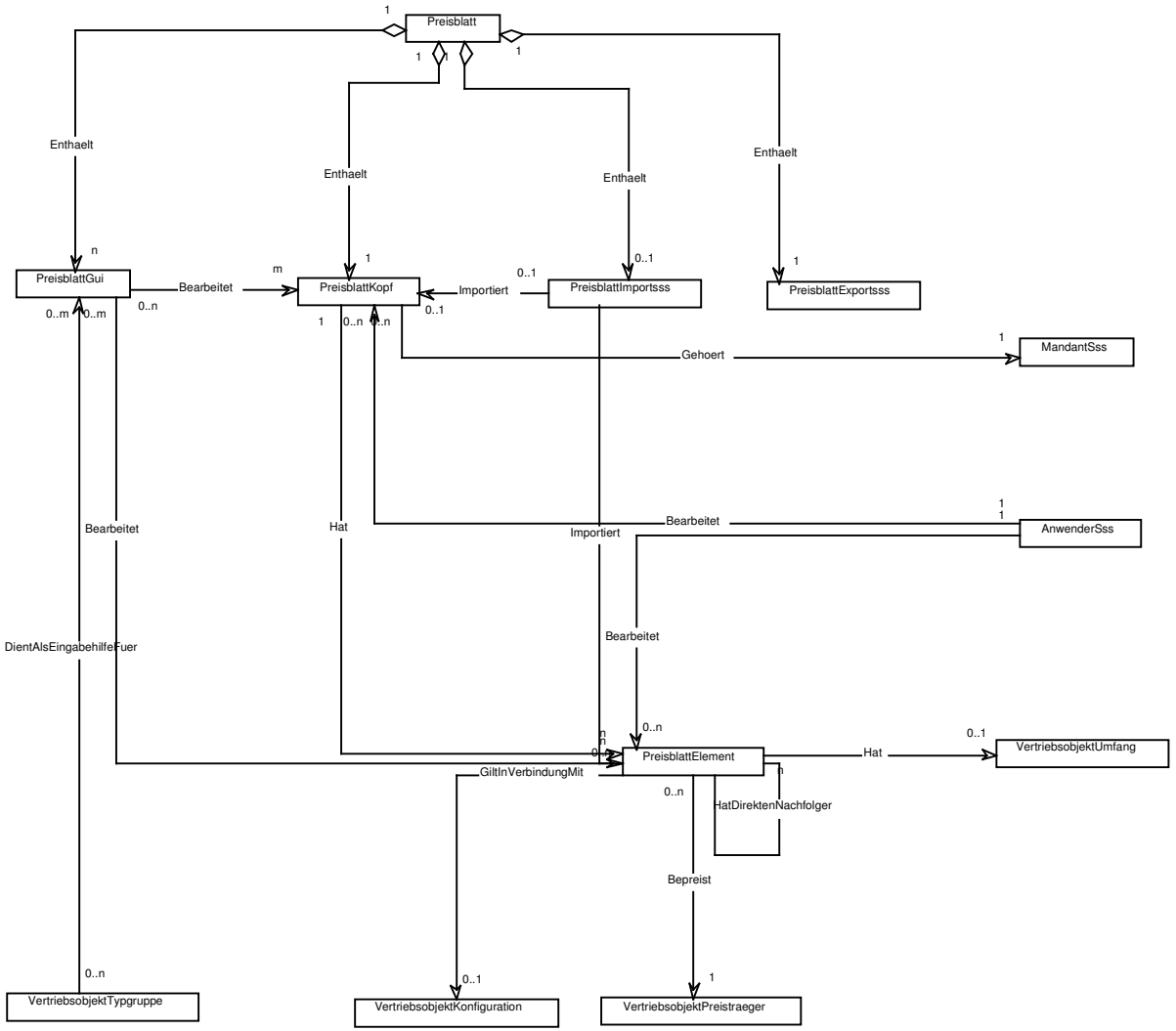


Abbildung 20: UML Klassendiagramm zur Detaillierung der Klasse „Preisblatt“

Hinweis. Anders als Entitäten haben Klassen keinen fachlichen Primärschlüssel, sondern nur eine nichtfachliche ID. Daher können UML-Tools erst nach einer entsprechenden Erweiterung zur Datenmodellierung genutzt werden.

4.2.2 UML Zustandsdiagramm

Für die Modellierung des Lebenszyklus einer Klasse bietet die UML Zustandsdiagramme an. Sie reichen vom einfachen Fall eines endlichen Automaten bis zu Statecharts mit der Möglichkeit, verteilte Zustände durch Hierarchisierung auszudrücken. In der Praxis kommerzieller Projekte bewährt sich eine Beschränkung auf endliche Automaten.

Ein Zustandsdiagramm zeigt Zustände und Zustandsübergänge, siehe Abbildung 21 für das Zustandsdiagramm der in Abbildung 20 dargestellten Klasse „Preisblatt“. Die Übergänge zwischen zwei Zuständen werden durch die Operationen dieser Klasse hervorgerufen. Allerdings gibt es auch Situationen, in denen ein Operationenaufruf den Zustand unverändert läßt. Zustandsübergänge können durch die Angabe eines auslösenden Ereignis detailliert werden.

Zusammen mit der Klassentabelle vervollständigt das Zustandsdiagramm die Spezifikation der Klasse als einer Gesamtheit aus Daten, Operationen und Lebenszyklus.

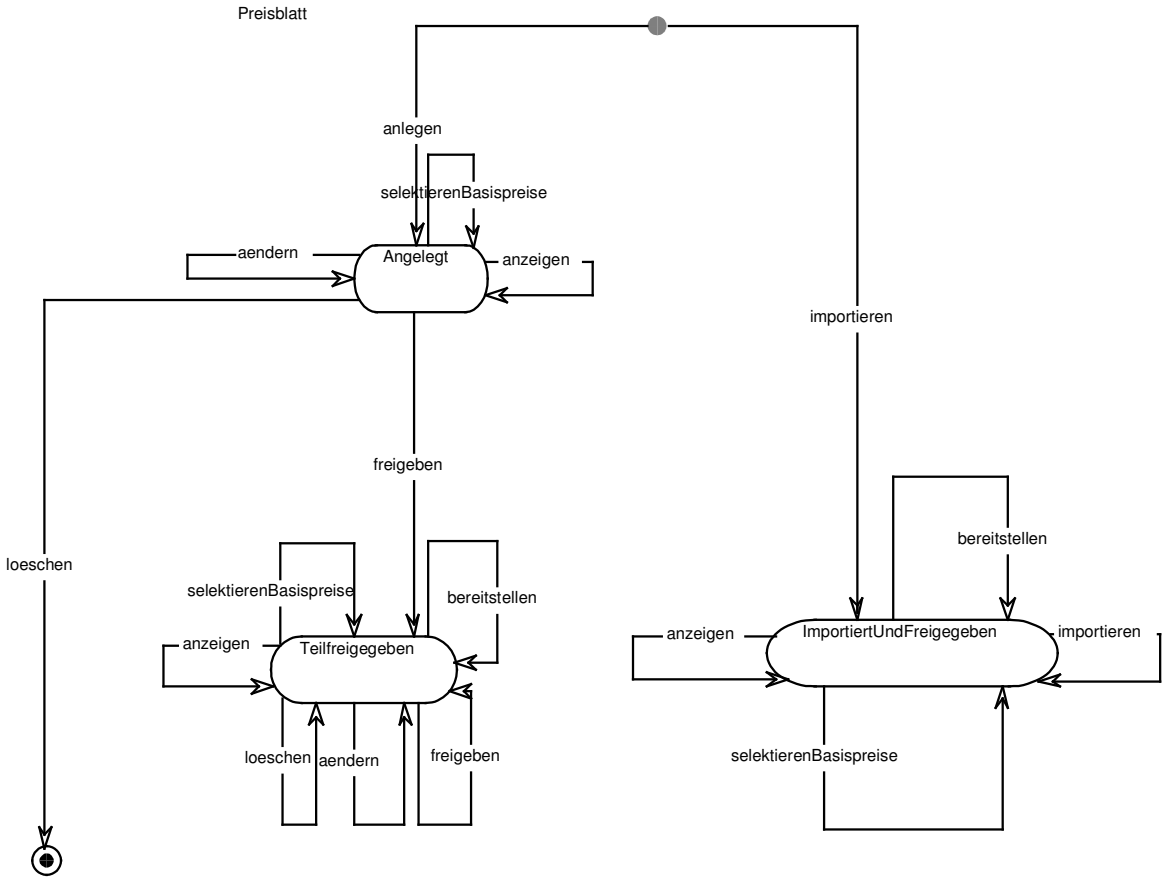


Abbildung 21: UML Zustandsdiagramm der Klasse „Preisblatt“

4.2.3 UML Usecase-Diagramm

Ein Usecase-Diagramm besteht aus einer Menge von Usecases und ihren Aktoren.

Im Sinne der Klassenbildung von Abschnitt 3.1 sind Usecases die Operationen der Geschäftsklassen einer bestimmten Detaillierungsebene. Sie sind daher schon in den Klassentabellen identifiziert. Die wesentliche Aussage der Usecasediagramme ist die Zuordnung von Aktoren zu einem Usecase.

Usecasediagramme sind die einzige Art von UML-Diagrammen im Fach- oder Grobkonzept, die ohne Einschränkung von den Anwendern verstanden und geschätzt werden. Anwender denken in fachlichen Rollen wie Preisblattverwalter oder Serviceberater. Diese Rollen legen ihre Aufgaben fest. Daher wollen die Anwender wissen, mit welchen Usecases das neue System ihre Rolle unterstützt und welche Berechtigungen an ihre Rollen geknüpft sind.

Wir empfehlen, auf die sorgfältige Zuordnung von Aktoren und Usecases großen Wert zu legen und diese Beziehung in Form von Usecasediagrammen zu zeigen. Dabei kann man entweder pro Aktor ein Usecasediagramm mit allen Usecases dieses Aktors anlegen. Oder man zeigt pro Geschäftsklasse ein Usecasediagramm mit allen Usecases dieser Geschäftsklasse und ihren Aktoren. Ein Beispiel für das letzte Vorgehen ist Abbildung 22 mit den Usecases der in Abbildung 20 gezeigten Klasse „Preisblatt“.

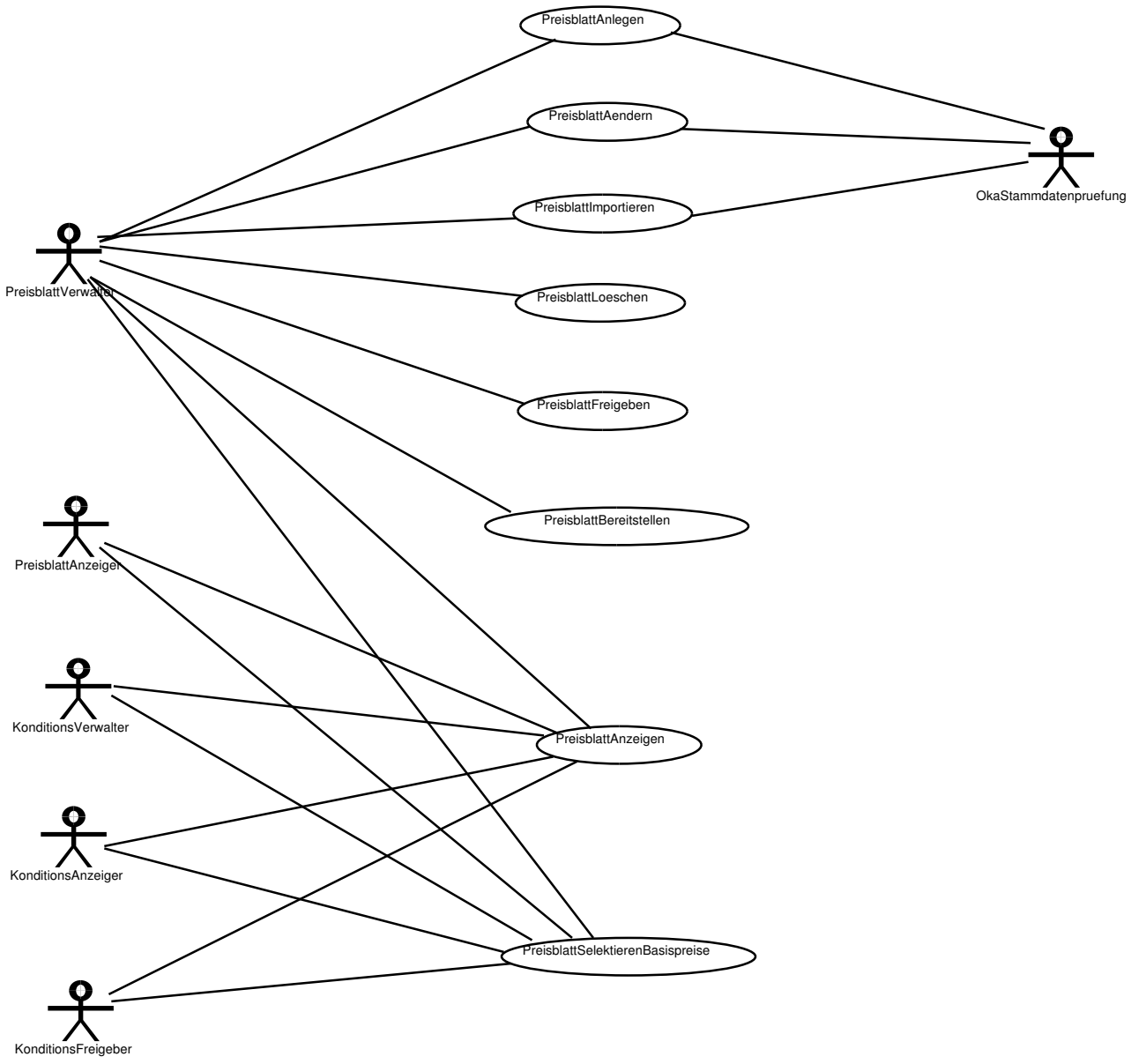


Abbildung 22: UML Usecase Diagramm der Klasse „Preisblatt“

Neben der Beziehung zwischen Aktor und Usecase bietet ein Usecasediagramm auch die Möglichkeit, zwischen zwei Usecases eine Beziehung mit der Semantik „Benutzt (uses)“,

oder „Erweitert (extends)“. Dabei stellt die „Benutzt“-Beziehung den Aufruf eines Subusecase durch einen übergeordneten Usecase dar. Eine „Erweitert“-Beziehung stellt eine optionale, zusätzliche Funktionalität zu einem Usecase dar.

Wir empfehlen, diese Art von Beziehungen restriktiv einzusetzen, um nicht an dieser Stelle des Fachkonzepts bereits eine Modularisierung von Funktionalität vorzunehmen. Als sinnvolles Beispiel aus der Praxis kennen wir z.B. die Auslagerung von Selektionsfunktionalität in einen Subusecase, der von mehreren Usecases mit „Benutzt“ aufgerufen wird. Ein Subusecase sollte keine eigenen Aktoren zeigen, da er von den Aktoren der übergeordneten Usecases benutzt wird.

4.2.4 UML Sequenzdiagramm

Ein Sequenzdiagramm zeigt das Zusammenspiel verschiedener Klassen durch wechselseitige Operationenaufrufe. Sequenzdiagramme unterstützen daher das Prinzip der Wechselwirkung, cf. Abschnitt 3.3. Die Operationenaufrufe können als Nachrichten angesehen verstanden werden, mit denen die Objekte der einzelnen Klassen Dienste von einander anfordern.

Die Wechselwirkung wird als linearer Ablauf dargestellt, bei dem die Zeit von oben nach unten abläuft und ein Schritt auf den anderen folgt.

Sequenzdiagramme können verwendet werden, um den fachlichen Ablauf zu modellieren, mit dem ein gegebener Usecase schrittweise abgearbeitet wird. Die Sequenz wird i. a. durch ein oder mehrere Ereignisse an der Benutzerschnittstelle ausgelöst und endet mit einem Ergebnis in der Datenbank oder einer Ausgabe an der Benutzerschnittstelle. Dazwischen findet eine Reihe von Operationenaufrufen der beteiligten Klassen statt.

Beispiel (siehe Abbildung 23): Wenn ein Preisblatt Verwalter im Menu den Dialogschritt *Preisblatt.Ändern* aufruft, so wird hieraus eine Reihe von Aufrufen von Klassen einer tieferen Ebene erzeugt. Z. B. werden die entsprechenden Operationen der Klassen *PreisblattKopf* und *PreisblattElement* aufgerufen.

Ein Sequenzdiagramm zeigt nicht die Vielfalt der Ausprägungen eines Usecases, sondern nur einen einzige Ablauf. Ein Sequenzdiagramm befindet sich daher auf der Instanzenebene, nicht auf der Typebene:

$$\frac{Sequenzdiagramm}{Aktivitätsdiagramm} = \frac{Instanz}{Typ}.$$

Daher modelliert ein Sequenzdiagramm einen linearen Ablauf ohne Alternativen, eventuelle Nebenläufigkeiten sind linearisiert. Der Versuch, Schleifen, Nebenläufigkeit oder Alternativen in einem Sequenzdiagramm zu modellieren, ist ein Versuch mit unzureichenden Mitteln. Für die Modellierung dieser Prozessprimitiven ist das Aktivitätsdiagramm, siehe Abschnitt 4.2.5 besser geeignet.

Bei dem Erstellen von Sequenzdiagrammen ist darauf zu achten, daß alle Diagramme denselben Grad von Detaillierung zeigen und in dieselbe Tiefe der Schichten der Systemarchitektur vorstoßen. Sequenzdiagramme werden seit langem in der Modellierung von Embedded Systems unter dem Namen „Message Sequence Charts“ eingesetzt.

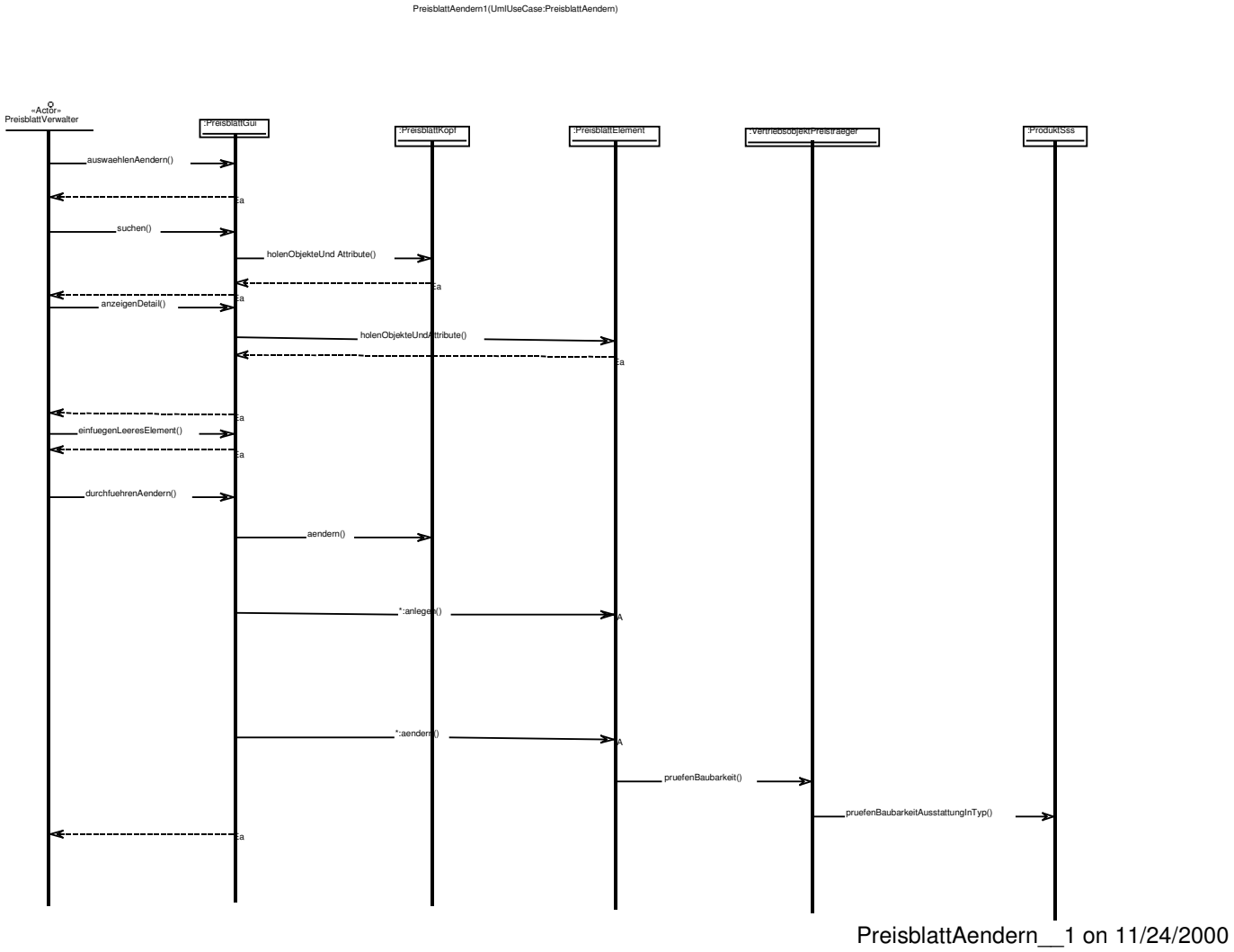


Abbildung 23: UML Sequenzdiagramm des Usecase „Preisblatt.ändern“

4.2.5 UML Aktivitätsdiagramm

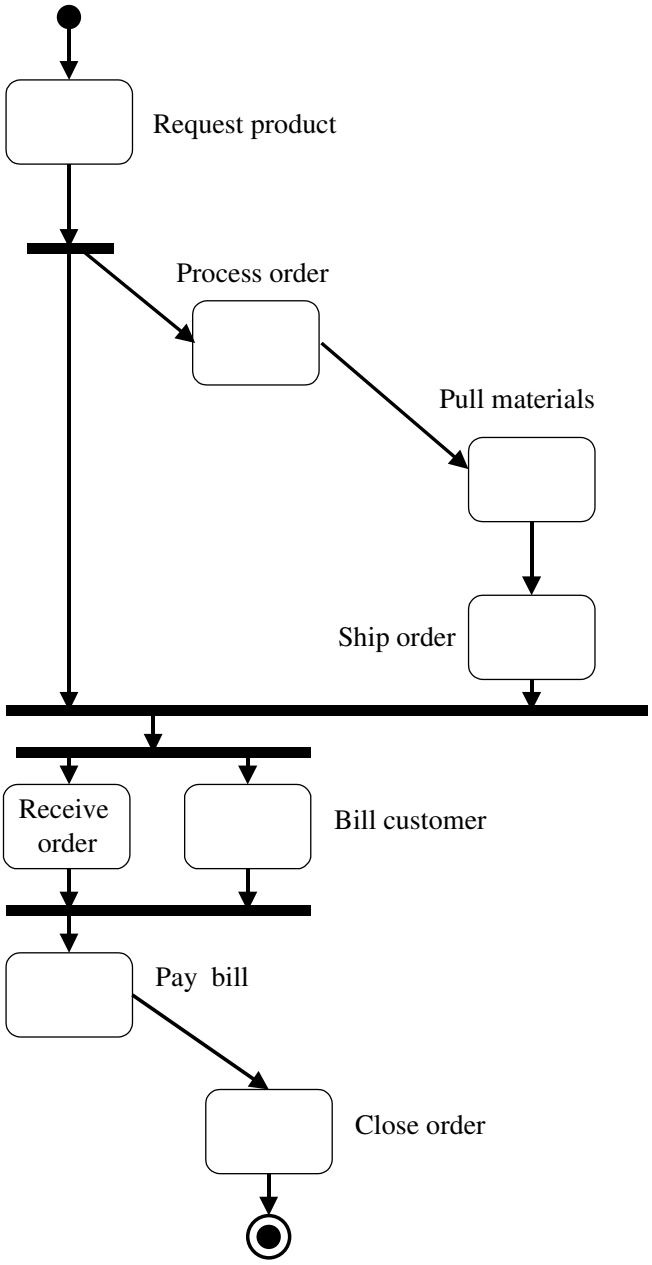


Abbildung 24: UML Aktivitätsdiagramm

Mit Aktivitätsdiagrammen lassen sich nicht nur sequentielle, sondern auch nebenläufige und alternative Abläufe darstellen. Denn Aktivitätsdiagramme bieten Konstrukte zur Modellierung von fork und join von Prozessen (Abbildung 24) und für die Modellierung von Alternativen.

Hinweis. Abbildung 24 stammt nicht aus einem Praxis-Projekt, sondern aus der Literatur [BRJ1999]. Es ist nicht klar, was die Synchronisierung der beiden Teilprozesse bedeutet, die aus der Aktivität „Request product“ entspringen.

In unserer Sicht leisten Aktivitätsdiagramme nicht viel mehr als die seit langem unter dem Namen „Flußdiagramm“ bekannten Bilder. Aktivitätsdiagramme bieten keine saubere Tren-

nung von Zuständen und Aktionen. Ein Indiz hierfür ist die Charakterisierung des wesentlichen Modellierungselementes des Aktivitätsdiagramms als „action state“.

4.2.6 UML Komponentendiagramm

Mit einem Komponentendiagramm läßt sich die funktionale Sicht der IT-Architektur einer Applikation. zeigen. i. a. handelt es sich um eine Schichtenarchitektur.

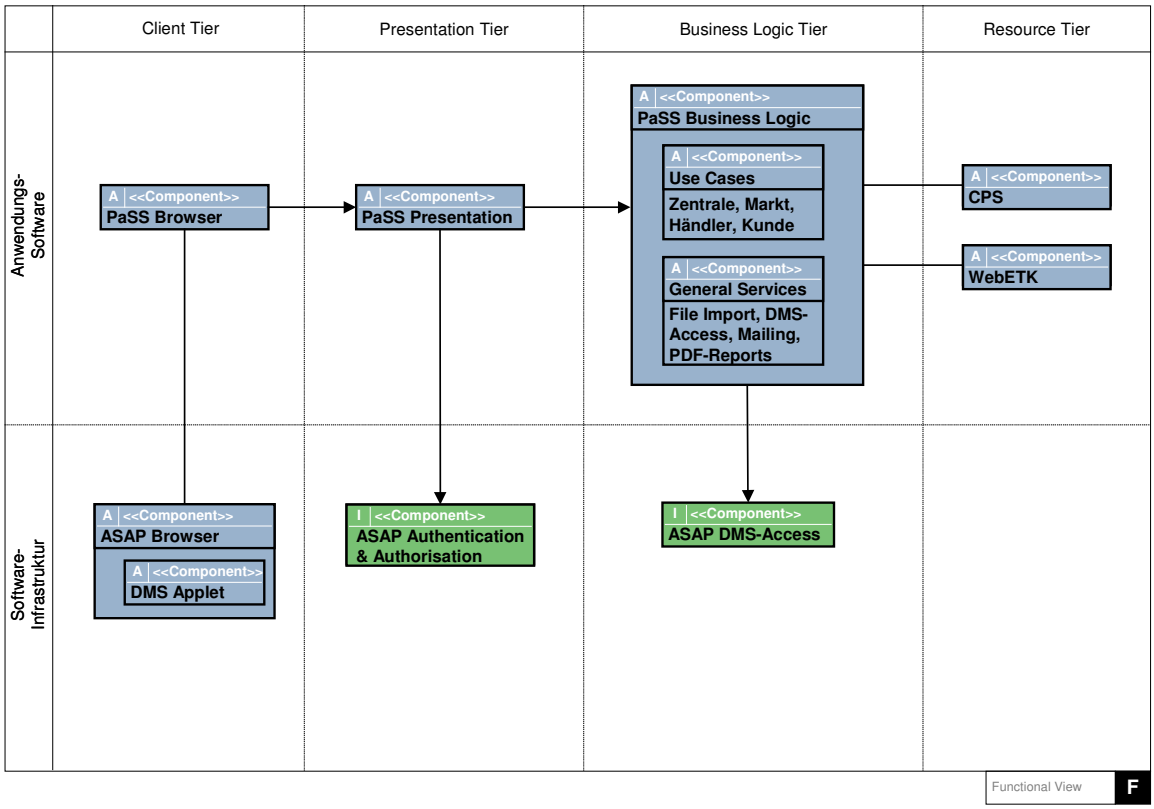


Abbildung 25: UML Komponentendiagramm einer Applikation

4.2.7 UML Verteilungsdiagramm

Ein Verteilungsdiagramm zeigt die Zusammenfassung der Komponenten der Applikation zu Deployment Units. Diese Deployment Units werden den jeweiligen Servern zugeordnet, auf denen sie laufen.

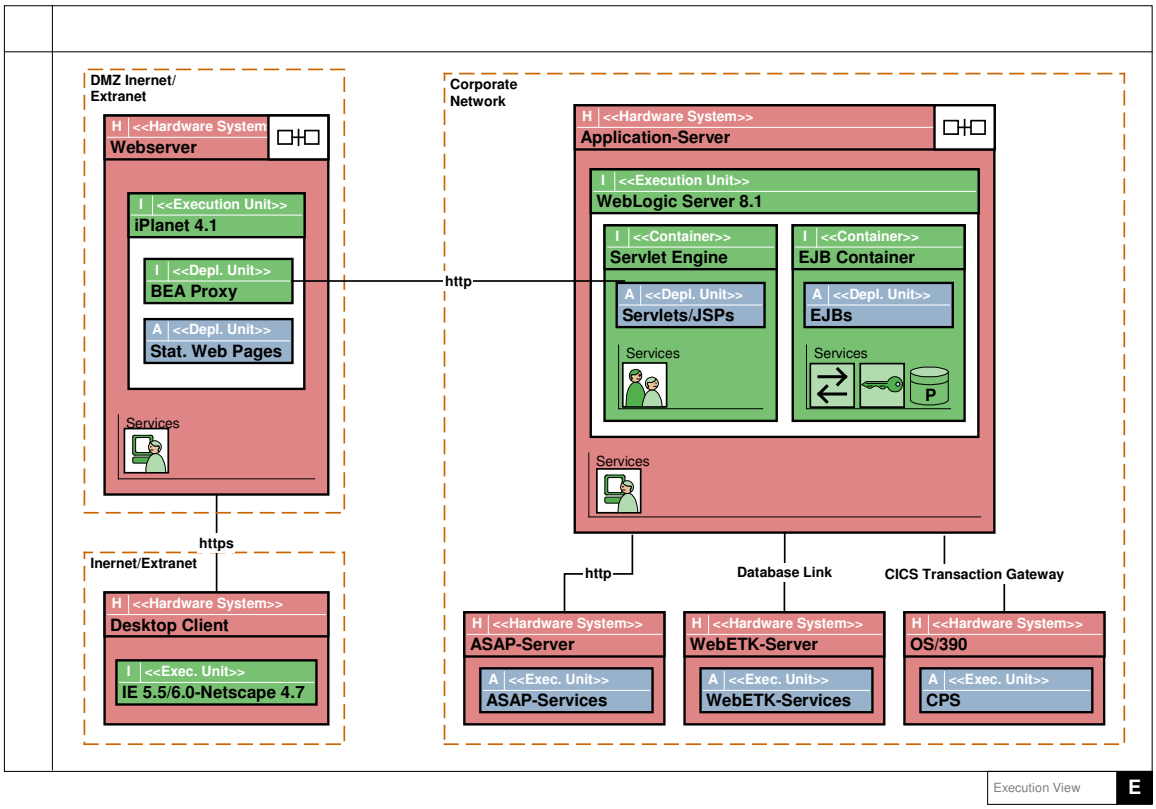


Abbildung 26: UML Verteilungsdiagramm einer IT-Applikation

4.2.8 Tabellarische Übersicht der Diagrammtypen

Die folgende Tabelle zeigt alle Diagrammtypen der UML 2.0 nach der in der UML üblichen Klassifizierung. Die Spalte „Zweck“ stammt aus [JRH2004] und stellt einen ersten Anhaltspunkt dar.

Diagrammtyp	Diagrammname	Zweck
Strukturdiagramm		
	Klassendiagramm	Wie sind Daten und Verhalten meines Systems im Detail strukturiert?
	Paketdiagramm	Wie kann ich mein Modell so darstellen, dass ich den Überblick bewahre?
	Objektdiagramm	Wie sieht ein Schnappschuß meines Systems zur Ausführungszeit aus?
	Kompositionsstrukturdiagramm	Wie sind die einzelnen Architekturkomponenten strukturiert und wie spielen sie zusammen?
	Komponentendiagramm	Wie ist mein System strukturiert und wie werden diese Strukturen erzeugt?
	Verteilungsdiagramm	Wie werden die Komponenten des Systems zur Laufzeit verteilt?
Verhaltensdiagramm		
	Use-Case-Diagramm	Was soll mein geplantes System eigentlich leisten?
	Aktivitätsdiagramm	Wie realisiert mein System ein bestimmtes Verhalten?
	Zustandsautomat	Wie verhält sich das System in einem bestimmten Zustand bei gewissen Ereignissen?
	Sequenzdiagramm	Wie läuft die Kommunikation in meinem System ab?
	Kommunikationsdiagramm	Welche Teile einer komplexen Struktur arbeiten wie zusammen, um eine bestimmte Funktion zu erfüllen?
	Timing-Diagramm	Wann befinden sich verschiedene Interaktionspartner in welchem Zustand?
	Interaktionsübersichtsdiagramm	In welcher Reihenfolge und unter welchen Bedingungen finden Interaktionen statt?

Abbildung 27: Diagrammtypen der UML 2.0

4.2.9 UML als Synonym für objektorientierte Modellierung?

UML ist ein Synonym für objektorientierte Modellierung, solange man nur auf den Markt kommerzieller Projekte, Tools und Zeitschriften schaut. Im kommerziellen Bereich wird heute kein objektorientiertes Projekt ohne UML gestartet.

Das Bild ändert sich in dem Augenblick, in welchem man seinen Horizont um die wissenschaftliche Seite des Software Engineering erweitert. Es gibt zahlreiche Forschungsprojekte mit anderen objektorientierten Modellierungssprachen, die bekannteste unter ihnen ist die Sprache gefärbter Petri Netze. Im Vergleich mit einer prozeßbetonten Sprache zeigen sich dann auch die Schwachstellen der UML:

UML als Modellierungssprache umfaßt die bewährten Prinzipien der objektorientierten Datenmodellierung. Diese Modellierung des *statischen* Aspekts eines Systems ist seit vielen Jahren erprobt und hat sich in der Praxis bewährt.

Die Dynamik eines Unternehmens spiegelt sich in seinen Geschäftsprozessen, die durch einen hohen Grad von Nebenläufigkeit gekennzeichnet sind. Dasselbe gilt auf der Systemebene für eine verteilte Architektur. Die Leistungsfähigkeit der UML bei der Modellierung der *dynamischen* Seite eines Systems ist umstritten.

Sequenzdiagramme erzwingen hier eine möglicherweise künstliche Serialisierung nebenläufiger Abläufe (Darstellung von „Concurrency“ durch „Interleaving“). Zustandsdiagramme erlauben in der Form von endlichen Automaten keine Modellierung verteilter Zustände, oder sie besitzen in der Form von *State Charts* eine schwer prüfbare Semantik. Aktivitätsdiagramme wiederum erlauben die Darstellung von Nebenläufigkeit und eine eingeschränkte Form von Alternativen – allerdings keine Zustände. Die Verteilung des dynamischen Aspekt eines Systems auf die genannten drei Arten von Diagrammen ergibt einen hohen Grad von Redundanz. Daraus resultiert das Problem, die einzelnen Darstellungen konsistent zu halten. Außerdem braucht man natürlich Modelle, in denen sowohl Zustände wie Aktionen spezifiziert werden.

Der für betriebswirtschaftlichen Anwendungen so wichtige Bereich der Geschäftsprozeßmodellierung wird durch die Aktivitätsdiagramme der UML unzureichend abgedeckt. Dieser Diagrammtyp stellt einen Rückschritt dar gegenüber einer jahrelang erfolgreich eingesetzten Prozessmodellierungssprache wie z.B. EPKs, gar nicht zu reden von der Ausdrucksmächtigkeit und Präzision von Petri-Netzen.

Die Anwender einer neuen Applikation, und das sind die Kunden und Auftraggebers eines Projektes, wollen sich i. a. nicht erst in eine formale Sprache einarbeiten müssen, bevor sie das Fachkonzept lesen. Daher sollte die UML im Fachkonzept restriktiv eingesetzt werden.

Erst das IT-Konzept, das sich ausschließlich an IT-Fachleute richtet, ist der geeignete Ort zum Einsatz einer formalen Sprache wie z.B. UML.

4.3 Petri-Netz

Petri-Netze sind eine formale Sprache zur Modellierung und theoretischen Untersuchung von verteilten und nebenläufigen Systemen. Ein System heißt *verteilt*, wenn sich sein globaler Zustand aus lokalen Zuständen seiner Komponenten zusammensetzt und wenn seine globalen Zustandsänderungen durch deren lokale Aktionen bewirkt werden. Es heißt *nebenläufig*, wenn es Zustände besitzt, in denen mehrere lokale Aktionen unabhängig voneinander ausgeführt werden können.

Bei der Analyse von Petri-Netzen können die mathematischen Verfahren der Linearen Algebra und der Theorie der linearen Ungleichungen eingesetzt werden. In der Praxis kommerzieller Projekte haben sich Petri-Netze als Modellierungssprache jedoch nicht durchgesetzt.

Jedoch: Immer dann, wenn es auf die korrekte Modellierung oder Analyse eines schwierigen dynamischen Problems ankommt, empfiehlt es sich, zunächst eine Lösung in der Sprache der Petri Netze zu finden. Im zweiten Schritt kann der Berater die gefundene Lösung für den Kunden in die Sprache der EPKs oder in eines der dynamischen UML-Diagramme übersetzen. Bei dieser Übersetzung müssen Kompromisse mit der schwächeren Ausdrucksfähigkeit dieser Sprachen in Kauf genommen werden.

4.3.1 Ungefärbtes Petri-Netz

Ein ungefärbtes (Synonym: gewöhnliches) Petri-Netz $PN = (N, \mu)$ ist ein gerichteter bipartiter Graph N (Synonym: Netz) zusammen mit einer Anfangsmarkierung μ. Die beiden Arten von Knoten heißen „Stelle“ bzw. „Transition“. Auf den Stellen können eine oder mehrere Marken liegen.

- Stellen modellieren die möglichen lokalen Zustände des Systems,
- Transitionen modellieren die möglichen lokalen Aktionen des Systems.
- Das Bestehen eines lokalen Zustandes wird durch die Markierung der zugehörigen Stelle ausgedrückt.

Damit zeigt die Markierung des Petri-Netzes den aktuellen globalen Systemzustand. Dieser Zustand kann sich nach folgender Schaltregel verändern:

Eine Transition ist *aktiviert*, wenn auf jeder ihrer Vorstellen mindestens eine Marke liegt. Eine aktivierte Transition kann *schalten*. Beim Schalten entfernt sie auf jeder ihrer Vorstelle genau eine Marke und erzeugt auf jeder ihrer Nachstellen genau eine Marke.

Das Schalten einer aktivierten Transition führt die Markierung vor dem Schalten in eine Folgemarkierung nach dem Schalten über. Eine Markierung heißt *erreichbar*, wenn sie von der Anfangsmarkierung aus durch eine endliche Folge von Schaltvorgängen als Folgemarkierung entsteht.

Ein Petri-Netz heißt *beschränkt*, wenn es eine Konstante K gibt, so daß jede erreichbare Markierung auf jeder Stelle höchstens K Marken trägt. Im Falle $K = 1$ heißt das Petri-Netz *sicher*.

Ein Petri-Netz heißt *lebendig,* wenn für jede erreichbare Markierung und jede Transition eine Folgemarkierung erreichbar ist, welche die gegebene Transition aktiviert.

Eine erreichbare Markierung ist ein *Deadlock*, wenn unter dieser Markierung keine Transition aktiviert.

Sicherheit ist eine sinnvolle Forderung an Petri-Netze, welche den Kontrollfluß modellieren. Lebendigkeit garantiert, daß jede Transition immer wieder aktiviert werden kann, daß also keine Netzteile aussterben. Aus der Lebendigkeit folgt immer die Deadlock-Freiheit, i. a. gilt die Umkehrung nicht.

Bei Petri-Netzen können sowohl Stellen wie Transitionen verzweigen. Eine Stelle mit mehreren Ausgangskanten beschreibt eine Alternative zwischen lokalen Aktionen, eine Stelle mit mehreren Eingangskanten bedeutet, daß verschiedene lokale Aktionen denselben lokalen Zustand erzeugen können. Eine Transition mit mehreren Ausgangskanten beschreibt die nebenläufige Verzweigung eines Prozeß und eine Transition mit mehreren Eingangskanten eine Synchronisation. Wichtig sind folgende Netzklassen:

- Ein *Zustandsautomat* (P-Netz) ist ein Netz, dessen Transitionen unverzweigt sind,
- ein *Synchronisationsgraph* (T-Netz) ist ein Netz, dessen Stellen unverzweigt sind.
- Eine Kombination beider ist ein *Free-Choice Netz*: Je zwei Transitionen mit mindestens einer gemeinsamen Vorstelle haben alle Vorstellen gemeinsam.

  Wenn in einem Free-Choice System mehrere Transitionen mindestens eine gemeinsame markierte Vorstelle haben (Konfliktfall), so sind entweder alle Transitionen aktiviert oder keine. Im ersten Fall kann man also „frei entscheiden“, welche Transition schaltet.

4.3.2 Beispiel eines Lebenszyklus als ungefärbtes Petri-Netz

Das folgende Beispiel stammt aus dem Fachkonzept eines Praxis-Projektes. Es zeigt den Lebenszyklus der Objekte einer gegebenen Geschäftsklasse.

Der Autor des Fachkonzepts will mit der Unterscheidung von Status und Eigenschaft ausdrücken, dass ein gegebener Status weiter detailliert werden kann durch lokale Statuswerte, die „Eigenschaften“ genannt werden. Z. B. bleibt ein Objekt der Geschäftsklasse im Status „in Bearbeitung“, während es die Detailzustände „vorgelegt“, „zur Übersetzung“ etc. durchläuft.

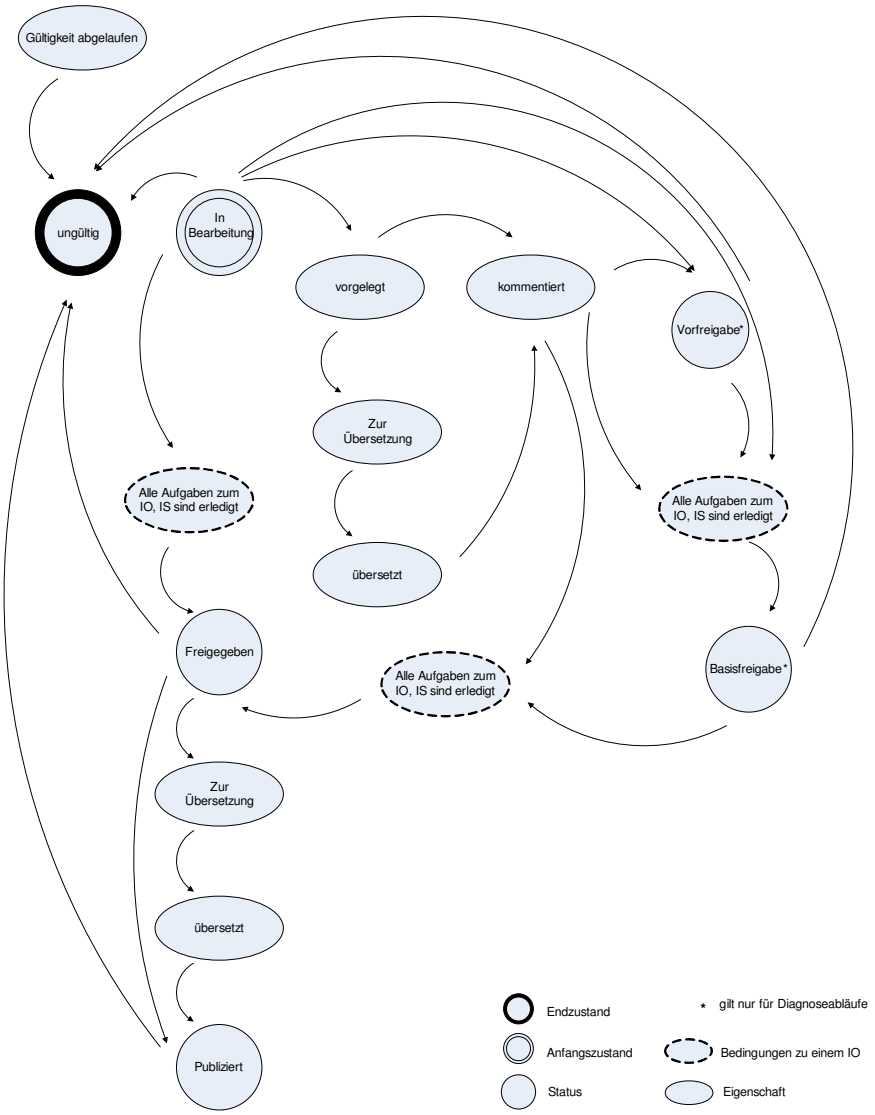


Abbildung 28: „Hierarchisches“ Zustandsmodell

Ein Zustandsdiagramm ohne Hierachisierung kann diesen Sachverhalt nur ungenügend ausdrücken. Wir zeigen daher in Abbildung 29, wie eine Modellierung des Lebenszyklus durch ein Petri-Netz aussieht.

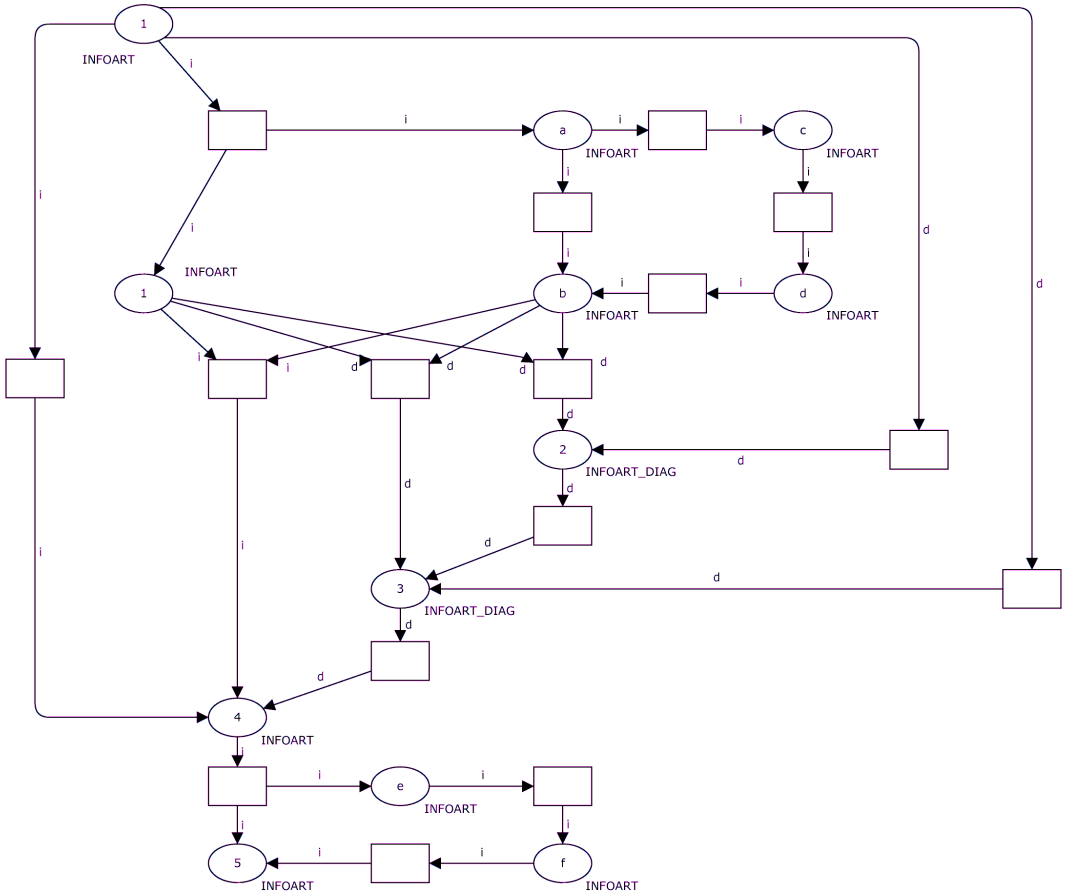


Abbildung 29: Petri-Netz des Lebenszyklus der Geschäftsklasse Informationsobjekt

Legende von Abbildung 29. Das Bild zeigt ein Petri-Netz: Ellipsen sind Stellen, Rechtecke sind Transitionen.

Die Stellen entsprechen den Status/Eigenschaften; im einzelnen:

1 = in Bearbeitung, 2 = Vorfreigabe, 3 = Basisfreigabe, 4 = Freigegeben, 5 = Publiziert

a = vorgelegt, b = kommentiert, c = zur Übersetzung, d = übersetzt, e = zur Übersetzung, f = übersetzt.

Hinweise. 1. Auf den Stellen mit Annotation „INFOART“ kann ein beliebiges Infoobjekt liegen, auf den Stellen mit Annotation „INFOART_DIAG“ können nur Infoobjekte einer bestimmten Art liegen. Entsprechend bedeutet der Buchstabe „i“ an einer Kante, dass hier ein beliebiges Infoobjekt fliessen kann, während der Buchstabe „d“ an einer Kante einschränkt, dass hier nur ein Infoobjekt der Diagnose fliessen kann. Aufgrund dieser Annotationen handelt es sich um ein gefärbtes Petri-Netz. Man erhält ein ungefärbtes Petri-Netz, wenn man die Annotationen aller Stellen und Kanten vergißt.

2. Von allen Stellen aus kann übergegangen werden in den Status „ungültig“; dieser Übergang ist in obigem Petri-Netz weggelassen. Auch die Stelle für den Zustand „Gültigkeit abgelaufen“ ist im Petri-Netz weggelassen.

Als einzige Stelle ohne Eingangskante hat die Stelle „1 = in Bearbeitung“ die Bedeutung des Anfangszustandes.

4.3.3 Modellierung des Gedächtnis einer Dialogsteuerung

Viele Standard-Dialoge zum Verwalten von Geschäftsobjekten – Aufträge, Kunden etc. - haben eine Dialogsteuerung wie in Abbildung 30. Der Anwender entscheidet im ersten Schritt, welche Art von Geschäftsobjekt er bearbeiten will, im zweiten Schritt welche Art von Bearbeitung er durchführen will, und im dritten Schritt an welcher konkreten Instanz er diese Bearbeitung durchführen will.

Wenn er z.B. die Gültigkeit der Inlands-Preisliste 4711 ändern will, so wählt er nacheinander: Geschäftsklasse *Preisliste*, Usecase *Preisliste.Ändern*, Objekt *Preisliste 4711*.

Das System sucht nach der dritten Eingabe des Anwenders nach einem konkreten Geschäftsobjekt und zeigt es auf der Maske an, die zu dem Usecase der zweiten Eingabe gehört. Die Masken der einzelnen Usecase unterscheiden sich: Auf Masken zum Ändern können Eingaben gemacht werden, auf Masken zum Anzeigen sind dagegen fast alle Felder gesperrt. Der Dialog muß sich also das Resultat des zweiten Schrittes, die Auswahl des Usecase, merken, und sich beim dritten Schritt daran erinnern. Diese Dialogsteuerung liefert ein Beispiel eines verteilten Zustandes, wenn der globale Zustand des Systems durch die Markierung der beiden Stellen *Gui.ÄndernGewählt* und *Gui.GeschäftsobjektGewählt* gegeben ist (Gui = Graphical User Interface).

Das Petri-Netz von Abbildung 30 enthält damit verteilte Zustände, die durch mehr als eine Marke gekennzeichnet sind. Nach der Auswahl eines Geschäftsobjektes zum Anzeigen liegt ein verteilter Zustand vor: Das Petri-Netz enthält zwei Marken, sein globaler Zustand setzt sich zusammen aus den beiden lokalen Zuständen *Gui.GeschäftsobjektGewählt* und *Gui.UsecaseAnzeigenGewählt*.

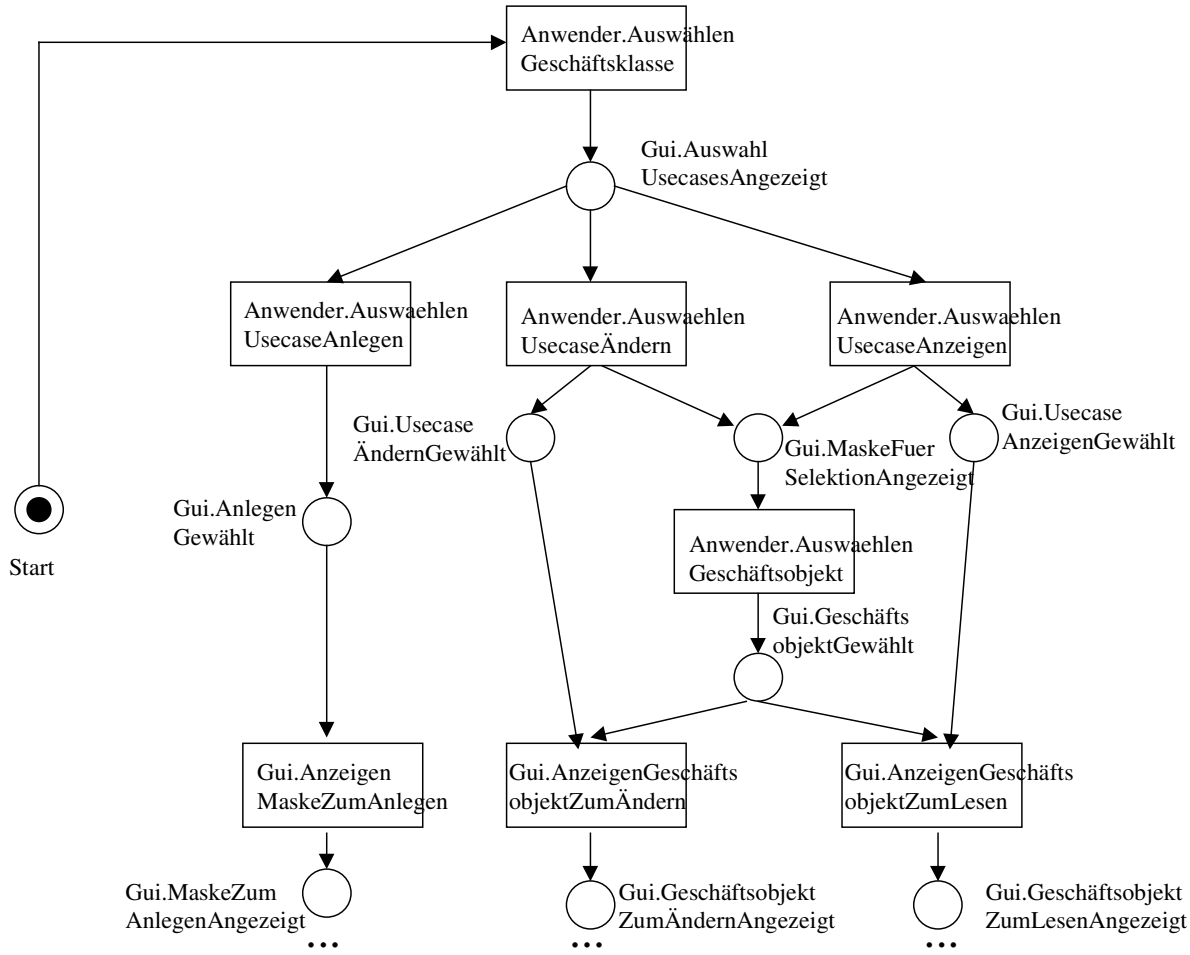


Abbildung 30: Petri-Netz einer Dialogsteuerung (Ausschnitt)

4.3.4 Übersetzung einer EPK in ein Free-Choice System

Um eine EPK zu verifizieren, empfiehlt es sich, sie in einen Testtreiber mit einem einzigen Ereignis „Start/Ziel“ einzuspannen, so daß ein stark-zusammenhängender Graph entsteht. Im nächsten Schritt wird die EPK in ein Petri-Netz übersetzt, um die Verfahren zur Analyse von Petri-Netzen anzuwenden.

EPKs ohne OR-Konnektoren, sogenannte AND/XOR-EPKs lassen sich in ein Free-Choice System übersetzen, wobei die Anfangsmarkierung die Stelle „Start/Ziel“ mit genau einer Marke markiert, siehe Abbildung 31.

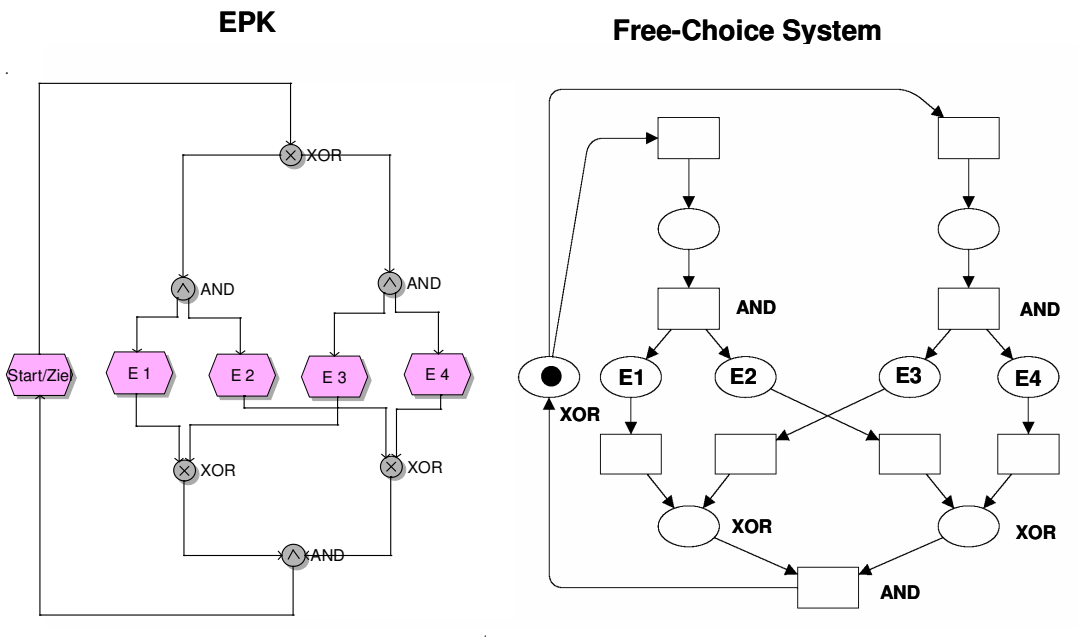


Abbildung 31: Übersetzung einer AND/XOR-EPK (Funktionen sind weggelassen) in ein Free-Choice System

Die Regeln zur Übersetzung einer AND/XOR-EPK in ein Free-Choice-System zeigt Abbildung 32. Aus syntaktischen Gründen, d.h. um einen bipartiten Graph zu erhalten, sind ggf. noch unverzweigte Stellen oder unverzweigte Transitionen einzufügen.

EPK	Free-Choice System
Ereignis	Stelle
Funktion	Transition
XOR-Konnektor	verzweigte Stelle
AND-Konnektor	verzweigte Transition
OR-Konnektor	???

Abbildung 32: Regeln zur Übersetzung einer AND/XOR-EPK in ein Free-Choice System

Eine AND/XOR-EPK heißt *wohlgeformt*, wenn das resultierende Free-Choice System lebendig und sicher ist. Insbesondere können die Stellen, die aus der Übersetzung eines XOR-Konnektors entspringen, unter keiner erreichbaren Markierung mehr als eine einzige Marke tragen.

Die folgende EPK von Abbildung 33 enthält zusätzlich OR-Konnektoren, die zudem nicht paarweise auftreten.

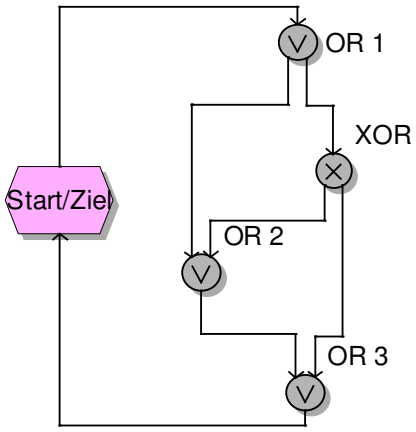


Abbildung 33: EPK mit OR-Konnektoren (vereinfacht ohne Funktionen und Ereignisse)

Es ist unklar, wie der OR-Konnektor bei der Übersetzung in ein ungefärbtes Petri-Netz behandelt werden soll. Denn er kann ja sowohl im AND-Modus als auch im XOR-Modus schalten. Wenn bei einem schließenden OR-Konnektor nur einer der Eingänge aktiviert ist, so ist lokal nicht entscheidbar,

- ob der andere Eingang auch noch aktiviert werden kann – in diesem Fall soll der Konnektor warten und noch nicht schalten –
- oder ob die Aktivierung des anderen Einganges unmöglich ist – in diesem Fall kann der Konnektor schalten.

Zur Entscheidung zwischen beiden Alternativen wird eine nicht-lokale Information über alle möglichen Abläufe der EPK gebraucht.

4.3.5 Übersetzung einer EPK in ein Boolesches System

Man kann OR-Konnektoren und beliebige andere logische Konnektoren jedoch in ein Petri-Netz übersetzen und ihnen damit eine lokale Semantik geben, wenn man gefärbte Petri-Netze betrachtet. In diesem Abschnitt werden Boolesche Systeme als einfache Beispiele für gefärbte Petri-Netze eingeführt.

Bei der Übersetzung in ein Boolesches System wird jeder logischer Konnektor – unabhängig von seiner logischen Bedeutung - in eine Transition übersetzt.

EPK	Free-Choice	Boole
Ereignis	Stelle	Stelle
Funktion	Transition	Transition
XOR-Konnektor	Stelle	**XOR-Transition**

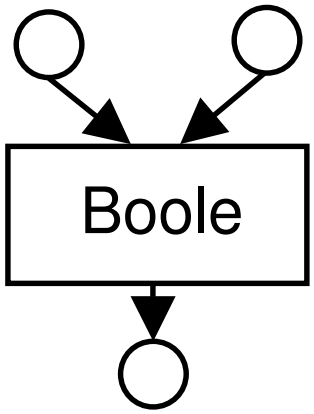
EPK	Free-Choice	Boole
AND-Konnektor	Transition	**AND-Transition**
OR-Konnektor	???	**OR-Transition**

Abbildung 34: Regeln zur Übersetzung einer EPK in ein Boolesches System

Die aus der Übersetzung der logischen Konnektoren entstandenen „Booleschen" Transitionen sind nur aktiviert, wenn jede ihrer Vorstellen mit mindestens einer Marke markiert ist. Auch eine schließende XOR-Transition oder eine schließende OR-Transition braucht also zwei Marken zur Aktivierung. Allerdings haben die Marken eine unterschiedliche Bedeutung: Die erste Art von Marken bedeutet die Anwesenheit des Kontrollflusses – wir nennen sie *high-Marken*. Die andere Art von Marken bedeutet den Ausschluß des Kontrollflusses – wir nennen sie *low-Marken*. In Abhängigkeit von der Kombination der beiden Markenarten auf ihren Vorstellen sind verschiedene Schaltmodi einer Booleschen Transition aktiviert.

Ein Boolesches System besteht aus einem ungefärbtem Netz N ,

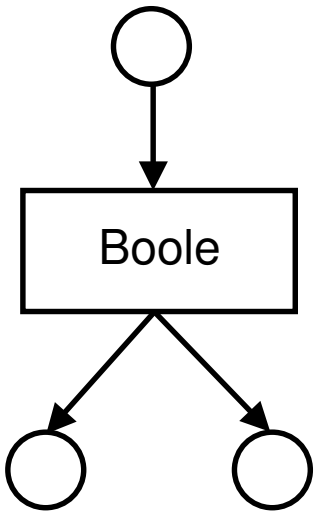
- bei dem jede Stelle eine Marke aus der Menge $BOOLE = \{ high, low \}$ tragen kann
- und bei dem jede Transition mehrere Schaltmodi hat, die gemäß den Tabellen von Abbildung 35 und Abbildung 36 aktiviert sind und schalten.



Typ	pre	post
AND	high, high	high
XOR	high, low	high
	low, high	high
OR	high, low	high
	low, high	high
	high, high	high
AND/XOR/OR	low, low	low

Abbildung 35: Schaltmodi einer schließenden Booleschen Transition

Durch Vertauschung der Rollen von „pre" und „post" in von Abbildung 35 entsteht das Schaltverhalten einer öffnenden Booleschen Transition in Abbildung 36.



Typ	pre	post
AND	high	high, high
XOR	high	high, low
	high	low, high
OR	high	high, low
	high	low, high
	high	high, high
AND/XOR/OR	low	low, low

Abbildung 36: Schaltmodi einer öffnenden Booleschen Transition

Abbildung 37 zeigt die Übersetzung der EPK von Abbildung 33 in ein Boolesches System mit OR-Transitionen.

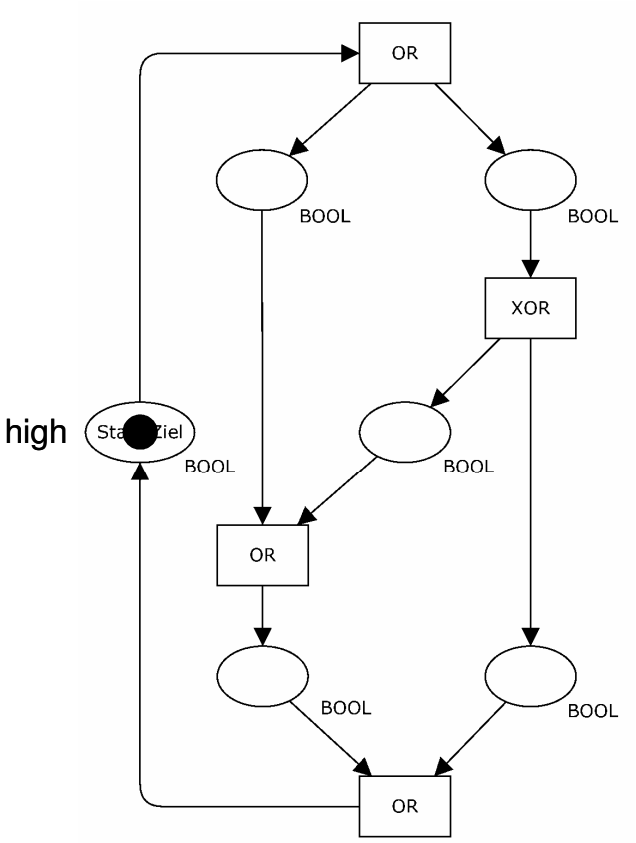


Abbildung 37: Übersetzung der EPK von Abbildung 33 (Funktionen sind weggelassen) in ein Boolesches System

Durch die Übersetzung in ein Boolesches System erhält jede EPK mit ihren öffnenden und schließenden Konnektoren eine lokale Semantik. Im Falle von Schleifen müssen diese einen wohldefinierten Anknüpfungspunkt an den Rest der EPK haben. Dieser Anknüpfungspunkt stellt den Schleifenanfang und das Schleifenende dar.

AND/XOR-EPKs haben damit zwei Semantiken, die Free-Choice Semantik und die Boolesche Semantik. Man kann zeigen, daß sie für wohlgeformte AND/XOR-EPKs äquivalent sind.

4.3.6 Gefärbtes Petri-Netz

Faßt man Petri-Netze als Modellierungssprache für Prozesse auf, so kann man verschiedene Dialekte von Petri-Netzen unterscheiden. Der einfachste Dialekt, die gewöhnlichen Petri-Netze, erhält Marken, die von einander nicht unterscheidbar sind, und jede Transition hat genau ein Schaltverhalten. Beim Schalten einer Transition fließt über jede ihrer Kanten genau eine Marke. Man kann die Sprache der gewöhnlichen Petri-Netze mit der Sprache Assembler vergleichen: Jeden anderen Dialekt von Petri-Netzen kann man in gewöhnliche Petri-Netze übersetzen. Aber gewöhnliche Petri-Netze stellen keine Hochsprache dar wie die Programmiersprachen Pascal oder C.

Die "Hochsprache“ zur Prozeßmodellierung sind gefärbte Petri-Netze. Hier kann man sowohl die Stellen als auch die Transitionen typisieren. Die Typisierung einer Stelle bedeutet, daß diese Stelle einer Klasse zugeordnet wird, Marken auf dieser Stelle bedeuten dann Instanzen dieser Klasse in einem bestimmten Zustand. Die Typisierung einer Transition bedeutet verschiedene Schaltmodi einer lokalen Aktion einzuführen. Boolesche Systeme sind eine einfache Klasse von gefärbten Petri-Netzen. Sie haben nur zwei Markenfarben und verschiedene Schaltmodi für Transitionen.

Ein *gefärbtes Petri-Netz* ist ein Tupel

$$CPN = \left(N, C(p)_{p\in P}, B(t)_{t\in T}, \left(w^{-/+}{}_{(t,p)}\right)_{(t,p)\in T\times P}, \mu_0\right)$$

bestehend aus

- einem ungefärbtem Netz N mit Stellenmenge P und Transitionenmenge T
- für jede Stelle $p\in P$ einer Menge $C(p)$ von möglichen Markenfarben der Stelle
- für jede Transition $t\in T$ einer Menge $B(t)$ von möglichen Schaltmodi der Transition
- für jedes Paar $(t,p)\in T\times P$ den beiden Inzidenzabbildungen $w^{-}{}_{(t,p)}, w^{+}{}_{(t,p)} : B(t)\longrightarrow C(p)_N$ in das von $C(p)$ erzeugte freie kommutative Monoid
- und der Anfangsmarkierung $\mu_0 : P\longrightarrow \bigcup_{p\in P} C(p)_N$ mit $\mu_0(p)\in C(p)_N$ für alle $p\in P$.

Unter einer Markierung $\mu : P\longrightarrow \bigcup_{p\in P} C(p)_N$, $\mu(p)\in C(p)_N$ für alle $p\in P$, ist ein Schaltelement

$$(t,b) \text{ mit } t\in T \text{ und } b\in B(t)$$

genau dann *aktiviert*, wenn auf allen Vorstellen genügend Marken liegen, d.h. wenn für alle Stellen $p \in P$ gilt

$$\mu(p) \geq w^{-}(t,p)(b).$$

Ein Schaltelement (t,b), das unter einer Markierung μ_{pre} aktiviert ist, kann *schalten*. Beim Schalten erzeugt es die Folgemarkierung μ_{post} mit

$$\mu_{post}(p) := \mu_{pre}(p) - w^{-}(t,p)(b) + w^{+}(t,p)(b),\ p \in P.$$

Gefärbte Petri-Netze unterstützen die Prinzipien einer objektorientierte Prozeßmodellierung und damit insbesondere die Modellierung des Datenflusses:

- Klassenbildung (Abschnitt 3.1): Die Farbmenge $C(p)$ einer Stelle $p \in P$ entspricht einer Geschäftsklasse, jede Transition ist eine Operation einer dieser Geschäftsklassen.
- Wechselwirkung (Abschnitt 3.3): Ein Ablauf (Run) des Petri-Netzes zeigt im Markenfluß die Wechselwirkung der Geschäftsklassen als eine Folge von Operationsaufrufen ihrer Geschäftsklassen.
- Multi-Skalen-Analyse (Abschnitt 3.4): Die Petri-Netze zweier benachbarter Ebenen unterschiedlicher Detaillierung werden durch eine vergröbernde Abbildung

  $$f : CPN_{fein} \longrightarrow CPN_{grob}$$

  in Beziehung gesetzt. Sie hat als Urbild einer Stelle $p \in P_{grob}$ ein stellenberandetes Teilnetz $f^{-1}(p) \subset CPN_{fein}$ und als Urbild einer Transition $t \in T_{grob}$ ein transitionsberandetes Teilnetz $f^{-1}(t) \subset CPN_{fein}$. Insbesondere kann der Datenfluß als eine Verfeinerung des Kontrollflusses angesehen werden. Dieser Zusammenhang läßt sich durch eine Abbildung

  $$f : CPN_{Daten} \longrightarrow CPN_{Kontroll}$$

  formalisieren.

4.4 BPEL

BPEL = BPEL4WS: Business Process Execution Language for Web Services.

BPEL ist eine Sprache zur Spezifikation von Geschäftsprozessen, die auf Web-Services beruhen ([BPEL2006]).

4.4.1 Grundzüge von BPEL 2.0

Ein Geschäftsprozeß importiert diese Web-Services: Damit steht er zu ihnen in einem Kunden-Lieferanten Verhältnis. Die Web-Services können von Komponenten erbracht werden, bei denen der Geschäftsprozeß nur die Schnittstelle kennt. Die Implementierung der Web-Services ist nicht Gegenstand von BPEL und wird nicht unterstützt: BPEL ist eine Sprache zur Spezifikation und Implementierung von Geschäftsprozessen, nicht zur Implementierung von Komponenten.

Die Services dieser Komponenten dienen als Bausteinen, um den Geschäftsprozeß im Sinne einer „serviceorientierten Architektur“ (SOA) zusammenzubauen („orchestrieren“).

BPEL überspannt den gesamten Bereich von der Modellierung bis zur Ausführung von Geschäftsprozessen. Damit enthält ein in BPEL erstellter Geschäftsprozeß auch den Datenfluß mit seinen steuernden und zeitbehafteten Aspekten. Außerdem zeigt das Modell nicht nur den OK-Fall des Geschäftsprozesses, sondern explizit auch alle Ausnahmebehandlungen. Diese sollen die bisherigen Aktivitäten zurücksetzen bzw. durch andere Aktivitäten kompensieren.

Folgende grundlegenden Konzepte übernimmt BPEL vom Komponentenmodell:

- Partner: Welche Web-Services ruft der Geschäftsprozeß auf, von welchen Clients wird er selbst aufgerufen?
- Partner-Link: Welche Schnittstellen bestehen zwischen den Partnern. Jede Schnittstelle ist ein Kommunikationskanal, über den die Operationen aufgerufen werden können, die der Partner zur Verfügung stellt. Eine Kommunikation über einen Partner-Link erfolgt entweder synchron oder asynchron.
- Aktivität: Einfache Aktivitäten sind die Aktionen des Geschäftsprozesses, mit denen er entweder Web-Services aufruft oder prozeßinterne Verarbeitungen durchführt. Daneben gibt es Aktivitäten zur Modellierung des Kontrollflusses.

Aktivität	Bedeutung
receive	Der Prozeß wartet auf den Callback eines Partners.
assign	Der Prozeß kopiert Daten.
invoke	Der Prozeß ruft einen Partner auf.
flow	Der Kontrollfluss des Prozesses durchläuft eine AND-Alternativer aus einem Paar von öffnenden und schließenden Konnektoren.
switch	Der Kontrollfluss des Prozesses durchläuft eine XOR-Alternative aus einem Paar von öffnenden und schließenden Konnektoren

Aktivität	Bedeutung
	aus einem Paar von öffnenden und schließenden Konnektoren.
while	Der Kontrollfluss des Prozesses durchläuft eine Schleife.

Abbildung 38: BPEL-Aktivitäten in Auswahl

4.4.2 Beispiel „Buchung Geschäftsreise“ in BPEL

Wir erläutern die Sprache BPEL an einem einfachen Beispiel aus der Literatur ([Jur2006]), siehe Abbildung 39:

Aus den Diensten externer Komponenten soll ein Prozeß gebaut werden, der einem Antragsteller in der Rolle des „Client“ ein Flugangebot macht. Der Prozeß muß dazu feststellen, welche Konditionen der Antragsteller aufgrund seines Status besitzt, und dann die Angebote der beiden Fluggesellschaften einholen und miteinander vergleichen.

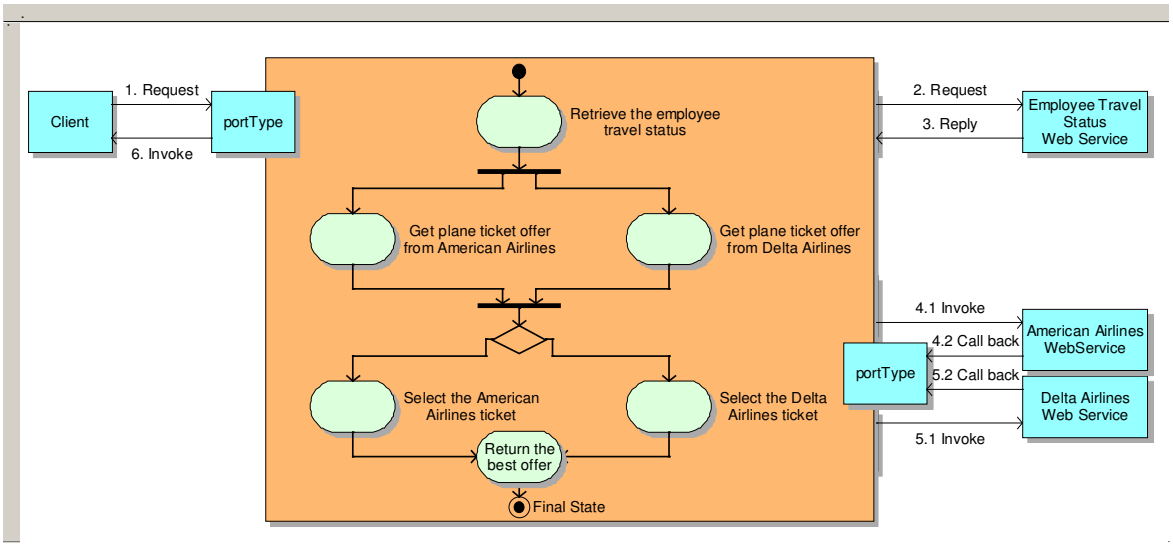


Abbildung 39: Geschäftsprozeß „Business Travel“

Das Beispiel von Abbildung 39 enthält:

- Fünf Partner:
  - Client,
  - Geschäftsprozeß,
  - Web-Service „Travel_Status“,
  - Web-Service „American_Airlines“,
  - Web-Service „Delta_Airlines“.

- Vier Partner-Links als Schnittstelle zwischen dem Geschäftsprozeß und einem der anderen Partner:
  - Prozeß-Client (asynchron),
  - Prozeß-Travel_Status (synchron),
  - Prozeß-American_Airline (asynchron),
  - Prozeß-Delta_Airline (asynchron).

  Bei einem Partnerlink für synchrone Kommunikation können nur an dem Port des Prozeßpartners (Partner role) Operationen der Aktivität „invoke“ aufgerufen werden. BPEL nennt diese Seite den Rollentyp „Partner role“.

  Bei einem Partnerlink für asynchrone Kommunikation ruft der Prozeß an dem Port des Prozeßpartners (Partner role) die Aktivität „invoke“ auf. An dem Port des Prozesses (my role) ruft der Prozeßpartner die Aktivität „receive“ auf.
- Verschiedene Operationen zur Implementierung der Aktivitäten.

Da BPEL auch den Datenfluß des Prozesses „Business Travel“ von Abbildung 41 zeigen soll, werden wir den Prozeß objektorientiert modellieren. Als erstes entwerfen wir im Sinne des Prinzips der Klassenbildung (Abschnitt 3.1) die beteiligten Klassen. Dieser Entwurf geschieht gemäß dem Prinzip der Multi-Skalen-Analyse (Abschnitt 3.4) auf zwei Ebenen der Detaillierung.

Auf der groben Ebene wird jeder der fünf Partner zu einer Klasse. Diese Klassen haben als Operationen die einfachen Aktivitäten des Geschäftsprozesses (siehe Abbildung 40).

Klasse	Operation	Zustand
BusinessTravel	• receiveTravelRequest • createEmployee (assign) • createAARequest (assign) • createDeltaAirlinesRequest (assign) • receiveAAOffer • receiveDeltaAirlinesOffer • createAAResponse (assign) • createDeltaAirlinesResponse (assign)	• BusinessTravel.Start • BusinessTravel.End
TravelStatus	deliverEmployeeStatus (invoke)	
AmericanAirlines	deliverOffer (invoke)	
DeltaAirlines	deliverOffer (invoke)	
Client	acceptTravelResponse (invoke)	

Abbildung 40: Klassenbildung der groben Ebene

Abbildung 41 zeigt den Prozeß der groben Ebene in BPEL.

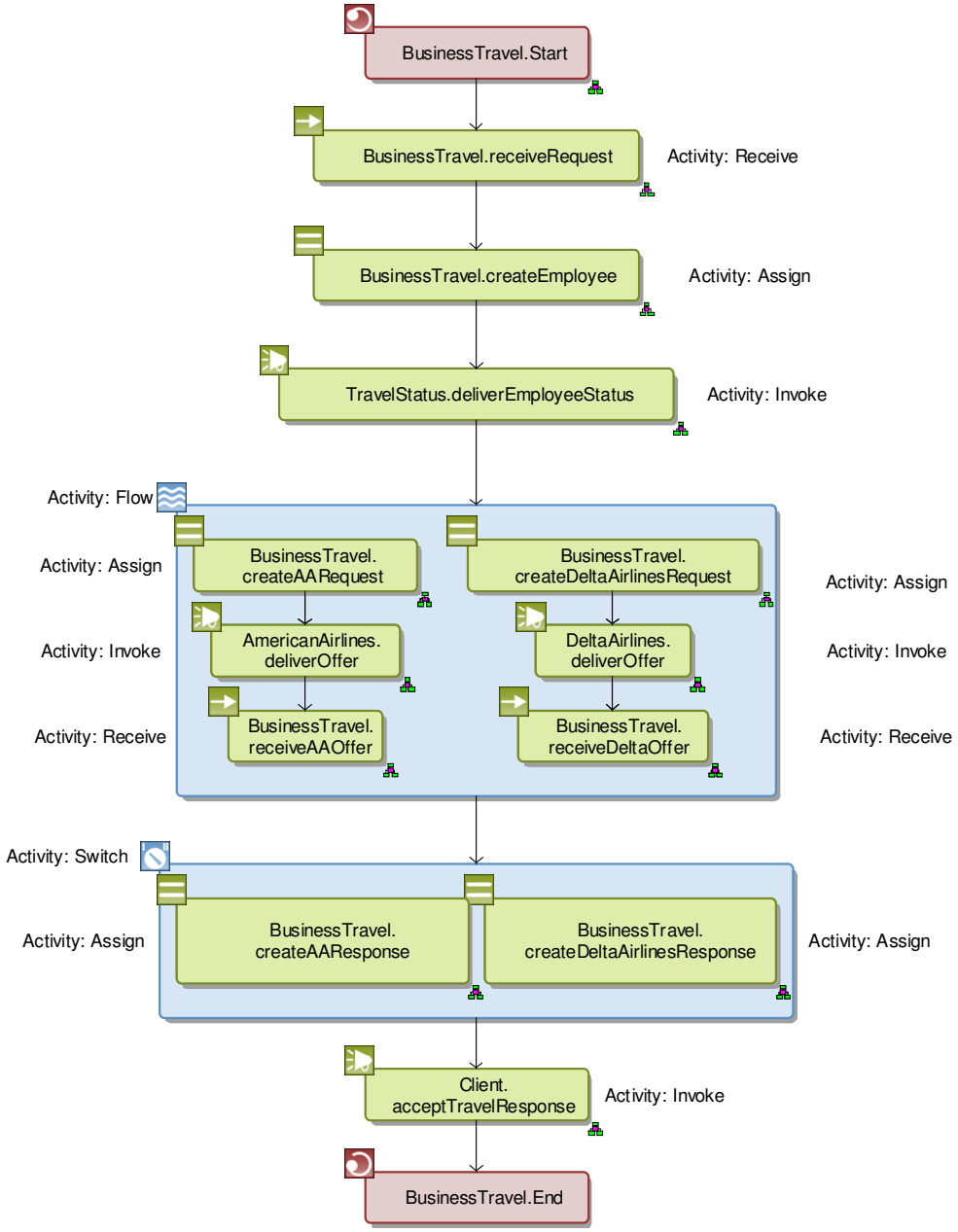


Abbildung 41: Geschäftsprozeß „Business Travel“ in BPEL

Die feine Ebene detailliert jede Aktivität der groben Ebene durch eine Operation einer Detailklasse. Wir führen die folgenden Detailklassen ein (siehe Abbildung 42):

Klasse	Operation
TravelRequest	create
Employee	• create • getTravelStatus
FlightAmericanAirlines	• create • computeOffer • getOffer
FlightDeltaAirlines	• create • computeOffer • getOffer
TravelOffer	• create • visualize

Abbildung 42: Klassenbildung der feinen Ebene

Aus der graphischen BPEL-Modellierung von Abbildung 41 generiert das Tool ARIS die folgende XML-Datei (siehe Abbildung 43).

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by the ARIS Business Architect, IDS Scheer AG. All rights reserved. www.ids-scheer.com-->
<process expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116" name="BusinessTravel.Start" queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116" targetNamespace="http://www.ids-scheer.com/bpel" xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:tns="http://www.ids-scheer.com/bpel/wsdl">
    <partnerLinks>
        <partnerLink myRole="roleName" name="DeltaAirlines" partnerLinkType="tns:DeltaAirlinesLT" partnerRole="roleName"/>
        <partnerLink myRole="roleName" name="Client" partnerLinkType="tns:ClientLT" partnerRole="roleName"/>
        <partnerLink myRole="roleName" name="AmericanAirlines" partnerLinkType="tns:AmericanAirlinesLT" partnerRole="roleName"/>
        <partnerLink name="TravelStatus" partnerLinkType="tns:TravelStatusLT" partnerRole="roleName"/>
    </partnerLinks>
    <sequence>
        <receive name="BusinessTravel_x2e__xd__xa_receiveTravelRequest" operation="TravelRequest.create" partnerLink="Client" portType="tns:Travelrequest"/>
        <assign name="BusinessTravel.createEmployee">
            <copy>
                <from variable="TravelRequest"/>
                <to variable="Employee"/>
            </copy>
        </assign>
        <invoke name="TravelStatus.deliverEmployeeStatus" operation="Employee.getTravelStatus" partnerLink="TravelStatus" portType="tns:TravelStatus"/>
        <flow name="">
            <sequence>
                <assign name="BusinessTravel_x2e__xd__xa_createAARequest">
                    <copy>
                        <from variable="TravelRequest"/>
                        <to variable="FlightAmericanAirlines"/>
```

```
                                    </assign>
                                    </copy>
                                    <invoke name="AmericanAirlines_x2e__xd__xa_deliverOffer" operation="FlightAmericanAirlines.computeOffer" partnerLink="AmericanAirlines" portTy-
pe="tns:FlightAmericanAirlines"/>
                                    <receive name="BusinessTravel.receiveAAOffer" operation="FlightAmericanAirlines.getOffer" partnerLink="AmericanAirlines" portTy-
pe="tns:AmericanAirlinesOffer"/>
                            </sequence>
                            <sequence>
                                    <assign name="BusinessTravel_x2e__xd__xa_createDeltaAirlinesRequest">
                                            <copy>
                                                    <from variable="TravelRequest"/>
                                                    <to variable="FlightDeltaAirlines"/>
                                            </copy>
                                    </assign>
                                    <invoke name="DeltaAirlines_x2e__xd__xa_deliverOffer" operation="FlightDeltaAirlines.computeOffer" partnerLink="DeltaAirlines" portTy-
pe="tns:FlightDeltaAirlines"/>
                                    <receive name="BusinessTravel_x2e__xd__xa_receiveDeltaAirlinesOffer" operation="FligthDeltaAirlines.getOffer" partnerLink="DeltaAirlines" portTy-
pe="tns:DeltaAirlinesOffer"/>
                            </sequence>
                    </flow>
                    <switch name="">
                            <case>
                                    <assign name="BusinessTravel_x2e__xd__xa_createDeltaAirlinesResponse">
                                            <copy>
                                                    <from variable="FlightDeltaAirlines"/>
                                                    <to variable="TravelOffer"/>
                                            </copy>
                                    </assign>
```

```
                        <case>
                        </case>   <assign name="BusinessTravel_x2e__xd__xa_createAAResponse">
                                    <copy>
                                        <from variable="FlightAmericanAirlines"/>
                                        <to variable="TravelOffer"/>
                                    </copy>
                                </assign>
                        </case>
                </switch>
                <invoke name="Client_x2e__xd__xa_acceptTravelResponse" operation="TravelOffer.visualize" partnerLink="Client" portType="tns:TravelOffer"/>
        </sequence>
</process>
```

Abbildung 43: Geschäftsprozeß „Business Travel“ in XML-Datei

Im nächsten Schritt ist das Modell um die Aktivitäten zur Fehlerbehandlung anzureichern.

Auf einer dritten Ebene der Detaillierung können die Operationen mit Variablen und Parametern versorgt werden. Damit erhält man ein BPEL-Modell, das auf einer BPEL-Engine ausführbar ist (siehe [Jur2006]).

5 Fallbeispiel eines kommerziellen Projekts

Abbildung 44 zeigt einige typische kommerzielle Projekte aus der Branche „Industrie“.

Nr.	Betriebswirtsch. Funktion	Phasen vgl. Abschnitt 2.3.1	Art der Software	Projektgegenstand
1	Faktura	1-4	Individualsoftware	Stammdatenpflege und Bepreisung aller Kundenaufträge
2	Teilevertrieb	2-4	Individualsoftware	Verkauf von Ersatzteilen an Werkstätten
3	Produktdaten	2 (Projekt wurde vom Kunden abgebrochen)	Individualsoftware geplant	Bereitstellung von Produkt- und Marketinginformationen für internationale Märkte
4	Service	2	Standardsoftware eines Dritt-Lieferanten	Stammdatenbereitstellung für internationale Märkte und Abwicklung des Werkstattaufenthaltes

Abbildung 44: Beispiele von IT-Projekten der Branche Industrie

In den Abschnitten 5.2 - 5.4 werden wir das Projekt Nr. 2 aus Abbildung 44 im Detail vorstellen.

5.1 Rollen im Projekt

Die Projektpartner treten in Rollen auf. Zunächst gibt es die beiden Rollen

- Auftraggeber
- Auftragnehmer.

Der Auftraggeber ist der Kunde, der Auftragnehmer ist der Lieferant. Der Auftraggeber erteilt einen Auftrag, der Auftragnehmer liefert eine Dienstleistung und erhält dafür vom Auftraggeber eine Bezahlung.

Je nach Marktsituation sind die Gewichte beider Rollen verteilt. In dem einen Extremfall kann sich der Auftragnehmer seine Auftraggeber aussuchen und sogar die betriebswirtschaftlichen Abläufe seines Kunden beeinflussen. Das war die Situation in der Anfangszeit der SAP-Systems. Im anderen Extremfall sitzt der Auftraggeber am längeren Hebel. Er diktiert dann nicht nur den Umfang des Auftrages, sondern auch den Preis der Dienstleistung. Diese Situation kann bei Konzernen auftreten, die ein Softwarehaus als eine ihrer Tochterfirmen gekauft haben, und nun gegenüber dem Softwarehaus als Auftraggeber auftreten.

In mittelgroßen Projekten sind in der Analysephase auf Seiten des Kunden als Auftraggeber ca. 20 Mitarbeiter beteiligt sind, auf Seiten des Dienstleisters als Auftragnehmer ca. 10. Diese Mitarbeiter nehmen verschiedene Rollen ein.

In größeren Unternehmen mit eigener IV-Abteilung differenziert sich die Rolle des Kunden in die einzelnen Rollen

- Fachabteilung: Sachbearbeiter in betriebswirtschaftlichen Funktionen - Produktion, Logistik, Vertrieb, Distribution oder auch Aktivgeschäft, Passivgeschäft, etc. - als Anwender der IV-Systeme
- Abteilungen für IV-Systeme: Betreiber der IV-Systeme, verwaltet das Projektbudget
- IV-Querschnittabteilungen: Erstellt IV-Richtlinien, bewertet IV-Tools etc.

Innerhalb des Unternehmens stehen die drei Abteilungsarten wiederum in einem Kunden-Lieferanten-Verhältnis: IV-Querschnittsabteilungen sind Dienstleister für Abteilungen für IV-Systeme, diese wiederum sind Dienstleister für die Fachabteilungen. Auftraggeber nach außen ist i. a. eine IV-Abteilung, nicht die Fachabteilung.

Auf Seiten des Auftragnehmers unterteilt sich die Rolle des Dienstleisters in die Rollen

- Projektmanager: Akquisition, Budget, Personalrekrutierung
- Projektleiter: Personaleinsatz, Projektverfolgung
- Berater: Analyse
- Systemarchitekt
- Entwickler

Dabei können die Rollen des Projektmanagers und des Projektleiters in der Analysephase von derselben Person wahrgenommen werden.

5.2 Projektauftrag Teilevertrieb

Der Projektauftrag für das Projekt Teilevertrieb (Projekt Nr. 2 aus Abbildung 44) ist gemäss den Vorgaben von Kapitel 2.3.2 (Projektdefinition) aufgebaut.

Das Projekt soll den Vertrieb von Ersatzteilen an Service-Werkstätten unterstützen. Es handelt sich um die Ersatzteile für die Produkte eines Industriebetriebes.

Zunächst wird ein Grobkonzept beauftrag:

Das bestehende System zum Vertrieb von Ersatzteilen soll durch ein neues System abgelöst werden. Dieses solle eine „zukunftssicherer Technologie“ verwenden und als Individualsoftware entwickelt werden. Das neue System soll mit den bestehenden Systemen im Servicebereich des Industriebetriebes integriert werden. Das sind *IT-technische* Anforderungen.

Ausserdem werden Anforderungen zum *betriebswirtschaftlichen* Inhalt des neuen Systems gestellt:

- Kundenverwaltung

- Angebotskalkulation
- Angebotsanforderung und -versendung über das Internet
- Unterstützung von Verkaufskampagnen
- Schnittstelle zu den Systemen der Werkstätten
- Bestellung über das Internet
- Bestellverfolgung
- Reporting und Controlling der operativen Vorgänge

Der Projektauftrag identifiziert die wesentlichen Schnittstellen zu Nachbarsystemen.

Der Projektauftrag erhält einen geschätzten Aufwand (50 BT) für die Erstellung des Grobkonzeptes und einen Fertigstellungstermin.

5.3 Grobkonzept Teilevertrieb

Das Grobkonzept wird intern durch das Unternehmen, d.h. ohne Beteiligung eines externen Dienstleisters, erstellt. Es hat ca. 90 Seiten.

Das Grobkonzept ist gemäss den Vorgaben von Kapitel 2.3.2 (Problemanalyse) aufgebaut. Es enthält u.a. die Kapitel:

- IST-Analyse (Systeme, Grobdatenmodell, Architektur, Schnittstellen)
- Ziele des Projekts (siehe Projektauftrag 5.2)
- Anforderungen
- Alternativen (mit Entscheidung für eine Alternative)
- SOLL-Analyse (Aktoren, Geschäftsprozesse, Usecases, Architektur, Schnittstellen)
- Allgemeine Punkte

Zwei Drittel des Fachkonzepts umfasst die Beschreibung der 34 Usecases, jeweils mit einem oder zwei Screenshots und den zugehörigen UML Usecasediagrammen (siehe Abschnitt 4.2.3).

Das Grobkonzept wurde nicht nach den in Abschnitt 3 genannten Prinzipien erstellt. Insbesondere wurden weder Geschäftsklassen noch andere Klassen gebildet. Alle Informationen haben denselben Grad der Detaillierung; es gibt kein SOLL-Datenmodell.

5.4 Fachkonzept Teilevertrieb

Das Fachkonzept ist gemäss den Vorgaben von Kapitel 2.3.2 (Aufgabendefinition) aufgebaut. Es wurde vom Auftragnehmer, einem Softwarehaus, unter Mitarbeit des Kunden aus dem Industriebetrieb erstellt.

Methodisch befolgt es die OO-Prinzipien von Kapitel 3. Abbildung 45 zeigt sein Inhaltsverzeichnis.

Abbildung 45: Fachkonzept Inhaltsverzeichnis

Das Fachkonzept ist nach dem Prinzip der Multi-Skalen-Analyse (Abschnitt 3.4) auf zwei Ebenen angelegt, der Übersichtsebene und der Detailebene.

Die Übersichtsebene beginnt gemäß dem Prinzip der Klassenbildung (Abschnitt 3.1) mit der Identifizierung und Beschreibung der Geschäftsklassen des Projekts.

5.4.1 Übersichtsebene: Geschäftsklassen und ihre Usecases

Abbildung 46 zeigt die Geschäftsklassen des Projekts und ihre Operationen in Form von Use-cases.

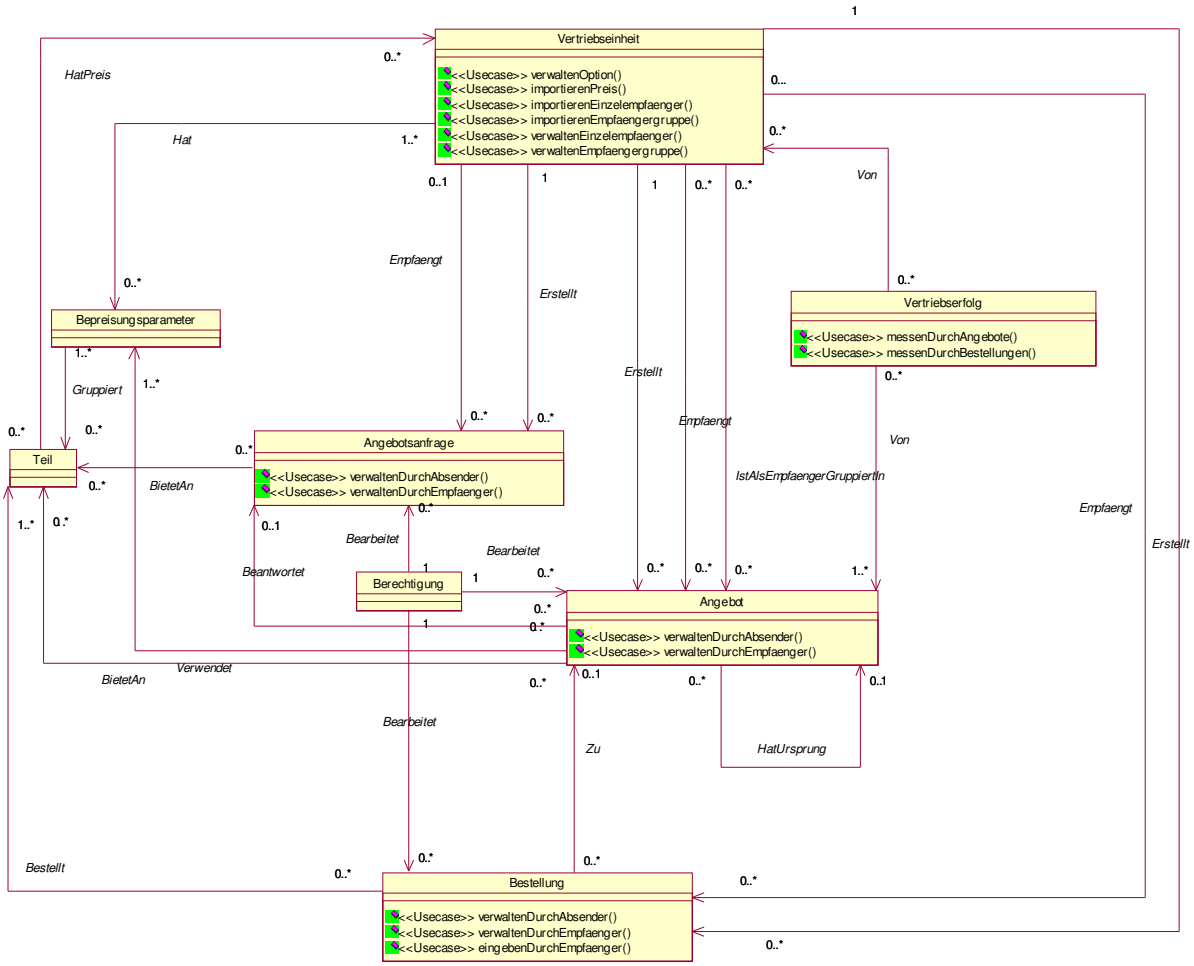


Abbildung 46: Geschäftsklassen und Usecases des Projektes (Übersichtsebene)

Die Beschreibung dieses Diagramms der Übersichtsebene beschreibt zugleich das gesamte Projekt:

Das Projekt „Teilevertrieb“ unterstützt den Teilevertrieb durch ein IV-System mit verschiedenen Funktionen.

Das Unternehmen hat eine mehrstufige hierarchische Vertriebsstruktur, die aus „Vertriebseinheiten“ aufgebaut ist. Die kaufmännischen Eigenschaften einer Vertriebseinheit werden durch den Usecase „Vertriebseinheit.verwaltenOption“ eingestellt. Die möglichen Kunden einer Vertriebseinheit, d.h. die Empfänger ihrer Angebote, und die für diese Kunden geltenden Preise werden mit Hilfe der Usecases „Vertriebseinheit.importierenEinzelempfänger“ und Vertriebseinheit.importierenEmpfängergruppe" aus anderen Systemen importiert. Unabhängig vom Import können weitere Kunden mit den Usecases „Vertriebseinheit.verwaltenEinzelempfänger“ und „Vertriebseinheit.verwaltenEmpfängergruppe“ geändert werden.

Eine Vertriebseinheit einer tieferen Ebene kann an ihre Vertriebseinheit der höheren Ebene eine „Angebotsanfrage“ richten. Dazu dient der Usecase „Angebotsanfra-

ge.verwaltenDurchAbsender“. Der Empfänger bearbeitet die Anfrage durch den Usecase „Angebotsanfrage.verwaltenDurchEmpfänger“.

Als Reaktion auf die Anfrage kann der Empfänger ein „Angebot“ erstellen. Gegenstand des Angebots ist der Verkauf von „Teilen“, ihre Preise werden mit Hilfe von „Bepreisungsparametern“ kalkuliert. Die Angebotserstellung geschieht im Usecase „Angebot.verwaltenDurch Absender“. Der Empfänger des Angebots kann das Angebot im Usecase „Angebot.verwaltenDurch Empfänger“ bearbeiten und seinerseits seinen Kunden der nächsttieferen Vertriebsebene zustellen.

Jeder Kunde kann auf Basis von erhaltenen Angeboten eine „Bestellung“ auslösen mit dem „Usecase Bestellung.verwaltenDurchAbsender“. Der Empfänger der Bestellung bearbeitet diese mit dem Usecase „Bestellung.verwaltenDurch Empfänger“.

Die Bearbeitung von Angebotsanfragen, Anfragen und Bestellungen mit der Applikation unterliegt einem Konzept für „Berechtigungen“, welches die Ausführung von Usecases an fachliche Rollen koppelt.

Jede Vertriebseinheit kann den „Vertriebserfolg“ ihrer Angebote durch Kennzahlen messen. Hierzu dienen die Usecases „Vertriebserfolg.messenDurchAngebote“ und „Vertriebserfolg.messenDurchBestellungen“.

5.4.2 Übersichtsebene: Beispiele von Geschäftsklassenbeschreibungen

Jede der in Abbildung 46 gezeigten Geschäftsklasse wird als gegliederter, freier Text beschrieben. Die Beschreibung enthält i. a. die Abschnitte: Definition, Schlüssel, Lebenszyklus. Die weiteren Abschnitte richten sich nach den Charakteristika der Geschäftsklasse.

Abbildung 47 zeigt die Gliederung der Geschäftsklasse „Angebot". Sie hat einen nichttrivialen Lebenszyklus und als Operationen eine Reihe wichtiger Usecases.

Abbildung 47: Geschäftsklasse „Angebot“

Ein Ausschnitt aus der Beschreibung der Geschäftsklasse „Angebot“:

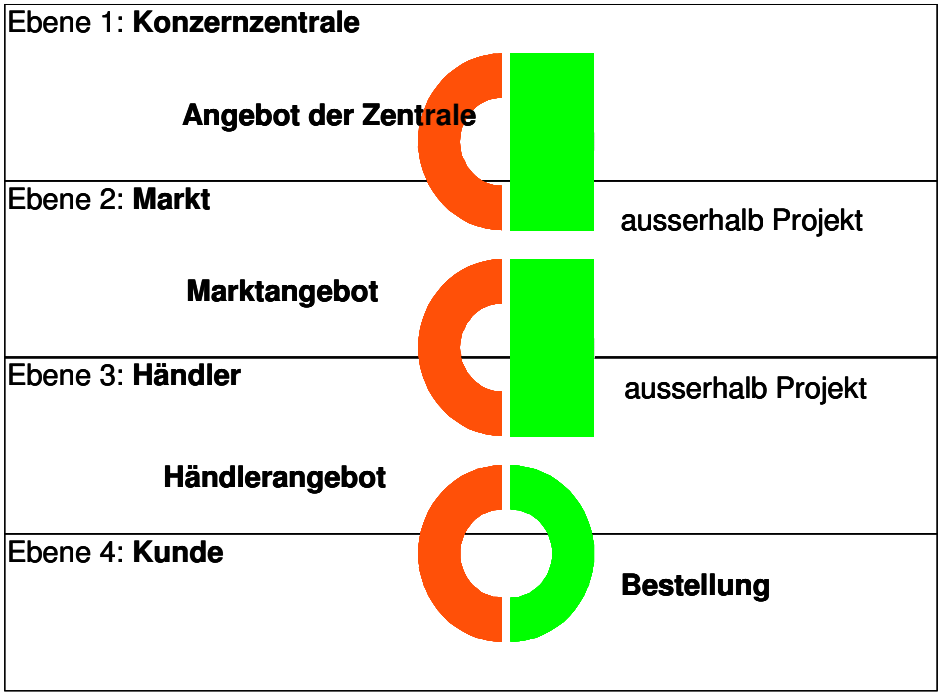


Abbildung 48: Geschäftsklasse Angebot

Definition
Ein `Angebot` ist eine Willenserklärung, die durch Annahme zu einem Rechtsgeschäft wird.

Im Rahmen des Projekts „Teilevertrieb“ bietet ein `Angebot` einem oder mehreren Empfängern die Lieferung von einem oder mehreren `Teilen` zu einem festgelegten Teilepreis an.

Das Projekt „Teilevertrieb“ arbeitet ausschliesslich mit freibleibenden `Angeboten`. Bei ihnen ist die Lieferverpflichtung durch eine Klausel wie z.B. „solange Vorrat“ eingeschränkt.

Angebotsarten
In Abhängigkeit von der Vertriebsebene (siehe Geschäftsklasse „Vertriebseinheit“), welche das `Angebot` erstellt, unterscheidet man folgende Arten von `Angeboten`:

- `Angebot der Zentrale`; erstellt von einem `Aftersales Manager` der Konzernzentrale.
- `Marktangebot`; erstellt von einem `Marktverantwortlichen` in einer Vertriebsgesellschaft oder bei einem Importeur.
- `Händlerangebot`; erstellt von einem `Mitarbeiter eines Händlers`.

Ein `Angebot` kann den Charakter einer Aktion haben: Es wird dann nicht regelmäßig, sondern nur zu besonderen Gelegenheiten erstellt und bietet die `Teile` zu einem günstigeren Preis an. Das Projekt „Teilevertrieb“ macht in seiner Funktionalität jedoch keinen Unterschied zwischen `Angeboten` mit dem Charakter von Aktionen und `Angeboten` ohne den Charakter von Aktionen: Der Aktionscharakter kann lediglich durch den Slogan des `Angebotes` verdeutlicht werden.

Schlüssel
Jedes `Angebot` wird durch einen Schlüssel („Angebotsnummer“) identifiziert. Er setzt sich zusammen aus der Verschlüsselung derjenigen `Vertriebseinheit`, welche das `Angebot` erstellt, und einer Nummer.

`Angebote`, die als Kopien anderer `Angebote` entstanden sind, tragen keinen Bezug zu ihren Kopiervorlagen.

Name
Jedes `Angebot` besitzt einen durch den Benutzer frei zu wählenden Namen. Das Projekt „Teilevertrieb“ erlaubt die Anlage mehrerer gleichnamiger `Angebote` innerhalb einer `Vertriebseinheit` – es liegt in der Verantwortung der Benutzer, `Angeboten` identifizierende Namen zu geben.

Gültigkeit
Das Gültigkeitsintervall eines `Angebots` ist ein endliches Intervall. Es hat einen endlichen Wert für GültigAb und einen endlichen Wert für GültigBis.

`Angebote` können nur zugestellt werden, wenn GültigAb >= jetzt.

Lebenszyklus
Abbildung 49 zeigt den Lebenszyklus der Geschäftsklasse „Angebot“. Die Zustandsübergänge zwischen den verschiedenen Statuswerten eines Angebots werden durch Szenarien des Usecase „Angebot.verwaltenDurchAbsender“ (vgl. Abschnitt 5.4.3) ausgelöst.

Die möglichen Statuswerte sind "In Bearbeitung", "Zugestellt" und „Abgelaufen“. Manche Funktionen zur Bearbeitung eines Angebotes können nicht in jedem Status durchgeführt werden.

- Status „In Bearbeitung“.

  Definition: Das Angebot ist vorhanden, aber noch keinem Empfänger zugestellt.

  Das Angebot kann beliebig geändert werden, sofern nach der Änderung gilt "Angebot.GültigAb >= Jetzt“. Insbesondere kann das Angebot neu kalkuliert werden.

  Das Angebot kann gelöscht werden. Es ist dann nicht mehr rekonstruierbar.

  Das Angebot kann zugestellt werden, wenn "Angebot.GültigAb >= Jetzt“.

- Status „Zugestellt“.

  Definition: Das Angebot ist mindestens einem Empfänger zugestellt und "Angebot.GültigBis >= Jetzt“.

  Das Angebot kann übernommen und kopiert, aber nicht geändert - insbesondere nicht gelöscht oder erneut zugestellt - werden. Der Zustand zum Zustelltermin ist rekonstruierbar.

- Status „Abgelaufen“.

  Definition: Das Angebot ist mindestens einem Empfänger zugestellt und „Angebot.GültigBis < Jetzt“.

  Das Angebot kann weder geändert, noch gelöscht oder übernommen werden. Es kann kopiert werden.

Angebote können nicht storniert werden.

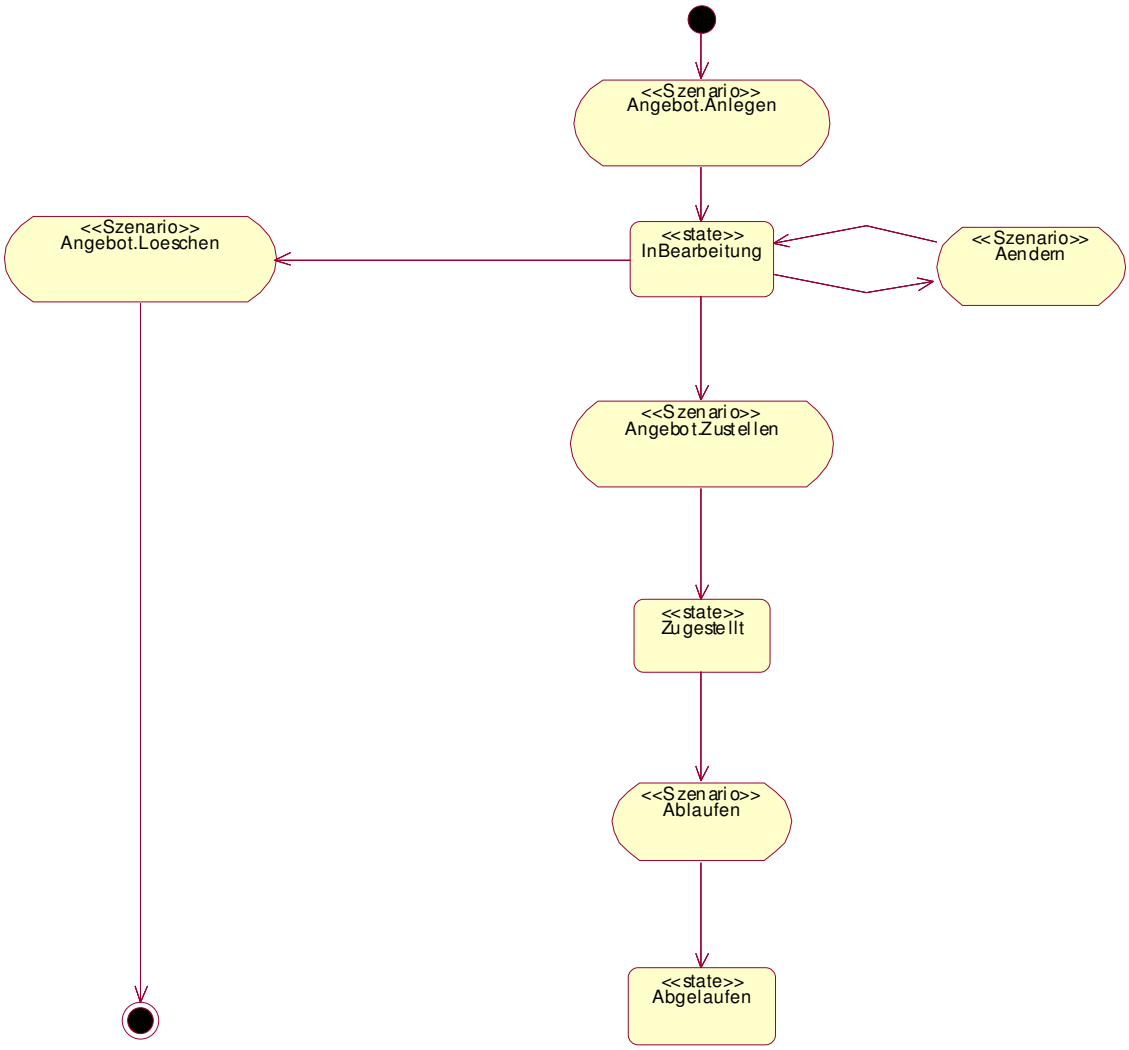


Abbildung 49: Lebenszyklus der Geschäftsklasse Angebot

Eine Geschäftsklasse ganz anderer Art zeigt das Inhaltsverzeichnis von Abbildung 50. Die Geschäftsklasse „Bepreisungsparameter“ enthält im wesentlichen Algorithmen zur Kalkulation von Angeboten.

Abbildung 50: Geschäftsklasse „Bepreisungsparameter“

5.4.3 Übersichtsebene: Beispiele von Usecasebeschreibungen

Die Operationen der Geschäftsklassen sind die Usecases. Abbildung 51 zeigt die Gliederung einer Usecasebeschreibung der Geschäftsklasse „Angebot“.

Alle Usecases werden nach diesem Template beschrieben.

Abbildung 51: Beispiel einer Usecasebeschreibung

Ein Ausschnitt aus der Beschreibung des Usecase „Angebot.VerwaltenDurchAbsender“:

Kurzbeschreibung

Mit diesem Anwendungsfall können `Mitarbeiter eines Händlers` auf der Vertriebsebene Händler `Händlerangebote` anlegen, ändern, löschen und an Empfänger der nächsttieferen Vertriebsebene (`Kunden`) zustellen.

Szenario Anlegen: `Händlerangebote` können auf verschiedene Arten angelegt werden:

- Ohne Kopieren: Ein Mitarbeiter eines Händlers legt ohne Vorlage ein Händlerangebot für die nächsttiefere Ebene an.
- Mit Kopieren: Ein Mitarbeiter eines Händlers hat ein empfangenes Marktangebot, ein eigenes Händlerangebot, eine empfangene Kundenanfrage, eine eigene Händleranfrage, eine empfangene Bestellung oder eine selbst eingegebene Bestellung durch Betätigen des Button „Kopieren“ in einem der entsprechenden Anwendungsfälle zu einem Händlerangebot kopiert. Dieses Händlerangebot wird in vorliegendem Anwendungsfall bearbeitet.
- Mit Übernehmen: Ein Mitarbeiter eines Händlers hat ein empfangenes Marktangebot oder eine empfangene Kundenanfrage durch Betätigen des Button „Übernehmen“ in einem der entsprechenden Anwendungsfälle zu einem Händlerangebot übernommen. Dieses Händlerangebot wird in vorliegendem Anwendungsfall bearbeitet.

Auswahl Anwendungsfall

1. Der Mitarbeiter eines Händlers aktiviert im Navigationsmenü den Menüpunkt „Angebote/Händlerangebote“.
2. Das System zeigt auf der Übersichtsmaske die Liste aller von diesem Händler angelegten Händlerangebote.

Szenario Anlegen

Variante „Ohne Kopieren“:

Maske „Übersichtsmaske“

1. Der Mitarbeiter eines Händlers betätigt den Button „Neuanlage“, um ein neues Händlerangebot ohne Benutzung einer Kopiervorlage zu erstellen.

Für alle Varianten:

Maske „Angebotskopf“

1. Der Mitarbeiter eines Händlers gibt die Kopfdaten des neuen Händlerangebotes ein.
2. Der Mitarbeiter eines Händlers wählt die Kalkulationsart und gibt den Kalkulationsfaktor ein. Wählt er die Kalkulationsart „EKP+“, so zeigt das System das Feld „Aufschlag“ an, in das der Aufschlag eingegeben werden muss. Wählt er die Kalkulationsart „KStufe“, so zeigt das System eine Tabelle aller marktspezifischen Kodierungsstufen an, in die der Mitarbeiter eines Händlers die $\text{Rabatte}_{\text{KStufe}}$ einträgt. Zur Vorbelegung der $\text{Rabatte}_{\text{KStufe}}$ kann er in das Feld „Händlerspanne Angebot“ einen Prozentwert eintragen. Die Felder $\text{Rabatte}_{\text{KStufe}}$ werden anschliessend automatisch durch das System PaSS befüllt. Wählt er die Kalkulationsart „Rabatt“, so zeigt das System das Feld „Rabatt“ an, in das der Rabatt eingegeben werden muss. Wählt er die Kalkulationsart „Standard“, zeigt das System kein zusätzliches Eingabefeld zur Eingabe eines Kalkulationsfaktors. Siehe hierzu auch Geschäftsklasse „Bepreisungsparameter“.

3. Der Mitarbeiter eines Händlers kann dem Händlerangebot Dateien als Attachments hinzufügen. Er betätigt dazu die Buttons „Durchsuchen...“, „Datei nutzen“ und ggf. „Datei(en) entfernen“.
4. Der Mitarbeiter eines Händlers kann den Button „Speichern“ betätigen.

Maske „Teileauswahl“

1. Sind dem Angebot bereits Empfänger zugeordnet, zeigt das System diese in der Liste „Auswahl“ an.
2. Sind dem Angebot noch keine Empfänger zugeordnet, zeigt das System die empfängerlose Teileliste in der Liste „Auswahl“ an.
3. Entweder sucht der Mitarbeiter eines Händlers Teile, für die das Händlerangebot gelten soll und überträgt sie durch Betätigen des Button „Übertragen“ aus der Liste „Suche“ in die Liste „Auswahl“. Oder er betätigt den Button „Teileliste aus System XXX“.
4. Sind Empfänger zugeordnet, kann der Mitarbeiter eines Händlers die Teileliste eines, mehrerer oder aller Empfänger expandieren.
5. Der Mitarbeiter eines Händlers kann sich zusätzliche Informationen zu einem Teil anzeigen lassen, indem er in der Spalte "Info" den Button "Info" betätigt.
6. Der Mitarbeiter eines Händlers kann sich die Verbaubarkeit aller angebotenen Teile durch Betätigen des Buttons „Teileverwendung“ anzeigen lassen.
7. Der Mitarbeiter eines Händlers kann den Button „Speichern“ betätigen.

Maske „Kunden“

1. Der Mitarbeiter eines Händlers sucht unter den Kunden seines Händlerbetriebes diejenigen, denen das Händlerangebot zugestellt werden soll und überträgt sie aus der Liste „Suche“ in die Liste „Auswahl“ durch Betätigen des Button „Übertragen“.
2. Der Mitarbeiter eines Händlers kann den Button „Speichern“ betätigen.

Maske „Kundengruppen“

1. Der Mitarbeiter eines Händlers sucht unter den Kundengruppen seines Händlerbetriebes diejenigen, denen das Händlerangebot zugestellt werden soll und überträgt sie aus der Liste „Suche“ in die Liste „Auswahl“ durch Betätigen des Button „Übertragen“.
2. Der Mitarbeiter eines Händlers kann den Button „Speichern“ betätigen.

Usecasebeschreibungen gehören zum Benutzermodell des Fachkonzepts. Sie zeigen und beschreiben die Funktionalität, welche das System dem Benutzer anbietet, das „Was?“. Sie beschreiben nicht, „wie“ das System diese Funktionalität realisiert, welche Methodenaufrufe durch eine Benutzeraktion ausgelöst werden.

Die Usecasebeschreibung einer Dialogapplikation nennt die möglichen Aktionen des Benutzers. Am Ende der *Benutzeraktionen* steht ein Satz wie „Der Benutzer kann den Button XYZ betätigen.“ Dadurch wird die *Systemaktion* ausgelöst. Diese wird nicht beschrieben, wohl aber

das dem Benutzer anschliessend auf einer Maske gezeigt Ergebnis, z.B. eine Kalkulation oder eine Erfolgs- oder Fehlermeldung.

Da die meisten Buttons auf den Masken fast aller Usecases vorkommen und ihre Betätigung häufig eine ähnliche Bearbeitung auslöst, werden die Buttons im Fachkonzept nur einmal an zentraler Stelle beschrieben, z.B. zu Beginn des Kapitels mit allen Usecases. Im jeweiligen Usecase werden dann nur noch die Besonderheiten des gegebenen Buttons beschrieben, ansonsten wird auf den allgemeinen Teil verwiesen.

In der Beschreibung des obigen Usecase „Angebot.VerwaltenDurchAbsender“ heißt es:

Button „Speichern“

Siehe auch Abschnitt "Grundsätze der Dialoge".

1. Szenario „Anlegen“: Das System legt unter Vergabe eines Schlüssels mit den eingegebenen Daten ein neues `Händlerangebot` an. Im Falle eines übernommenen `Marktangebotes` oder einer übernommenen `Kundenanfrage` wird auch eine Referenz zur Quelle angelegt.

Im genannten Abschnitt "Grundsätze der Dialoge" steht:

Button „Speichern“

Bei Betätigung des Button „Speichern“ verarbeitet das System alle Masken des Anwendungsfalles, nicht nur die zuletzt sichtbare. Bei Geschäftsobjekten, welche Masken für "Kopf", "Empfänger" und "Teile" haben, verarbeitet das System die Masken in der Reihenfolge "Kopf", "Empfänger", "Teile".

1. Das System führt eine syntaktische Prüfung durch:

   • Mussfelder gefüllt?

2. Im Szenario „Anlegen“ vergibt das System bei `Angeboten`, `Angebotsanfragen` und `Bestellungen` einen neuen Schlüssel. Bei allen anderen Geschäftsobjekten prüft das System, dass es noch keinen Satz mit gleichem Schlüssel gibt.

3. Das System führt folgende semantische Prüfungen durch:

   Bei den Geschäftsobjekten `Angebot`, `Angebotsanfrage` und `Bestellung` werden pro `Teil` geprüft:

   • Existenz.

   • Zuordnung zur `Kodierungstufe`.

   Bei den Geschäftsobjekten `Angebot` und `Bestellung` wird zusätzlich geprüft pro `Teil`:

   • Existenz und Gültigkeit aller Standardpreise.

   Beim Geschäftsobjekt `Angebot` wird zusätzlich geprüft pro `Teil`:

   • Einhaltung der durch `Mindestaufschlag` bzw. `Mindesthandelsspanne` festgelegten Plausibilitäten, siehe Geschäftsklasse „Bepreisungsparameter“.

   • Angebotspreis <= Standardpreis.

Variante „Prüfung OK“:

1. Das System führt ggf. einen Zustandsübergang des Geschäftsobjektes durch.
2. Das System speichert das gesamte Geschäftsobjekt.
3. Das System bringt eine Erfolgsmeldung, die zu bestätigen ist - der Focus liegt auf dem Button „OK“.
4. Der Benutzer quittiert die Meldung.
5. Das System berechnet eventuelle Preise.
6. Das System zeigt die zuletzt geöffnete Maske in aktualisiertem Zustand.

Variante „Prüfung Nicht OK“:

1. Das System bricht mit der ersten Maske ab, auf der ein Fehler gefunden wird. Das System zeigt die entsprechende Maske und weist auf den Fehler hin.
2. Der Benutzer korrigiert den Fehler und betätigt erneut den Button "Speichern".

5.4.4 Detailebene: Datenmodell

Als Detaillierung der Übersichtsebene modelliert das Fachkonzept gemäß der Multi-Skalen-Analyse (Abschnitt 3.4) noch eine zweite Ebene, die Detailebene.

In vorliegendem Projekt detailliert die Detailebene nur den Entitätsaspekt des Klassenmodells von Abbildung 46. Usecases wurden nicht weiter durch Operationen der Detailklassen detailliert. Auf der Detailebene wurde also kein Klassenmodell, sondern nur ein Datenmodell erstellt, d.h. es wurden keine detaillierenden Operationen definiert, da der Aufwand hierzu als zu hoch eingeschätzt wurde.

Das Datenmodell detailliert jede Geschäftsklasse einschließlich ihrer Beziehungen zu Nachbargeschäftsklassen von Abbildung 46 durch ein eigenes Entity-Relationship-Modell (ERM).

Abbildung 52 zeigt das Entity-Relationship-Modell der Geschäftsklasse „Angebot“. Alle ihre Detailklassen tragen das Präfix „Angebot.“ Die detaillierten Beziehungen zu den Detailklassen der Nachbarklassen enden bei Entitäten, welche mit dem Namen der Nachbarklasse präfixiert sind.

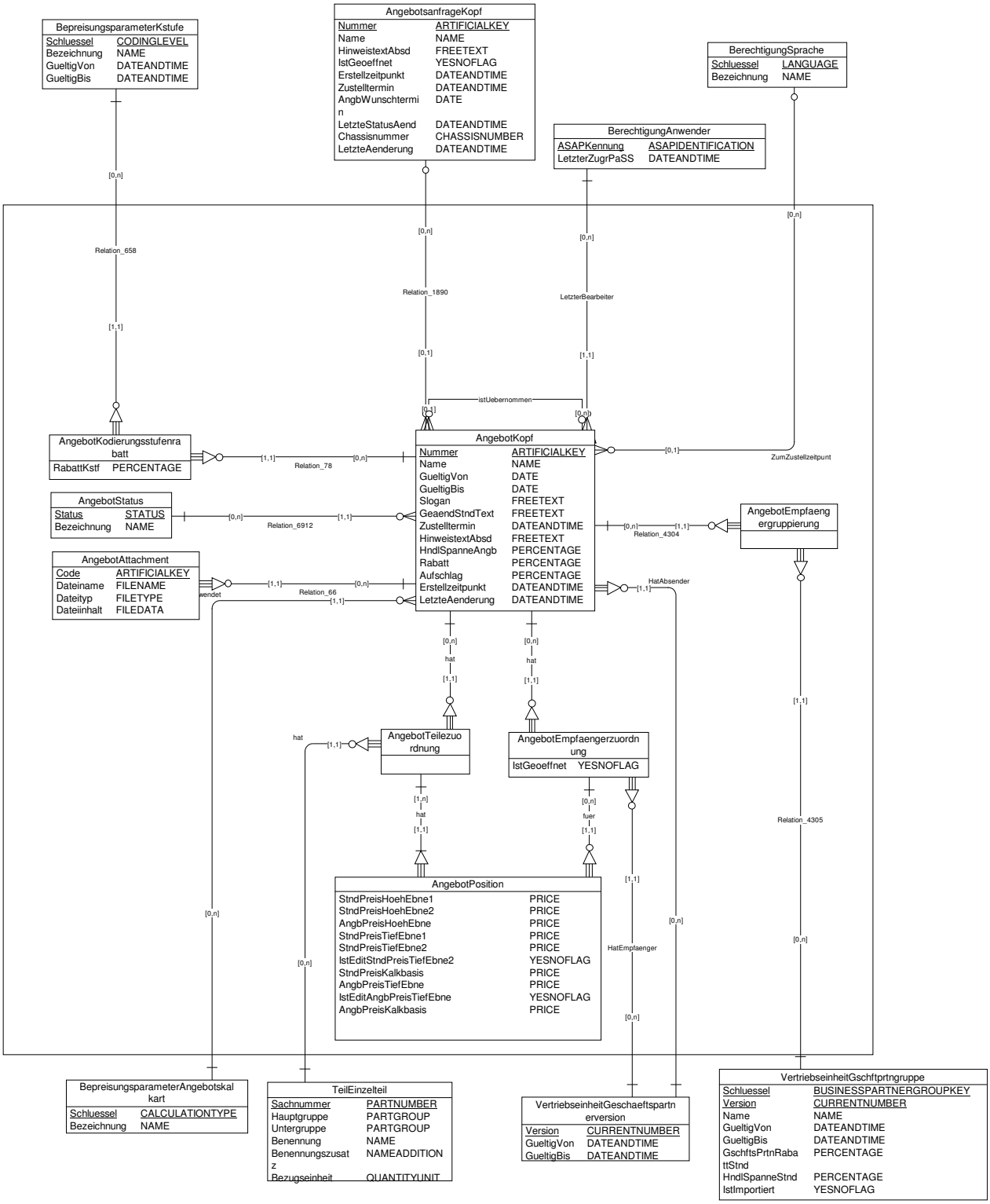


Abbildung 52: Detaillierung der Geschäftsklasse „Angebot“ und ihrer Beziehungen

Im Anschluss wird jede Entität mit ihren Attributen und dem jeweiligen fachlichen Datentyp spezifiziert.

Für jede Maske wird eine Zuordnungstabelle zwischen den Maskenfeldern und den Attributen des Datenmodells angelegt. Diese Zuordnung stellt sicher, daß alle Maskenfelder im logischen Datenmodell als Attribut berücksichtigt sind und daß umgekehrt alle Attribute des Datenmodells auch von den Maskenfeldern der Benutzeroberfläche gefüllt werden können. Ausnahmen betreffen Maskenfelder, die dynamisch, d.h. zur Laufzeit berechnet werden.

Abbildung 53 zeigt diese Konsistenz für die Maske „Angebotskopf“ des in Abschnitt 5.4.3 behandelten Usecase „Angebot.VerwaltenDurch Absender“.

Maskenfeld	Attribut des logischen Datenmodells
Überschrift. Angebotsname	*wird angezeigt, sobald der Benutzer eine Eingabe in das Feld Angebotskopf.Angebotsname gemacht hat AngebotKopf.Name
Angebotsnummer	AngebotKopf.Nummer
Angebotsname	AngebotKopf.Name
Slogan	AngebotKopf.Slogan
Status	GeschaeftsklasseAttributanzeige.Anzeige
Gültig von	AngebotKopf.GueltigVon
Gültig bis	AngebotKopf.GueltigBis
Kalkulationsart SELECT-Liste	• BepreisungsparameterAngebotskalkart. Schluessel • vorbelegt mit „EKP+“
Aufschlag	• AngebotKopf.Aufschlag • vorbelegt mit 0;
Rabatt	• AngebotKopf.Rabatt • vorbelegt mit 0;
Kodierungsstufe n	BepreisungsparameterKStufe.Schluessel
Rabatt Kodierungsstufe n	• AngebotKodierungsstufenrabatt. RabattKstf • vorbelegt mit 0;
Standardtext	• nicht editiert: VertriebseinheitStandardtext.Text • editiert: AngebotKopf.GeaendStndText • *vorbelegt mit VertriebseinheitStandardtext.Text
Hinweistext	AngebotKopf.HinweistextAbsd
Pfadangabe	–
Dateien	• AngebotAttachment.Dateiname • kateniert mit AngebotAttachment. Dateityp

Abbildung 53: Konsistenz zwischen Benutzeroberfläche und logischem Datenmodell

5.4.5 Detailebene: Schnittstellen

Für jede Schnittstelle des Systems wird ein Schnittstellenvertrag angelegt. Er folgt einem Template.

Für die Schnittstelle zwischen der Applikation des Projekts „Teilevertriebs“ mit dem System „XXX zum Import von Händlern“ sind u.a. Punkte des Template ausgefüllt:

Ausgetauschte Information
Die Schnittstelle beantwortet folgende Fragen:

Händler zu Markt XXX
Welche aktiven Händler gibt es zu einer gegebenen Vertriebsschiene „B1“ zu einem gegebenem Vertriebszweig „T“ und einer Adressverwendungsart „Standard“ in dem gegebenen Markt „XXX“?

- VERTRIEBSSCHIENE.VERTRIEBSSCHINE = „B1“ (PKW)
- VERTRIEBSZWEIG. VERTRIEBSZWEIG = „T“ (Teile)
- ADR_VERWENDUNGSART = „Standard“
- Markt: „XXX“

Position	Feldname	Pflichtfeld	Bemerkungen
1	*erste* Händlernummer	JA	default Händlernummer für „B1/T“
2	erste VP_Nummer	JA	-
4	Firma (Händlername)	JA	-
5	Firma_Zusatz (Händlername Zusatz)	JA	-
6	Strasse		-
7	Strasse_Postfach	JA	-
8	Ort	JA	-
9	Ortsteil	JA	-
10	PLZ	JA	-
11	Landesteil	JA	-
12	Land	JA	-
13	*zweite* Händlernummer	JA	-
14	*zweite* VP_Nummer	JA	-
...	...	...	-
n	*letzte* Händlernummer	JA	-
n+1	*letzte* VP_Nummer	JA	-
n+2	...	...	-

Ablaufbeschreibung
Der Anstoß der Schnittstelle erfolgt aus dem System XXX.

Die Liste aller Händler wird durch ein zu entwickelndes Programm in eine Datei exportiert.

5.5 Erfolgsfaktoren und Risiken kommerzieller Projekte

Entscheidend in der Analysephase eines Projekts sind nicht IV-Systeme, sondern der Umgang mit Menschen. Betriebswirtschaftliches Verständnis und Beherrschung der Methode werden als selbstverständlich vorausgesetzt. Typische Schwierigkeiten - oder Herausforderungen – in dieser Phase sind:

- Die Anwender arbeiten in ihrem Tagesgeschäft und haben nicht ausreichend Zeit für das Projekt
- Die Anforderungen der Anwender aus unterschiedlichen Abteilungen sind widersprüchlich, der Berater muß die Anwender immer wieder an einen Tisch bringen
- Der Umfang, in dem OOA eingesetzt wird, ist noch in Diskussion. Der Einsatz mehrerer Berater in einem gemeinsamen Projekt kann zu einem Streit über die "richtige" Methode führen.
- Um den Auftrag zu erhalten, hat der Auftragnehmer das Fachkonzept zu einem geringeren Preis kalkuliert als die eigenen Berater geschätzt haben.

  Hierzu eine Erfahrung aus der Praxis. Sie zeigt die Schwierigkeit, zu einer verläßlichen Kalkulation zu kommen: Im Rahmen einer Ausschreibung bittet der Projektmanager A die beiden Berater B und C, die mit dem Pflichtenheft des Kunden vertraut sind, ebenfalls den voraussichtlichen Aufwand für das Fachkonzept zu schätzen. Die drei Schätzungen klaffen weit auseinander. Die Schätzung des Beraters C ist doppelt so hoch wie die seines Kollegen B, in der Mitte von beiden liegt die Schätzung von A selbst. Das Softwarehaus bietet nun die Erstellung des Fachkonzepts zu einem Preis an, welcher dem mittleren, von A geschätzten Wert, entspricht. Am Ende der Analysephase – nach zwei Jahren – stellt sich heraus, daß die Schätzung von A jedoch zu niedrig war und die deutlich höhere Schätzung von C zutrifft. Daraufhin wird der Projektmanager A von seinem Vorgesetzten gerüffelt und gibt einen Teil der Schuld an C weiter: Dieser hätte wissen müssen, daß auf Basis der hohen Schätzwerte von C kein Auftrag akquiriert werden kann - der Berater B mit der niedrigsten Schätzung hat inzwischen das Projekt verlassen.
- Der Dienstleister eskaliert Probleme seines Kunden nicht an den Projektleiter seines Kunden, um diesem keine Schwierigkeiten zu machen.

Erfolgsfaktoren sind: Budget an Zeit und Geld vorhanden, der Kunde arbeitet mit, der Dienstleister beherrscht sein Handwerk.

Ein Projektleiter eines Kunden schreibt zu seinen Erfahrungen:

Aus Modellierungssicht ist eine Trennung von Fachkonzept und IT-Konzept an der Grenze zwischen Plattform-unabhängigem und Plattform-abhängigem Modell ideal. Zu erwarten, dass an dieser Stelle tatsächlich die Grenze gezogen wird, kann allerdings unrealistisch sein. Im Projektgeschäft bestimmen andere Kriterien:
Zeit und Kosten sind meist fester vorgegeben als inhaltlicher Umfang und Qualität. Wenn also Umfang oder Qualität vermindert werden müssen, fällt die Wahl normalerweise auf die Qualität: Mindere Qualität akzeptiert das typische Projektumfeld leichter als Abstriche am Projektumfang. Termine, Kosten und Umfang sind leicht messbar; Qualität nicht.

Deshalb kommt es beispielsweise vor, dass eine Anwendung termin- und kostengerecht eingeführt wird, für die weder die fachlichen Anforderungen geklärt sind, noch läuft sie stabil. Selbst bei der Schulung stürzt sie mehrmals ab. Dass der Anwender das Projekt nie entlastete, hatte keinen Einfluss auf den weiteren, positiven Karriereverlauf des Projektleiters.
Manche Fachkonzepte werden von Fachbereichs-Experten geschrieben. Die methodische fachliche Ausdetaillierung erwarten sie im IT-Konzept.
Um IT-Projekte zur Risiko-Verminderung klein zu halten, nehmen manche IT-Projekte Fachkonzepte aus anderen Projekten an. Die hier nötige fachliche Abstimmung erfolgt manchmal nicht in einem zusätzlichen Fachkonzept, sondern im IT-Konzept. Eine Diskussion darüber ist nicht immer möglich, z.B. weil der Ansprechpartner nicht unterscheidet zwischen fachlicher Abstimmung und der im IT-Konzept richtig platzierten Abstimmung der Fachlichkeit mit den IT-Belangen.

Noch ein ganz anderer Aspekt: Um zeitraubende Grundsatzdiskussionen zu vermeiden und um möglichst früh eine Vorstellung davon zu haben, in welche Richtung die IT-Umsetzung laufen wird, wird oft bereits vor der eigentlichen Fachkonzeption die Plattform ausgewählt. Das Modell im Fachkonzept ist dann nicht mehr Plattform-unabhängig.

6 Ausblick

Der wichtigste Arbeitgeber für Absolventen der Informatik sind Software-Firmen oder große Konzerne mit eigener IV-Abteilung.

6.1 Kosten, Zeit, Qualität

In kommerziellen Projekten werden andere Ansprüche an die Lösung eines fachlichen Problems gestellt als bei wissenschaftlichen Arbeiten an der Universität. Bei der Alltagsarbeit in einem Softwarehaus steht an erster Stelle nicht die wissenschaftliche Perfektion, sondern der vorgegebene Zeit- und Kostenrahmen: Die Lösung einer Aufgabe muß richtig sein, aber sie braucht nicht optimal zu sein - weniger ist hier manchmal mehr. Entscheidend ist, daß die Lösung in der vorgegebenen Zeit erfolgt. Der Stand der Forschung spielt für den Lösungsweg selten eine Rolle, brandneue Informationen liefern Newsgroups.

Qualität heißt, daß die Lösung alle Anforderungen aus dem Fachkonzept erfüllt. Im einzelnen kann das auf verschiedenen Wegen, mit unterschiedlichen Objekten, Operationen und Algorithmen erreicht werden. Daß die ausgelieferte Software dieses Kriterium erfüllt, ist eine Selbstverständlichkeit. Ein Softwarehaus, das hier Abstriche macht, handelt nicht seriös. Diese Qualität erreicht man durch ein geplantes Vorgehen, den Einsatz von Methoden und permanentes Testen auf den verschiedenen Ebenen der Integration.

Das Ringen um Termintreue ist eine ständige Herausforderung in allen Projekten. Termine werden in vielen Fällen nicht gehalten - ohne bösen Willen. Grund ist häufig die unterschätzte Komplexität, die man erst im Laufe des Projekts erkennt. Ein ganz anderer Grund für nicht gehaltene Terminzusagen sind bewußt zu niedrige Aufwandsschätzung in der Phase der Akquisition.

Wenn ein Softwarehaus Projekte ablehnt, weil der Kunde den Aufwand nicht in der geschätzten Höhe zahlen will, zeugt das vom Bewußtsein der eigenen Stärke auf dem Markt und ist ein gutes Zeichen für das Unternehmen.

6.2 Information über einen potentiellen Arbeitgeber

Die beste Möglichkeit, potentielle Arbeitgeber unverbindlich aus eigener Erfahrung kennenzulernen, sind zwei oder drei Ferienjobs bei verschiedenen Softwarefirmen oder sogar ein Praktikumssemester. Die wichtigste Quelle aus zweiter Hand sind natürlich die Informationen von Bekannten, die bereits in einer Firma fest angestellt sind. Außerdem gibt es gute Ratgeberliteratur zur Prüfung des Arbeitgebers.

Über einen potentiellen Arbeitgeber in der IT-Branche sollten insbesondere folgenden Informationen gesammelt werden:

- Wie lange existiert das Unternehmen schon? Wem gehört es?

  Wirtschaftliche Kennzahlen enthält im Falle einer AG der Geschäftsbericht. Man kann ihn bei der Firma anfordern.

- Wer sind die wichtigsten Kunden? Beteiligt sich das Unternehmen an Rüstungsprojekten?

- Wie stellt sich das Unternehmen auf seinen Internetseiten dar? Was bleibt übrig, wenn man die Schlagworte abzieht? Welche Methoden werden zum Software Engineering eingesetzt? Gibt es Publikationen von Mitarbeitern des Unternehmens?
- Wie schnell erhält man Antwort auf seine Bewerbung? Wie hoch ist die Fluktuation unter den Mitarbeitern, wer geht schon in der Probezeit?
- Wie geht das Unternehmen mit Überstunden um: Gibt es Überstunden in nennenswertem Maße, werden sie stillschweigend erwartet, werden sie als Freizeit abgegolten oder finanziell vergütet?

  Für das Unternehmen sind Überstunden der vorhandenen Mitarbeiter in der Regel billiger als die Einstellung zusätzlicher neuer Mitarbeiter.

Job und Karriere: Karriere in einem Softwarehaus bedeutet Arbeit als Manager mit Budget- und Personalverantwortung und der Pflicht zur Akquisition. Als Annehmlichkeiten gibt es Firmenfahrzeug, Beiträge zur Lebensversicherung, Aktienoptionen im Falle einer AG. Ab einer gewissen Stufe des Aufstiegs ist Mehrarbeit im Gehalt inbegriffen. Das bedeutet: Nicht 8 Stunden Arbeit am Tag, sondern 10 Stunden. Eine Karriere als Experte für Softwaremethoden ist eine ganz seltene Ausnahme.

6.3 "Habe Mut, dich deines eigenen Verstandes zu bedienen!"

Kant propagierte diese Aufforderung als Wahlspruch der Aufklärung im 18. Jahrhundert. Wer diese Aufforderung heute in seine Berufspraxis der IT-Branche umsetzen will, muß zunächst kritisch mit der Sprache umgehen. Nur so kann man Immunität erwerben gegen die Schlagworte, mit denen Unternehmen potentielle Kunden anziehen und ihre Mitarbeiter führen und motivieren wollen.

Eine Auswahl plakativer Leitsätze von Unternehmen der IT-Branche zeigt Abbildung 54. Diese Leitsätze stammen aus dem Jahre 2002 und müssen in regelmäßigen Abständen fortgeschrieben werden. ☺

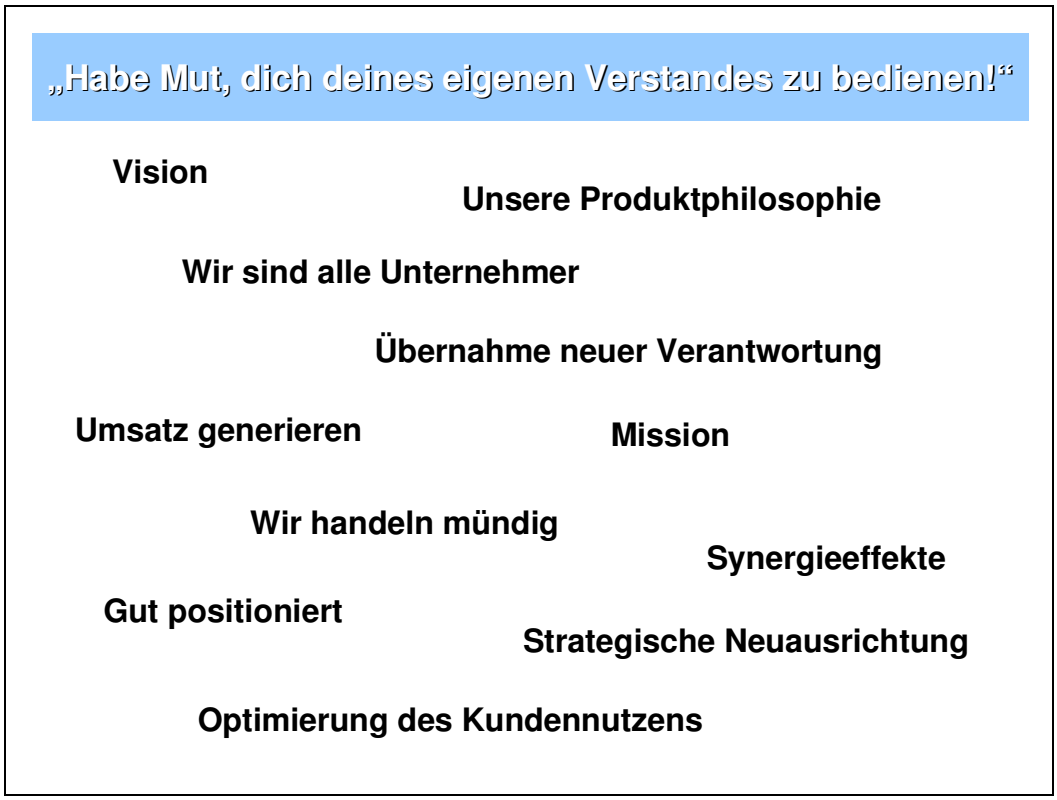


Abbildung 54: Schlagworte aus der IT-Branche im Jahr 2002

Wer will, kann mitspielen auf dieser Klaviatur der Sprechblasen - aber mit einem ironischen Augenzwinkern. Nüchterne und kritische Beiträge und Analysen aus der IT-Branche stellen die empfehlenswerten Kolumnen von Jens Coldewey in der Zeitschrift [SIG] dar.

Kein leeres Schlagwort ist der Leitsatz:

> Der Kunde steht im Mittelpunkt.

Dieser Satz ist die wichtigste Richtlinie für ein Dienstleistungsunternehmen wie ein Softwarehaus.

7 Literatur

7.1 Wirtschaftsinformatik

Ott, Hans Jürgen: Betriebswirtschaftslehre für Ingenieure und Informatiker. München 1995
Scheer, August-Wilhelm: Wirtschaftsinformatik. Springer, Berlin et al., 5. Auflage 1994
Scheer, August-Wilhelm: ARIS - vom Geschäftsprozeß zum Anwendungssystem. ARIS - Modellierungsmethoden, Metamodelle, Anwendungen. 2 Bd. Springer, Berlin et al., 3. Aufl. 1998

7.2 Software Engineering

Brössler, Peter; Siedersleben, Johannes (Hrsg.): Softwaretechnik: Praxiswissen für Software-Ingenieure. Hanser, München Wien, 2000
Denert, Ernst: Software-Engineering. Springer, Berlin et al. 1991
Siedersleben, Johannes: Moderne Software-Architektur. Heidelberg 2006

7.3 OOA

Kappel, Gerti; Schrefl, Michael: Objektorientierte Informationssysteme: Konzepte, Darstellungsmittel, Methoden. Springer Wien et al. 1996

7.4 OOD

Orfali, Robert; Harkey, Dan; Edwards, Jeri: Client/Server Survival Guide. Wiley, New York et al. 3. rd Edition 1999
[SIG] *SIGS-DATACOM GmbH*: OBJEKTspektrum.. Erscheint 2-monatlich

7.5 OOP

Flanagan, David; Farley, Jim; Crawford, William; Magnusson, Kris: Java Enterprise in a Nutshell. O'Reilly, Peking et al. 1999
Monson-Haefel, Richard: Enterprise JavaBeans. O'Reilly, Peking et al. 2.ed. 2000

7.6 Petri-Netze

[Rei1986] *Reisig, Wolfgang:* Petrinetze: Eine Einführung. Springer, Berlin et al. 2. Aufl. 1986
Starke, Peter: Analyse von Petri-Netz-Modellen. Stuttgart, 1990
Desel, Jörg; Esparza, Javier: Free Choice Petri Nets. Cambridge, 1995
[Jen1992] *Jensen, Kurt*: Coloured Petri Nets. Basis Concepts, Analysis Methods and Practical Use. 3 Bände, Springer, Berlin et al. 1992ff.

7.7 UML

Breu, Ruth: Objektorientierter Softwareentwurf: Integration mit UML. Springer, Berlin et al. 2001
[BRJ1999] *Booch, Grady; Rumbaugh, James; Jacobson, Ivar*: The Unified Modeling Language User Guide. Addison-Wesley, Reading Mass. et al. 1999
Fowler, Martin; Scott, Kendal: UML - konzentriert: die neue Standard-Objektmodellierungssprache anwenden. Addison-Wesley, Bonn 1998

[FR1999] *France, Robert; Rumpe, Bernhard* (Eds.): <<UML>> '99 - The Unified Modeling Language. Beyond the Standard. Second International Conference, Fort Collins, CO, USA, October 1999. Lecture Notes in Computer Science 1723. Springer, Berlin et al. 1999
Hitz, Martin; Kappel, Gerti: UML@Work. Von der Analyse zur Realisierung. dpunkt, Heidelberg 1999
[JRH2004] *Jeckle, Mario; Rupp, Chris; Hahn, Jürgen; Zengler, Barbara; Queins, Stefan*: UML 2.0, glasklar. Hanser, München, 2004
Oestereich, Bernd: Objektorientierte Softwareentwicklung. Analyse und Design mit der Unified Modeling Language. Oldenbourg, München et al., 4. Auflage 1998

7.8 BPEL

[BPEL2006] Web Services Business Process Execution Language Version 2.0. Committee Draft, 17th May, 2006. http://www.oasis-open.org/apps/org/workgroup/wsbpel/ Abruf 6.11.2006

[Jur2006] *Juric, Matjaz*: A Hands-on Introduction to BPEL. http://www.oracle.com/technology/pub/articles/matjaz_bpel1.html. Download 6.12.2006