Objektorientierte Modellierung der Kosten in Hochbauprojekten

Von der Fakultät Architektur, Bauingenieurwesen und Stadtplanung
der Brandenburgischen Technischen Universität Cottbus zur Erlangung des akademischen Grades eines Doktor-Ingenieurs
genehmigte Dissertation

vorgelegt von

Dipl.-Ing. Abed Alzoobi
aus Aljizeh, Syrien

Gutachter: Prof. Dr.-Ing. habil. Dieter Weitendorf
Prof. Dr.-Ing. Peter Osterrieder
Prof. Dr.-Ing. Eberhard Petzschmann
Prof. Dr.-Ing. Karl Beucke

Tag der Disputation: 19.11.2003

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als Doktorand am Lehrstuhl Bauinformatik, Fakultät Architektur, Bauingenieurwesen und Stadtplanung der Brandenburgischen Technischen Universität Cottbus.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Eberhard Petzschmann für seine wertvollen Anregungen sowie die hilfreichen Hinweise, die wesentlich zur Gestaltung dieser Arbeit beigetragen haben.

Zu Dank verpflichtet fühle ich mich ebenso Herrn Prof. Dr. rer. nat. Claus Lewerentz und Herrn Dr.-Ing. Hans-Gerd Köhler vom Lehrstuhl Software-Systemtechnik der BTU-Cottbus sowie Herrn Prof. Dr. rer. nat. habil. Bernhard Thalheim, Lehrstuhl Datenbank- und Informationssysteme der BTU-Cottbus für die hohe Diskussionsbereitschaft.

Ferner möchte ich Herrn Prof. Dr.-Ing. Werner Voigt und Herrn Prof. Dr. rer. nat. habil. Karl-Heinz Schlüßler für ihre Mühe des Korrekturlesens dieser Arbeit herzlich danken.

Mein Dank gilt auch meinen Eltern, die mich geprägt haben. Meiner Frau und meinen Kindern danke ich für die moralische Unterstützung und das entgegengebrachte Verständnis.

Cottbus, im November 2003

Abed Alzoobi

Zusammenfassung

In dieser Arbeit werden Ansätze zur Softwareentwicklung auf Objektorientierter Basis im Bereich der Kosten in Hochbauprojekten beschrieben.
Die Konzepte der Objektorientierten Technologie zur Softwareentwicklung werden in dieser Arbeit umfassend eingesetzt.
Zur Modellierung des Systems wird die Unified Modeling Language (UML) eingesetzt. Die bedeutenden Gesichtspunkte dieser Modellierung sind die Entwicklung des Objektmodells in der Analysephase und deren Verfeinerung in der Entwurfsphase. Dieses Objektmodell bildet die Basis bei der Softwareentwicklung im bereich der Kosten in Hochbauprojekten. Im Rahmen dieser Modellierung werden auch die Aspekte der Soft- und Hardwareverteilung spezifiziert. Ferner werden Konzepte zur Implementierung der Ergebnisse der Modellierung erläutert. Hier wird das Objektmodell des Systems in die Programmiersprache Java übersetzt.

Abstract

This work describes the approaches to software development on an object oriented basis for cost in structural engineering projects. The concepts for object oriented technology for software development will be used extensively. The modeling of the system is in the Unified Modeling Language (UML).
The significant points in this modeling are the development of the object model in the analysis phase as well as its refinement in the design phase. This object model establishes the basis for software development pertaining to costs for structural engineering projects. In the context of this modeling, aspects of software and hardware distribution are specified.
Moreover, concepts about the implementation of the results of the modeling are discussed. The object model is translated into the Java programme-language.

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangssituation

Der wachsende Einsatz des Internets macht Konzepte wie Enterprise Application Integration (EAI) überflüssig. Hier ist die Realisierung einer Internet Application Integration unumgänglich [Gilp99]. Die wichtigsten Anforderungen diesbezüglich sind:

- Unterstützung der Internet-Technologien und heterogenen Plattformen,
- Integration von Komponenten wie z. B. Enterprise JavaBeans (EJB) und von heterogener Namens- und Verzeichnisdienste,
- Komponentenzentrierte Architektur und Leichtgewichtige Klienten sowie
- flexible Kopplung durch Message-Broker, insbesondere dem Java Nachrichtendienst.

Datenorientierte Techniken zur Abbildung von Daten in Relationalen Datenbanken wie die Entity-Relationship-Modellierung (kurz: ERM) werden zwar immer noch zur Modellierung von baubetrieblichen Daten eingesetzt, sie werden aber in einem steigenden Tempo von der Objektorientierten Modellierung abgelöst. Die Objektorientierte Modellierung hat sich hierfür im Laufe der Zeit als die besser geeignete Technik zur Modellierung von solchen Daten erwiesen. Bis zur Entwicklung der UML gab es auf dem Gebiet der Objektorientierten Softwareentwicklung keine einheitlichen Verfahrensweisen, die sich auf einer gemeinsamen Sprache oder Notation aufbauen.

Die Vorteile der Objektorientierten Technologie können jedoch nur dann erreicht werden, wenn die Softwareentwicklung auf der Basis eines effizienten Vorgehensmodells erfolgt. Durch die Analyse von Bereichen der realen Welt des Bauwesens und die Abbildung dieser Teile in Modellen, können komplexe Aufgaben effektiver gelöst werden. Die Qualität der entwickelten Modelle hängt stark von der Integrität dieser Analyse und Darstellung ab.
Die Methoden der klassischen Systementwicklung wie z. B. die strukturierte Analyse und die ERM haben sich bei der Abbildung von Modellen der realen Welt zwar bewährt. Der Einsatz der Objektorientierung als Methode zur Entwicklung von solchen Modellen hat aber zu einer bedeutenden Verbesserung bei der Abbildung geführt.

Die Entwicklung von effizienten Softwarelösungen für den Hochbau setzt ein umfassendes Gebäudemodell voraus. Der Standardisierungsgrad der Objekte im Hochbau ist wesentlich gestiegen, insbesondere nach der Entwicklung der Industry Foundation Classes (IFC).

Im Bereich der Modellierung der baubetrieblichen Prozesse existieren bereits ganzheitliche Betrachtungen, die im Kapitel 2.8 der vorliegenden Arbeit näher beschrieben werden.
Jedoch sind die Modellierungseinsätze in diesem Bereich wegen der Konzentration der Forschung auf andere Problembereiche des Bauwesens noch sehr in den Anfängen. Auch konzentrieren sich die vorhandenen Modellierungseinsätze hauptsächlich auf die Planungsphase eines Bauprojektes.

Zur Entwicklung der Anwendungssoftware Aristoteles der Hochtief Software GmbH wird die Anwendungssoftware SAP R/3 von der SAP AG als Basis für die Betriebswirtschaft verwendet. Dabei wurde SAP R/3 von Hochtief Software speziell für die Anforderungen des Bauwesens modifiziert und weiterentwickelt. Bekanntlich wird die Anwendungssoftware SAP R/3 von SAP AG datenorientiert (auf der Basis des ER-Modells) entwickelt.
Die Programmiersprache ABAP, die in SAP/R3 verwendet wird, beinhaltet nur ab der Version ABAP 4 Erweiterungen, welche den Einsatz der Konzepte der Objektorientierten Techno-

logie erlauben. Diese Erweiterungen wurden jedoch bei der Entwicklung von Aristoteles nicht berücksichtigt, weil sie z. Z. der Entwicklung von Aristoteles nicht zur Verfügung standen.

Die Verfahrensweisen bei der Programmierung der Anwendungssoftware ARRIBA *bauen*+ von RIB SOFTWARE AG sind nach Herstellerangaben Objektorientiert. Ebenfalls nach Objektorientiertem Schema ist das ARRIBA-Datenmodell konzipiert und aufgebaut. ARRIBA verwendet deshalb als Datenbank ausschließlich das Objektorientierte Datenbankmanagementsystem ObjectStore des Herstellers Excellon (vormals Object Design).
Diese Verfahrensweisen sowie das ARRIBA-Datenmodell sind aber nicht frei verfügbar. Damit besteht keine Möglichkeit diese zu untersuchen und weiterzuentwickeln.

Erfahrungen im Bereich der Objektorientierten Modellierung der Strukturen zur Kalkulation der Kosten in Hochbauprojekten, wie sie in dieser Arbeit umfassend ausgeführt wurden, liegen noch nicht vor.

1.2 Zielsetzung

Eine effiziente Strategie zur Softwareentwicklung im Bereich der Bauwirtschaft setzt voraus, dass die Softwareentwickler auf effiziente Architekturen zurückgreifen können. Zur Schaffung solcher Architekturen sollen die Systeme anhand von wirkungsvollen Konzepten modelliert werden. In diesem Zusammenhang bietet sich die Objektorientierung als geeignete Technologie an. Diese Technologie bietet die zweckmäßigen Konzepte zur Modellierung sowohl der statischen als auch der dynamischen Aspekte des Systems zur Kalkulation der Kosten in Hochbauprojekten, sowie deren effiziente Implementierung auf Objektorientierter Basis.

Zur Schaffung einer stabilen Basis für die Softwareentwicklung im Bereich der Kalkulation der Kosten im Hochbau ist es notwendig:

- vereinfachte und IT-taugliche Strukturen und Verfahrensweisen und
- eine systematische Spezifizierung dieser Strukturen und Verfahrensweisen

zu entwickeln bzw. durchzuführen.

Zur Erhöhung der Performance bei der Modellierung in diesem Bereich werden folgende Aspekte in der vorliegenden Arbeit berücksichtigt:

- Erstellung von übersichtlichen, zusammenwirkenden Anforderungsanalysemodellen in der Analysephase;
- Bearbeitung von Entwurfsmodellen unter Berücksichtigung von solchen Konzepten wie Entwurfsmuster, die den Einsatz dieser Modelle effizienter macht;
- Erstellung von Implementierungsmodellen, in denen die Konzepte wie die komponentenbasierte Softwareentwicklung und Network-Computing, einschließlich des Einsatzes von CORBA und Java-Technologie berücksichtigt werden;
- Darstellung der geeigneten vernetzt-kooperativfähigen und internetbasierten Architektur eines verteilten Systems, in dem die Soft- und Hardware verteilt werden.

Der Produktlebenslauf im Hochbau kann folgendermaßen gegliedert werden (Abb. 1.1):

- Angebotsphase zur Bestimmung der Anforderungen des Bauprodukts.
- Entwurfsphase zur Erstellung von detaillierten Angaben über das Bauobjekt, welches an dem gewählten Ort erstellt wird.
- Planungsphase zur Festlegung der Verfahren zur Realisierung des Entwurfs als vollständiges Bauobjekt.

- Realisierungsphase zur Verwirklichung des Entwurfs zu einem fertigen Bauobjekt entsprechend der Planungskriterien.
- Betriebsphase zur Betreibung des Bauobjekts in der vorgeschlagenen Art und Weise und dessen Instandhaltung in dem möglichen Zustand in dem es realisiert wird.

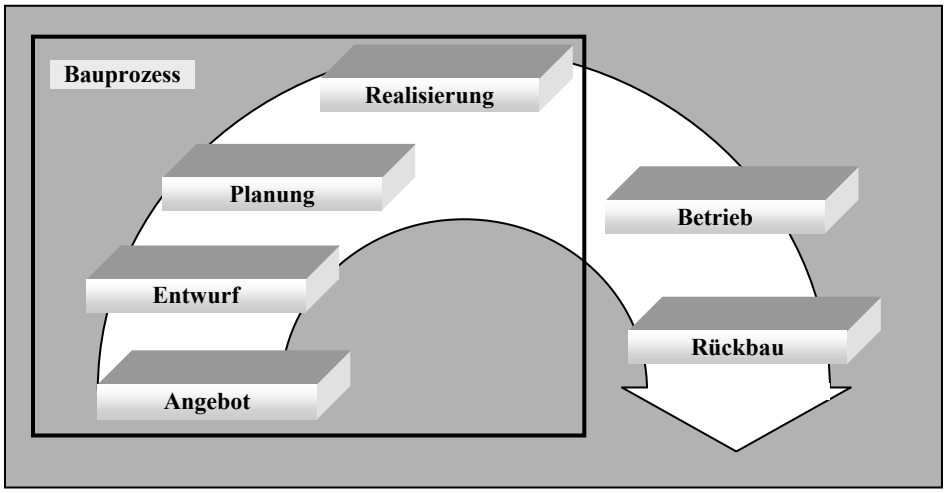


Abbildung 1.1: Produktlebenslauf im Hochbau

Das System der Kalkulation der Kosten in Bauprojekten des Hochbaus beinhaltet die Strukturen der kostenbezogenen Sachverhalte und deren Beziehungen zu den anderen Bestandteilen des Gesamtsystems. Bei der Modellierung in der vorliegenden Arbeit wird das Teilsystem Bauprozess vom Gesamtsystem des Produktlebenslaufs im Hochbau abgegrenzt.

Eine korrekte Kalkulation der Kosten in Bauprojekten ist ein wichtiges wirtschaftliches Ziel. Dieses Ziel kann nur durch eine effiziente, Objektorientierte Modellierung der Kalkulation der Kosten erreicht werden. Eine solche Modellierung, bei der auch der Bedarf nach mehr Investitionen in der frühen Phasen der Softwareentwicklung wie die Anforderungsanalyse wahrgenommen wird, soll auf festen Architekturen und nach stabilen Regeln erfolgen. In diesem Kontext sind der Einsatz und die Kombination solcher Techniken wie *Unified Modeling Language* (UML) und Entwurfsmuster hilfsreich.

Zu einer effizienten Modellierung der Kalkulation der Kosten in Hochbauprojekten, wie sie innerhalb der vorliegenden Arbeit durchgeführt wird, gehört einerseits die korrekte Darstellung der Kosten in Bezug auf die Bauobjekte und Ressourcen und andererseits die Integration der Prozesse der Baukalkulation in den gesamten Bauprozess innerhalb eines Bauprojektes durch wohldefinierte Schnittstellen. Ferner sollen im Rahmen dieser Modellierung die Aspekte der Soft- und Hardwareverteilung spezifiziert werden. In diesem Zusammenhang sollen Verteilungsarchitekturen bezüglich ihrer Tauglichkeit untersucht werden.

Der vordergründige Anspruch der vorliegenden Arbeit besteht darin, einen Ansatz zur Strukturierung der Daten des Systems der Kalkulation der Kosten in Hochbauprojekten zu beschreiben und hierfür die notwendigen Anforderungen und Konzepte der Objektorientierten Technologie einzusetzen und ihre Tauglichkeit diesbezüglich festzustellen. Dementsprechend werden die Datenstrukturen des Systems der Kalkulation der Kosten in Hochbauprojekten analysiert. Außerdem werden die Objektklassen und Klassenhierarchien, die für die integrale

Bearbeitung der baubetrieblichen Prozesse zur Baukalkulation notwendig sind, festgelegt. Es werden auch die dynamischen Aspekte dieser Prozesse analysiert und modelliert.

Durch den Einsatz der Konzepte der Objektorientierten Technologie in diesem Bereich wird einerseits die Effizienz gesteigert, andererseits wird die Durchgängigkeit der Modellierung der Strukturen erlangt.

Zum anderen soll mit der vorliegenden Arbeit eine Grundlage für Diskussionen gegeben werden, da sie am Berührungspunkt von Baubetrieb/Bauwirtschaft und Informatik angesiedelt ist.

Mit der Objektorientierten Modellierung der Kalkulation der Kosten innerhalb der vorliegenden Arbeit wird ein Objektmodell zur optimalen Beschreibung der Objekte des Systems und deren statischer und dynamischer Zusammenhänge entwickelt. Bei der Beschreibung dieser Objekte wird die Erhaltung der Integrität des Gesamtsystems berücksichtigt. Daher werden solche Objekte, die auch Objekte anderer Bestandteile des Gesamtsystems sind, nicht nur kostenbezogen beschrieben, sondern auch entsprechend ihrer Rolle im Gesamtsystem. Diese Vorgehensweise spielt eine wichtige Rolle bei der Integration der Teilsysteme im Gesamtsystem.

Die Ergebnisse der Modellierung in der vorliegenden Arbeit führen u. a. zur Erhöhung des Standardisierungsgrades der Objekte innerhalb des Systems der Baukalkulation in Hochbauprojekten. Damit wird einen Beitrag zu einer effizienten Modellierung der Baukalkulation im Hochbau als Grundlage zur Entwicklung und Integration von Softwarelösungen in diesem Bereich geleistet.

1.3 Inhaltsübersicht

Die vorliegende Arbeit, welche mit einer Einführung in **Kapitel 1** beginnt, ist in neun Kapitel und einen **Anhang** in dem die UML-Notation dargestellt wird, gegliedert.

In **Kapitel 2** wird der gegenwärtige Stand der Wissenschaft auf dem Gebiet der rechnergestützten Modellierung eingeführt. Hier werden zuerst die Begriffe System und Modell definiert. Ferner wird eine Analyse der Forschungen und Normen auf dem Gebiet der Modellierung dargestellt. Dabei werden die Bereiche geometrische Modellierung, Produktmodellierung, STEP als Norm zur Produktmodellierung, das Produktmodellierungskonzept von STEP im Bauwesen, die Objektorientierte Modellierung und Produktmodellierung im Bauwesen, der EDV-Einsatz bei der Ermittlung der Kosten sowie die Tendenzen zur Standardisierung der Objektorientierten Technologie erläutert. Der Stand der Forschungen und Normen auf dem Gebiet der Objektorientierung wird in diesem Kapitel dargestellt, wobei die Basiskonzepte dieser Technologie gesondert im Kapitel 3 dargestellt werden. Die in diesem Kapitel erworbenen Kenntnisse bilden die Basis für die weitere Bearbeitung in den nächsten Kapiteln.

Im **Kapitel 3** werden die Grundlagen der Objektorientierten Softwareentwicklung dargestellt. Hier werden zuerst die datenorientierten Modellierungskonzepte, wie das Entity-Relationship-Model und das 3-Ebenenkonzept von ANSI/X3/SPARC, den Ausgangspunkt zur Objektorientierten Modellierung bilden. Danach werden die Basiskonzepte der Objektorientierung dargestellt. Ferner wird in diesem Kapitel die Objektorientierte Analyse beschrieben, welche sich hier in Anforderungsmodellierung und Analysemodellierung gliedert. Die Analysemodellierung gliedert sich ihrerseits in Strukturmodellierung und dynamische Modellierung.

Weiter wird der Objektorientierte Entwurf dargestellt. Dieser unterteilt sich in Systementwurfs- und Detailentwurfsmodellierung. Anschließend werden in diesem Kapitel die Aspekte der Objektorientierten Implementierung des Entwurfs dargestellt. Dabei werden die Objektorientierte Programmierung und die Objektorientierte Datenhaltung erläutert. Die Darstellung der Objektorientierten Datenhaltung unterteilt sich hier in Objektorientierte Datenbankmodelle, Objektorientierte Datenbanksysteme und Objektrelationale Datenbanksysteme.

Im **Kapitel 4** wird das System der Kosten in Hochbauprojekten dargestellt. In diesem Zusammenhag wird zuerst eine allgemeine Beschreibung des Systems unternommen, in der die Datenbasis im Hochbau klassifiziert und die Grundbegriffe der Kosten in Hochbauprojekten beschrieben werden. Ferner wird das System der Kosten in Hochbauprojekten vom Gesamtsystem des Hochbaus abgegrenzt und anschließend beschrieben.

Im **Kapitel 5** wird die Objektorientierte Analyse des Systems der Kosten in Hochbauprojekten durch dem Einsatz der UML ausgeführt. Zuerst wird hier die Vorgehensweise zur Modellierung des Systems der Kosten dargestellt. Danach wird das Anforderungsmodell anhand von Anwendungsfall- und Aktivitätsdiagrammen beschrieben. Weiterhin wird die System- und Modellhierarchie festgelegt. Als Ausgangspunkt zur Erstellung des Analysemodells wird eine Beschreibung des Systems der Baukalkulation durchgeführt. Danach wird bei der Erstellung des Analysemodells die Struktur und Dynamik des Systems anhand von Klassen- bzw. Sequenzdiagrammen modelliert.

Im **Kapitel 6** wird aufbauend auf dem Analysemodell der Objektorientierte Entwurf des Systems der Baukalkulation ausgeführt. Hier werden zunächst die angewendeten Konzepte beschrieben. Die Java-Umsetzung einiger UML-Konzepte, die wiederholt im Rahmen der Modellierung in der Arbeit vorkommen, werden nachfolgend beschrieben. Ferner werden die Klassendiagramme der Analysephase verfeinert. Danach erfolgt die Überarbeitung und Partitionierung des Strukturmodells der Analysephase. Hier wird das Strukturmodell so überarbeitet, dass Erweiterungen und Änderungen der Funktionalität des Systems möglichst begrenzte Wirkung haben sollen. Bei der Partitionierung des Strukturmodells handelt es sich hier um dessen Darstellung in Paketen.
In diesem Kapitel werden ferner die Aspekte der Verteilung und Kommunikation des Modells erläutert. Hierzu wird die Architektur des geeigneten verteilten Systems dargestellt und die Verteilung und Kommunikation des Modells anhand von Komponenten- und Verteilungsdiagrammen beschrieben.

Im **Kapitel 7** werden die Aspekte der Implementierung des Systems erläutert. Hierzu wird die verfeinerte Version des Strukturmodells der Entwurfsphase als Grundlage zur Erstellung des Codes in Java verwendet. Außerdem werden die Gesichtspunkte der Realisierung der spezifizierten Persistenzmechanismen dargestellt.

Im **Kapitel 8** werden Anwendungsbeispiele zur Implementierung des Systems der Angebotskalkulation vorgestellt.

Im **Kapitel 9** werden einige Schlussbetrachtungen wie Zusammenfassung, Beurteilung und einen Ausblick auf weitere Entwicklungen dieser Arbeit erörtert.

2 Stand der Wissenschaft auf dem Gebiet der rechnergestützten Modellierung

2.1 Definition der Begriffe System und Modell

Ein **System** wird in [Broc95] als Sammlung von zueinander in Beziehung stehenden Elementen betrachtet. Jedes System, das seine Elemente nach bestimmten Regeln einsetzt, beinhaltet Daten, die direkt oder nur mit Messgeräten erfasst werden können. Systeme können in Teilsysteme gegliedert werden, wobei diese Gliederung nach bestimmten Kriterien erfolgen soll.

Der Begriff **Modell** wird in den unterschiedlichen wissenschaftlichen Richtungen benutzt. Ein Gegenstand A ist ein Modell eines Gegenstands B, falls ein Betrachter C Gegenstand A zur Beantwortung von Fragen über Gegenstand B benutzen kann [Mins68]. Ein Modell ist in [Booch99] als eine Vereinfachung der Realität des zu modellierenden Systems betrachtet. Nach Mochel [Moch93] repräsentiert ein Modell ein reales System von einem bestimmten Standpunkt. Ein Modell ist in [Stac73] als eine Abbildung eines Systems beschrieben. Dieses Modell hat eine identische oder ähnliche Struktur wie das zu modellierende System. Ein Modell ist damit eine vereinfachte Beschreibung eines Systems, wobei hier die relevanten Problemaspekte dieses Systems berücksichtigt werden sollen. Nur dadurch kann ein Modell als konsistent betrachtet werden. In [Zang73] sind die Modelltypen folgendermaßen gegliedert:

- Analoge Modelle zur Darstellung der Eigenschaften eines Objektes durch andere ähnliche Eigenschaften.
- Bildhafte Modelle zur Beschreibung der erkennbaren Eigenschaften der Objekte.
- Formale Modelle zur Darstellung der Eigenschaften der Objekte durch logische oder mathematische Methoden.

Der Begriff **Datenmodell** beschreibt die statische Struktur eines Teils der realen Welt. Dabei werden Entitäten, ihre Attribute und die Beziehungen zwischen diesen Entitäten dargestellt. Der Begriff Datenmodell ist in dieser Hinsicht nicht mehr präzise. Eher kann hier der Begriff Entitätsmodell verwendet werden [Hess94].
Ein **Objektmodell** ist eine Darstellung der statischen Struktur von Objekten in einem System. Dabei werden die Identität, die Attribute, die Methoden und die Beziehungen zwischen den Objekten beschrieben [Rumb93]. Ein Objektmodell wird danach durch ein dynamisches Modell und ein funktionales Modell erweitert.
Ein **Produktmodell** ist in [Suhm93] als „ein Informationsmodellschema, das alle relevanten Informationen über ein Produkt abbilden kann“ beschrieben. Ein integriertes Produktmodell beinhaltet nach Polly [Poll96] „die Abbildung aller relevanten Produktmerkmale, die in den einzelnen Lebensphasen entstehen, auf der Basis einer einheitlichen, allgemeinen, lebensphasenübergreifenden und redundanzfreien Grundstruktur“. In Hinsicht auf die Überschaubarkeit der Darstellung wird nach Seiler [Seil85] das Produktmodell in Partialmodelle zerlegt. Ein Partialmodell definiert nach Pätzold [Pätz91] eine “abgeschlossene Menge von Objekttypen sowie deren semantische Zusammenhänge als Teilmenge eines Produktmodells, die eine anwendungsunabhängige Klasse von Produktmerkmalen beschreiben“. Um die Komplexität des integrierten Produktmodells zu reduzieren, reicht die Zerlegung des integrierten Produktmodells in Partialmodelle nach Pätzold [Pätz91] nicht aus. Ein integriertes Produktmodell kann daher durch Produktmodellsichten spezifiziert werden. Nach Pätzold definiert eine Produktmodellsicht „ein anwendungsorientiertes Informationsmodell unter Einbeziehung aller Produktmerkmale der relevanten Partialmodelle des integrierten Produktmodells.“

2.2 Geometrische Modellierung

Als geometrisches Modellieren innerhalb des CAD-Prozesses bezeichnen Spur und Krause [Spur91] "den gesamten mehrstufigen Vorgang, ausgehend von der aus einer Aufgabenstellung resultierenden gedanklichen Vorstellung, dem Entwurf, bis hin zur Abbildung des vollständig gestalteten Produkts in einer rechnerinternen Darstellung". Nach Nowacki [Nowa83] befasst sich die Disziplin des geometrischen Modellierens mit der "Erzeugung geometrischer Objektbeschreibungen zum Zwecke der rechnerinternen Darstellung geometrischer Objekte. Der Nutzer liefert dem Modelliersystem über seine Eingabegeräte Informationen zur Definition der Objektgeometrie, aus denen die rechnerinterne Darstellung des Modells aufgebaut und fortgeschrieben wird".

Die geometrische Modelldarstellung gliedert sich in (Abb. 2.1):

- Ebenendarstellung (2D),
- Erweiterte Ebenendarstellung (2.5 D),
- Räumliche Darstellung (3D).

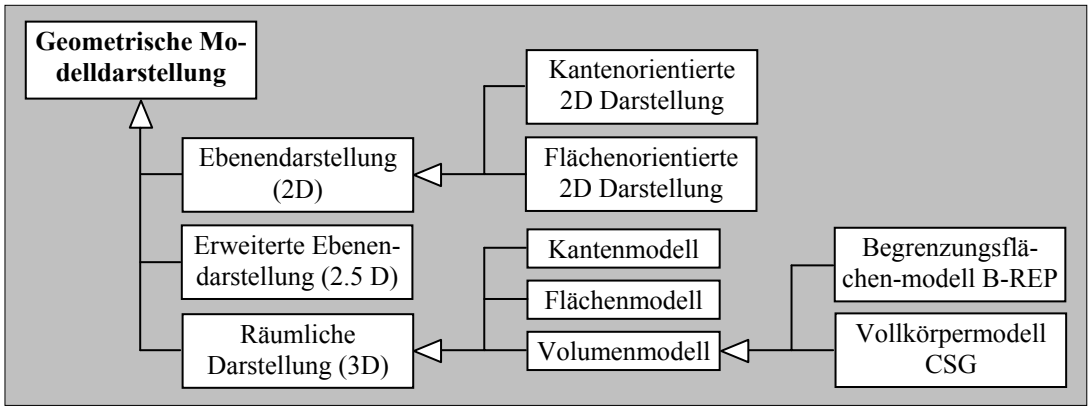


Abb. 2.1: Formen der geometrischen Modelldarstellung

Die 2D-Darstellung ist nur sinnvoll, wenn sich alle wichtigen geometrischen Daten eines Objektes in einer Ebene abbilden lassen. Bei der 2.5D-Darstellung werden die geometrischen Objekte in einer 2D-Ansicht beschrieben und gespeichert, durch mathematische Operationen, wie z. B. Rotation, wird jedoch eine räumliche Darstellung entwickelt. Eine vollständige und korrekte rechnerinterne Darstellung von geometrischen Objekten kann durch eine 3D-Darstellung erreicht werden. Für diese Darstellung sind drei Modelle geeignet:

- **Kantenmodell**
  Man spricht von einem Kantenmodell (auch Drahtmodell oder Linienmodell genannt), wenn ein geometrisches Objekt durch Punkte und Konturen abgebildet wird. Räumliche Darstellungen können jedoch auch mit einem Kantenmodell erzeugt werden, dann sind sie aber nicht eindeutig. Dieses Modell kann angewendet werden, falls die geometrischen Objekte durch ebene Flächen begrenzt sind [Bern93].
- **Flächenmodell**
  Bei dem Flächenmodell wird ein Objekt geometrisch durch seine Oberfläche dargestellt. Die Flächen werden mittels Konturen begrenzt, die durch zwei benachbarte, sich durchdringende oder beruhende Flächen entstehen. Spezifische Orte bei der Festlegung von Konturen sind die Punkte.

- **Volumenmodell**
  Geometrische Objekte werden bei einem Volumenmodell durch ihre Rauminhalte abgebildet [Tori93]. Basiskörper und mengentheoretische Verknüpfungen (Vereinigung, Durchschnitt und Differenz), die ihrerseits in einer Baumstruktur dargestellt werden können, sind die Grundlage dieses Modells. Das Volumenmodell gliedert sich in zwei Modelle:
  - **Begrenzungsflächenmodell**
    Volumenelemente werden in diesem Modell mittels ihrer Oberflächen definiert. Eine derartige Darstellung der Objekte wird auch B-REP *(boundary representation)* genannt. Das Begrenzungsflächenmodell ist eine effiziente Methode zur geometrischen Modellierung, erfordert aber einen höheren Beschreibungsaufwand [Roll95].
  - **Vollkörpermodell**
    Volumen werden in diesem Modell durch mengentheoretische Verknüpfungen von Basiskörpern modelliert. Dieses Modell wird auch als CSG *(constructive solid geometry)* bezeichnet [Schu95].

In [Berk96] ist die Anwendung der geometrischen Modellierung auf Trassierungen im dreidimensionalen Geländemodell erörtert. Darin wird die Vorgehensweise für die Trassenmodellierung unabhängig von den verfügbaren Trassierungselementen beschrieben.
In Bezug auf die Bedeutung des Geometriemodells eines Gebäudes ist in [Krus96] die Euler-Modellierung dreidimensionaler Körper beschrieben. Hier werden die topologischen Grundlagen eingesetzt und so erweitert, dass beliebige Körper in 3D entworfen werden.

2.3 Produktmodellierung

Produktmodelle können folgendermaßen gegliedert werden [Abel95]:

- geometrische Produktmodelle
- featureorientierte Produktmodelle
- strukturorientierte Produktmodelle
- wissensbasierte Produktmodelle
- integrierte Produktmodelle.

Integrierte Produktmodelle werden als Ergebnis der Integration von geometrischen, featureorientierten, strukturorientierten und wissensbasierten Produktmodellen betrachtet. Features sind geometrieorientierte Objekte; sie werden als Trägerobjekte semantischer Daten zur Modellierung von nicht geometrischen Daten betrachtet.

In den letzten Jahren sind unterschiedliche Forschungsprojekte zur Produktdatentechnologie wie z. B. CAD*I [Grab91], NEUTRABAS [Muel92], IMPPACT [Giel93], KCIM [Grab94] und SFB346 [Hain99] erarbeitet worden. Ziel dieser Forschungsprojekte war es, die Möglichkeit zu schaffen, die einmal gewonnenen Produktdaten in einer Prozesskette ständig weiterzuverarbeiten, um ihren Datenfluss zu steuern und die dabei neu entstehenden Daten zu verwalten.
In [Spur94] sind Modellierungsmethoden für rechnerintegrierte Produktionsprozesse dargestellt. Hier werden Modelle des Produktionsprozesses wie u. a. das STEP-Modell, *Reference Model for Shop Floor Production* (ISO Technical Report 10314-1), Modell des CAM-I, sowie zahlreiche CIM-Modelle beschrieben.

Ein Konzept zur Gestaltung von prozess- und integrationsgerechten Produktmodellen ist in [Warn95] beschrieben. Es wird hier zuerst die Produktentwicklung als Einsatzfeld der prozess- und integrationsgerechten Produktmodelle analysiert. Weiterhin wird die Struktur der Produktdaten beschrieben. Danach wird ein Referenzmodell der Produktdaten dargestellt. Ferner wird eine Referenzarchitektur zur Entwicklung von prozess- und integrationsgerechten Produktmodellen gezeigt. Aufbauend auf dem Referenzmodell und auf der Referenzarchitektur wurde ein Produktmodell im Bereich der Automobilindustrie entwickelt.

In [Rich91] sind wissensbasierte CAD-Systemkomponenten zum Entwurf montagegerechter Produkte dargestellt. Hier wird die Notwendigkeit betont, die geometrischen Daten zur Darstellung der Produktform zusammen mit den technischen Produktdaten rechnerintern zu verwalten. Hierzu sind die festgelegten technischen Relationen von Bedeutung, mit deren Speicherung rechnerintern die Produkttopologie dargestellt wird.

2.4 STEP als Norm zur Produktmodellierung

Im Anschluss an intensive langjährige Normierungsarbeiten unter Beteiligung zahlreicher internationaler Teilnehmer wurde STEP *(Standard for the Exchange of Produkt Model Data)* entwickelt und als Initial Release der Normenreihe ISO 10303 im 1994 freigegeben. Im Gegenteil zu den anderen rein geometrieorientierten Austauschformaten wie z. B. IGES und VDA-FS übertragen die STEP-Prozessoren auch nichtgeometrische Daten wie Maße, Oberflächenangaben, Stammdaten, Materialdaten sowie Arbeitspläne. In Abbildung 2.2 ist die Entwicklung der wichtigsten Standards zum Produktdatenaustausch dargestellt. STEP wird hier als Weiterentwicklung der anderen Standards betrachtet.

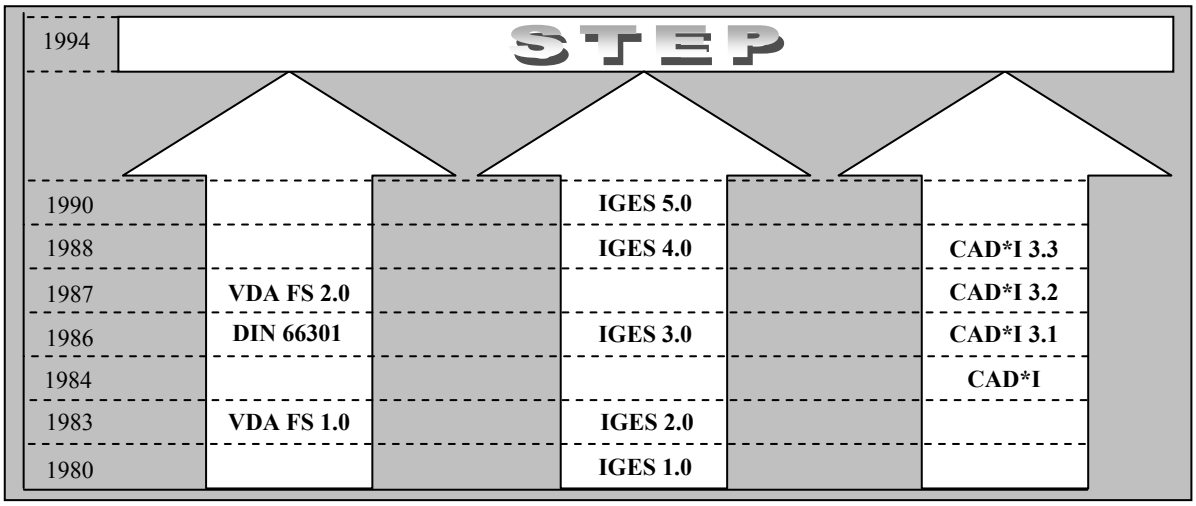


Abb. 2.2: Entwicklung der wichtigsten Standards zum Produktdatenaustausch

STEP bezieht sich als Norm zur Produktmodellierung auf Methoden zur Gewinnung, Speicherung, Verwaltung, Austausch, Archivierung und Dokumentation von Produktdaten.
STEP ist also eine Norm zur implementierungsunabhängigen Beschreibung von Produktdaten, die sich über den gesamten Produktlebenslauf ausweitet. Das Ziel von STEP ist es, ein Werkzeug anzubieten, das in der Lage ist, Produktdaten über den gesamten Lebenszyklus eines Produktes unabhängig von einem bestimmten EDV-System zu beschreiben.
Bei der Produktmodellierung nach STEP werden die zur Unterstützung der Prozessketten erforderlichen Teilmengen des integrierten Produktmodells in Form von Anwendungsprotokollen *(Application Protocols)* dargestellt. Durch diese Darstellung der Produktdaten wird die Anpassungsfähigkeit bei den Datenverarbeitungssystemen in vollem Umfang erhöht, weil die Datenmodellierung nicht mehr auf eine bestimmte Implementierungsform angewiesen ist, sondern neutral mit der Datenmodellierungssprache EXPRESS dargestellt wird. Da die Portabilität und Verteilung von Komponenten bei dem Einsatz der EDV-Systeme eine andauernd größere Rolle erlangt, ist die Entwicklung von Konzepten und Komponenten unter der Anwendung von STEP und CORBA (**C**ommon **O**bject **R**equest **B**roker **A**rchitecture) notwendig.

2.4.1 Gliederung der STEP-Norm

Die Normenreihe ISO 10303 ist in verschiedene Paletten aufgeteilt.
Die 0-Palette beinhaltet Dokumentation zur Erläuterung des Entwicklungsziels und die Gliederung von STEP. In dem ersten Dokument dieser Reihe, das unter ISO 10303-1 [ISO1:94] "Überblick und grundlegende Prinzipien" *(Overview and Fundamental Principles)* veröffentlicht wurde, wurden fundamentale Definitionen sowie der Aufbau von STEP dargestellt.

Die 10er Serie der ISO 10303 beinhaltet die "Beschreibungsmethoden" *(description methods)*. Diese Beschreibungsmethoden, deren Kern die Modellierungssprache EXPRESS [ISO11:94] (Teil 11 und Teil 12) ist, dienen der Spezifikation des Integrierten Produktmodells.
Die 20er Serie bildet die "Implementierungsmethoden" *(Implementation methods)*. Im Teil 21 "Klartext-Kodierung der Austauschstruktur" *(Clear text encoding of the exchange structure)* [ISO21:94] ist die Analyse von Methoden zum Austausch von mit EXPRESS beschriebenen Produktdaten zwischen den unterschiedlichen EDV-Systemen dargestellt. Im Teil 22 *(Standard Data Acces Interface, SDAI)* werden Einführungsmethoden analysiert, die den Zugriff auf eine mit EXPRESS beschriebene Datenbasis ermöglichen.

Ferner umfasst die ISO 10303 in der 30er Serie "Methodologie und Rahmenwerk für Konformitätstests" *(conformance testing methodology and framework)* Vorschriften zur Kontrolle der Konformität von Implementierungen. Die notwendigen Konformitätsanforderungen für ein Anwendungsprotokoll sind innerhalb dieses AP niedergeschrieben. Dabei muss man zwischen optionalen und solchen Konformitätsanforderungen, die auf jeden Fall erfüllt werden sollen, unterscheiden. Innerhalb dieser Serie sind Verfahren zum Konformitätsnachweis festgelegt, die zur Gültigkeitserklärung von STEP-basierter Software dienen.

Eine weitere Palette bildet die 40er Serie der ISO 10303, die als „Allgemeine integrierte Ressourcen" *(integrated generic resources)* bezeichnet werden. Es handelt sich hier um Kernmodelle, die Produktdaten anwendungsunabhängig in EXPRESS darstellen. Die „Allgemeinen integrierten Ressourcen" gewinnen ihre Bedeutung aus der Tatsache, dass sie als Kernmodelle die Grundlage der 100er und 200er Serie der ISO 10303 darstellen.

Die 100er Serie der ISO 10303 bildet die "Anwendungsbezogene integrierte Ressourcen" *(integrated application resources)*. Sie vervollständigen die allgemeinen integrierten Ressourcen (40er Serie) um Besonderheiten von festgelegten Anwendungsbereichen. Die 100er Serie bildet zusammen mit der 40er Serie das integrierte Produktmodell von STEP. Die Darstellung dieses Produktmodells beginnt jedoch in der 40er Serie durch die Beschreibung und Konfiguration des Produktes.

Die 200er Serie "Anwendungsprotokolle" *(application protocols, AP)* beinhaltet die anwendungsbezogenen Modelle von STEP. Da diese Anwendungsprotokolle (Abk. AP) den Inhalt und Ausmaß der zwischen verschiedenen Systemen austauschbaren Produktdaten festlegen, sind sie für den Anwender von besonderer Bedeutung. Ein Anwendungsprotokoll beinhaltet die folgenden drei Modelle:

- Das **Prozessmodell** (*Application Activity Model, AAM)*:
  Innerhalb dieses Modells wird die Prozesskette, die durch das Anwendungsprotokoll unterstützt wird, beschrieben. Es handelt sich dabei um einen Kontext der Aktivitäten der Entities zur Beschreibung des Produktdatenmodells. Zur Darstellung des Prozessmodell wird die Sprache IDEF0 [IDE092] angewendet.
- Das **Referenzmodell** *(Application Reference Model, ARM)*:

Hier werden die Produktdatenobjekte, die im AAM identifiziert wurden, durch geeigneten Beschreibungsmethoden (wie z. B. EXPRESS-G) in einem Modell dargestellt.

- Das **Interpretierte Modell** *(Application Interpreted Model, AIM)*:
  Dieses Modell beschreibt, auf welche Weise die Strukturen des integrierten Produktmodells anzuwenden sind, um die Forderungen, die im ARM festgelegt sind, umfassend zu realisieren. Das AIM wird in der Sprache EXPRESS geschrieben.

Mit dem Einsatz von Funktionsblöcken *(Units of Functionality, UoF)*, die als Zusammenstellung von Entities, Attributen und Zwangsbedingungen betrachtet werden, wird die Bedienung durch Modularisierung innerhalb der Anwendungsprotokolle verbessert.
Falls zwei Anwendungsprotokolle gleichartige Funktionsblöcken beinhalten, wie es in AP 201 und AP 202 der Fall ist, entstehen damit gleichartige Strukturen in den interpretierten Modellen dieser Anwendungsprotokolle. Dieses Überschneiden von zwei interpretierten Modellen wird interpretierter Ressourcenblock *(Application Interpreted Construct, AIC)* genannt.

2.4.2 Entwicklungsmethoden der STEP-Norm

2.4.2.1 Beschreibungsmethoden

▪ Die Datenmodellierungssprache EXPRESS

EXPRESS [ISO 10303-11] ist eine Datenmodellierungssprache, deren Grundlage das Entity-Relationship-Modell ist. Diese Spezifikationssprache ist der Ausgangspunkt sämtlicher Modellbeschreibungen in ISO 10303. Sie wurde von der "International Organization for Standardization", ISO, im Rahmen von STEP entwickelt. EXPRESS wurde 1994 als internationaler Standard ISO 10303-11 registriert. Bedeutender Vorteil der Anwendung dieser Sprache ist die deutliche Definition der Objekte und die Unterstützung moderner Objektorientierter Konzepte. So stellt sie Mechanismen zum Aufbau von Objekthierarchien zur Verfügung und gestattet die mehrfache Vererbung. EXPRESS ist keine Programmiersprache. Im Vergleich zu Programmiersprachen fehlen ihr nur die Funktionen zur Ein- und Ausgabe sowie zur Speicherverwaltung. Sie ist eine ausdrucksvolle und umfangreiche Spezifikationssprache. EXPRESS ist auch als Datenmodellierungssprache außerhalb des Gültigkeitsbereichs von STEP geeignet. Formal ist EXPRESS auf der Basis der Wirth Syntax Notation (WSN) [WIR77] beschrieben. Die wichtigsten Sprachelemente von EXPRESS sind in den folgenden Deklarationen enthalten (Abb. 2.3):

- **Schema**
  Innerhalb eines Schemas werden semantisch zusammengehörige Daten zusammengefasst. Die Anwendung von Definitionen aus anderen Anwendungsbereichen ist realisierbar, indem diese in der Schnittstellenspezifikation (interface_spec) angegeben werden.
- **Datentypen**
  Datentypen gliedern sich in EXPRESS in einfachen, aggregierten, konstruierten, benannten und generalisierten Datentypen.
- **Regeln**
  Durch Regeln *(rules)* und mit Hilfe von WHERE-Klausel können Integritätsbedingungen im EXPRESS realisiert werden. Diese Regeln gliedern sich in lokale Regeln, die innerhalb eines Entity definiert werden und für seine Attribute Gültigkeit haben, und globale Regeln, die innerhalb eines Schemas definiert werden und für seine Entities Gültigkeit haben.

- **Funktionen, Prozeduren und Algorithmen**
  In EXPRESS erfordern Integritätsbedingungen, die durch die Klausel WHERE und RULE sowie die abgeleiteten Attribute realisierbar sind, eine Beschreibung durch Algorithmen. Hierbei werden Funktionen und Prozeduren zur Verfügung gestellt. Eine Funktion ist ein Algorithmus, das an Parametern operiert und sich daraus ein Einzelwert ergibt. Eine Prozedur ist ein Algorithmus, der Parameter von der Anrufungsstelle empfängt und operiert, sowie den gewünschten Abschlusszustand produziert.

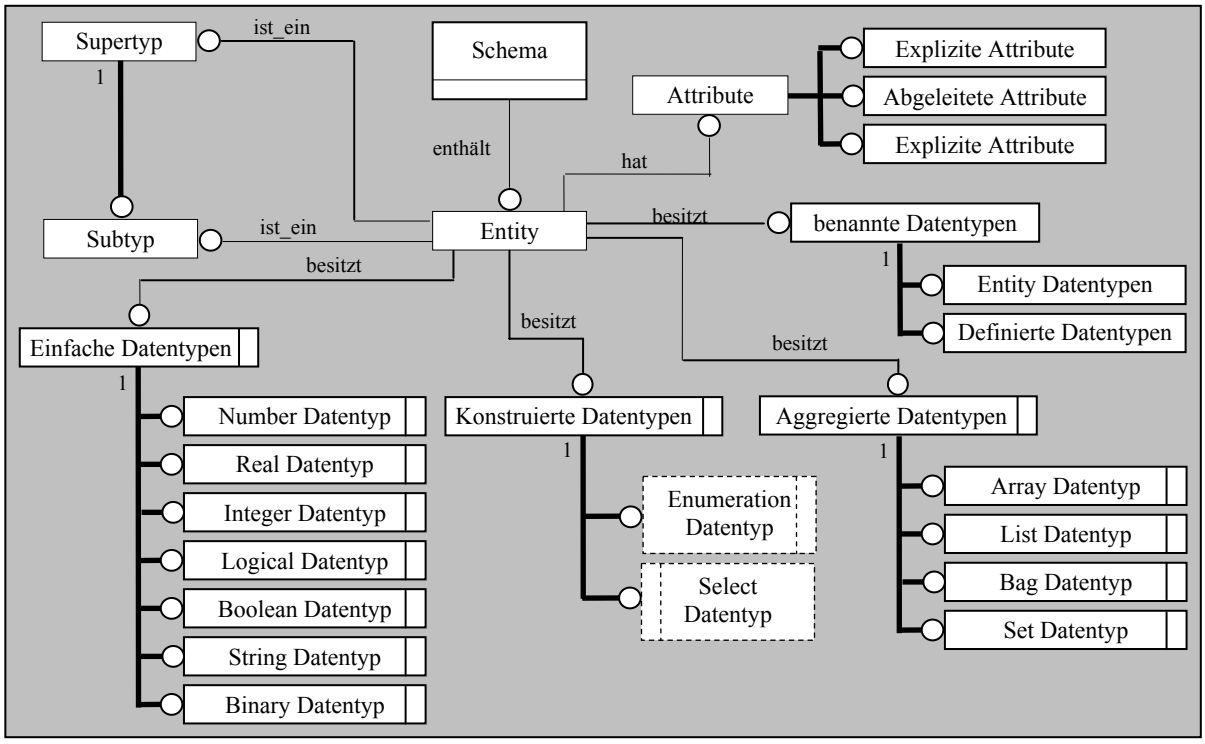


Abb. 2.3: Darstellung der Elemente von EXPRESS mit der graphischen Notation EXPRESS - G

Neben der textuellen Sprache wurde auch die graphische Notation EXPRESS-G (ISO 10303 Part 11, Annex D) entwickelt. EXPRESS-G [ISO 10303-11] ist eine formale graphische Notation zur Darstellung von Spezifikationen, die in der Datenmodellierungssprache EXPRESS definiert sind. In EXPRESS-G werden die Elemente der Sprache EXPRESS durch graphische Symbole repräsentiert. Diese Symbole, die bestimmte Diagramme bilden, gliedern sich in Definitionssymbole zur Darstellung von Datentypen und Schema-Deklarationen, Relationensymbole zur Darstellung von Beziehungen zwischen den Definitionen und Kompositionssymbole zur Darstellung von Diagrammen mit mehr als einer Seite.

2.4.2.2 Implementierungsmethoden

- **Physical File**

Im Teil 21 von STEP *(Physical File)* ist beschrieben, wie aus einem EXPRESS-Schema ein Format für eine Austauschdatei zu diesem Schemata entwickelt wird [ISO 10303-21]. Das Physical File beinhaltet Methoden zum Lesen und Schreiben von STEP Dateien. Dieser Teil der ISO 10303 gliedert sich in Austauschstruktur *(exchange structure)*, Sektion *(section)* und Zeichen *(token)*. Bei der Austauschstruktur handelt sich um eine Datei, die zum Speichern und Austauschen von STEP-Daten dient. Eine Sektion ist eine Sammlung von gleichartigen STEP-Daten. Zeichen sind solche Symbole, die innerhalb der Austauschstruktur verwendet werden. Diese Zeichen können als Schlüsselworte oder einfache Datentypen definiert werden.

- **Das Standard Data Access Interface (SDAI)**

Produktdaten können auch in einer STEP-Datenbank gespeichert werden, wobei die Anwendungen auf diese Daten durch eine genormte Schnittstelle zugreifen können. Als Schnittstelle für den Zugriff auf eine STEP-Datenbank ist in STEP das SDAI *(Standard Data Access Interface)* definiert [ISO 10303-22]. SDAI ist also ein Standard für den Zugriff auf die gespeicherten STEP-Daten. Dieser Standard gliedert sich in zwei Hauptabschnitte, in die Datenstrukturen sowie in die Zugriffsoperationen. Zur Verbesserung der Funktionalität werden für jede Operation eine Beschreibung, die Ein- und Ausgabe Parameter, die denkbaren Fehlermeldungen und die Auswirkungen auf die SDAI-Umgebung angekündigt.

- **Language Binding**

Da die SDAI Schnittstelle unabhängig von einer Programmiersprache entwickelt wurde, wurde innerhalb von STEP die sogenannte *Language Binding* entworfen. Sie beschreibt die Art und Weise der Umwandlung der in Teil 22 definierten Konstrukte in einer Programmiersprache. STEP beinhaltet *Language Binding* für C++ (Teil 23), C (Teil 24), Fortran (Teil 25), IDL (Teil 26) und Java (Teil 27). In Teil 28 wird die *XML representation of EXPRESS schemas and data* beschrieben.

2.5 Das Produktmodellierungskonzept von STEP im Bauwesen

Das Produktmodellkonzept von STEP im Bauwesen ist der Grundstein für die Standardisierung der Bauobjekte. Eine konsequente Definition der Bauobjekte leistet einen Beitrag zur Steigerung der Effizient und der Qualität im Bauprozess und wird als Fundament für die Automatisierung im Bauingenieurwesen betrachtet. Im Bereich des Bauwesen sind folgende STEP-Normen verabschiedet bzw. in der Entwicklungsphase:

- Teil 106 *Building Construction Core Model*
- Teil 202 *Associative Draughting*
- Teil 225 *Building Elements Using Explicit Shape Representation*
- Teil 227 *Plant Spatial Configuration*
- Teil 228 *Building Services: Heating, Ventilation and Air Conditioning*
- Teil 230 *Building Structural Frame: Steelwork*

Zur Entwicklung anwendungsabhängiger Sichten des STEP-Produktmodells im Bauwesen werden Implementierungsvorschriften in Form von Anwendungsprotokollen definiert. Bei deren Entwicklung werden normalerweise die entsprechenden „Allgemeinen integrierten Ressourcen“ (40er Serie) und die „Anwendungsbezogenen integrierten Ressourcen“ (100er Serie) verwendet.

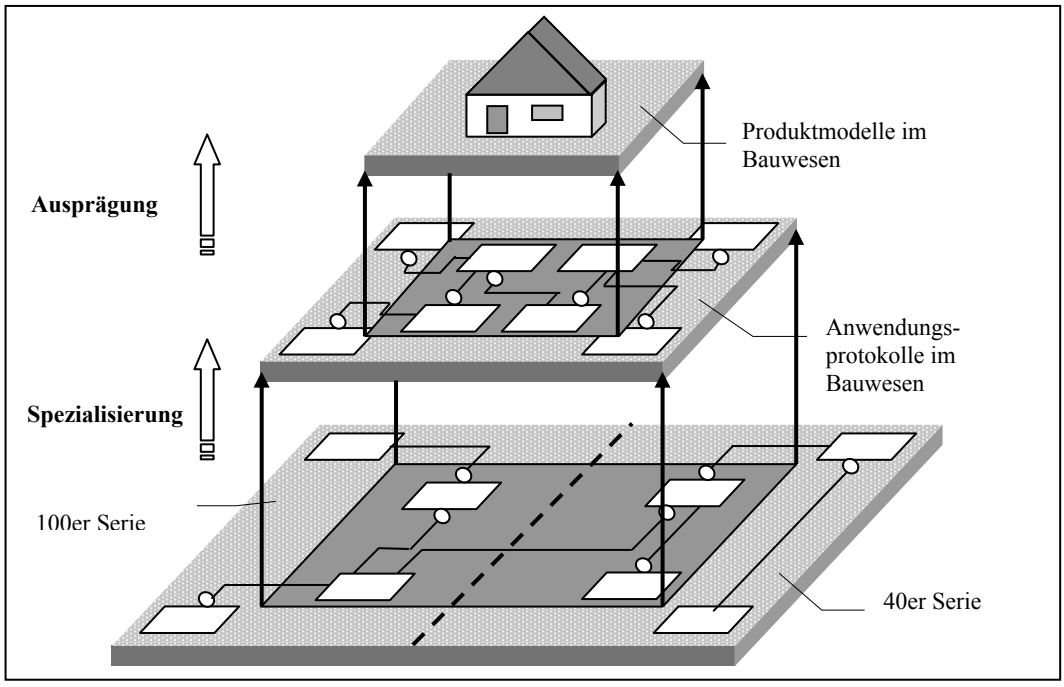


Abb. 2.4: Entwicklung von Produktmodellen im Bauwesen nach STEP

In Abbildung 2.4 wird die Entwicklung von Anwendungsprotokollen im Bauwesen dargestellt. In dieser Abbildung werden diese Anwendungsprotokolle als Spezialisierung der entsprechenden „Allgemeine integrierte Ressourcen“ (40er Serie) und der „Anwendungsbezogene integrierte Ressourcen“ (100er Serie) wie Teil 106 angegeben. Hier wird auch dargestellt, wie Produktmodelle im Bauwesen als Ausprägung aus den entsprechenden Anwendungsprotokollen abgeleitet werden können.
Nachfolgend werden AP 225, Teil 106 sowie STEP-CDS beschrieben.

2.5.1 AP 225: Building Elements using Explicit Shape Representation

Zum Zeitpunkt des Beginns der Entwicklung von AP 225 *(Teil 225: Building Elements Using Explicit Shape Representation)* existierten keine anwendungsbezogenen integrierten Ressourcen (100er Serie) für das Anwendungsgebiet Bauwesen. Erst einige Jahre später begann die Entwicklung von Teil 106 *(Building Construction Core Model)* als Teil der „Anwendungsbezogenen integrierten Ressourcen" für das Bauwesen. Die Entwicklung von AP 225 und Teil 106 wurde zuerst unabhängig voneinander vorangetrieben. Danach wurde entschieden, Maßnahmen zur Harmonisierung zwischen den unterschiedlichen Produktmodellen im Bauwesen, die sich in der Entwicklungsphase befinden, zu ergreifen.

AP 225, der im Jahr 1999 als ISO-Norm (ISO 10303-225) verabschiedet wurde, ist ein Anwendungsprotokoll zur Entwicklung von räumlichen Gebäudemodellen. Dieses AP ermöglicht den Datenaustausch von 3D-Gebäudemodellen. Dabei wird Wert auf eine korrekte Übertragung der 3D-Geometrie und der Struktur von Gebäuden gelegt.
Mit dem Einsatz von 3D-CAD Systemen können 3D-Zeichnungen erzeugt werden, welche die Kommunikation zwischen dem Planer, Auftraggeber und dem Lieferanten deutlich verbessern. Ferner können aus 3D-Gebäudemodellen durch die Angabe von Schnittebenen Zeichnungen zum Teil automatisch abgeleitet werden. Dadurch können die Schnitt- und Konturlinien der Teile eines Gebäudes automatisch erzeugt werden [Haas95].

Innerhalb von AP 225 werden folgende Anwendungsgebiete einbezogen:

- Terrainmodelle zur Beschreibung der Baustellen und der Baugrube in den verschiedenen Bauprozessphasen.
- Baukonstruktionen wie z. B. Dächer, Decken, Stützen, Treppen, Träger und Wände.
- Technische Gebäudeausrüstung wie Heizung, Lüftung und klimatechnische Einbauten, Wasserver- und Entsorgung und Elektroinstallation.
- Räume wie Zimmer, Säle und Korridore.

Das Referenzmodell des AP 225 setzt sich aus den Funktionsblöcken *(units of functionality, UoF)* zusammen, welche die Objekte diese AP beschreiben. Bei der Entwicklung dieses Anwendungsprotokoll werden die zur Darstellung eines Produktes notwendigen ISO-Normen wie ISO 8824-1 *(Specification of basic notation)*, ISO 10303-1, ISO 10303-11, ISO 10303-21 und ISO 10303-31 verwendet. Von den „Allgemeinen integrierten Ressourcen", die bei der Entwicklung eines Anwendungsprotokoll eine zentrale Rolle spielen, werden hier die ISO-Normen ISO 10303-41, ISO 10303-42, ISO 10303-43, ISO 10303-44 und ISO 10303-45 eingesetzt. Mit der Produktdatenmodellierung auf der Grundlage von AP 225 kann ein qualitativ hochwertiger Datenaustausch zwischen den am Bauprozess beteiligten Seiten realisiert werden. Dies führt auch zur bedeutenden Zeit- und Kosteneinsparungen in den unterschiedlichen Phasen des Bauprozesses.

2.5.2 STEP-CDS

Da z. Z. die meisten im Bauwesen angewendeten CAD-Systeme 2-dimensional sind, wurde 1997 auf Initiative des Verbandes der Automobilindustrie und der Pro STEP e. V. die Schnittstelle STEP-CDS *(Construction Drawing Subset)* definiert, die solche Systeme unterstützt. Mit dem Einsatz von STEP-CDS wird eine qualitativ hochwertige Schnittstelle zum CAD-Datenaustausch bereitgestellt [Haas97] [Hilb97]. STEP-CDS ist eine Untermenge der Daten der Conformitäts Klasse 4 des AP 214 *(core data for automotive mechanical design)* bzw. Conformitäts Klasse 2 des AP 212 *(electrotechnical design and installation)*, die für Industriebau relevant sind.

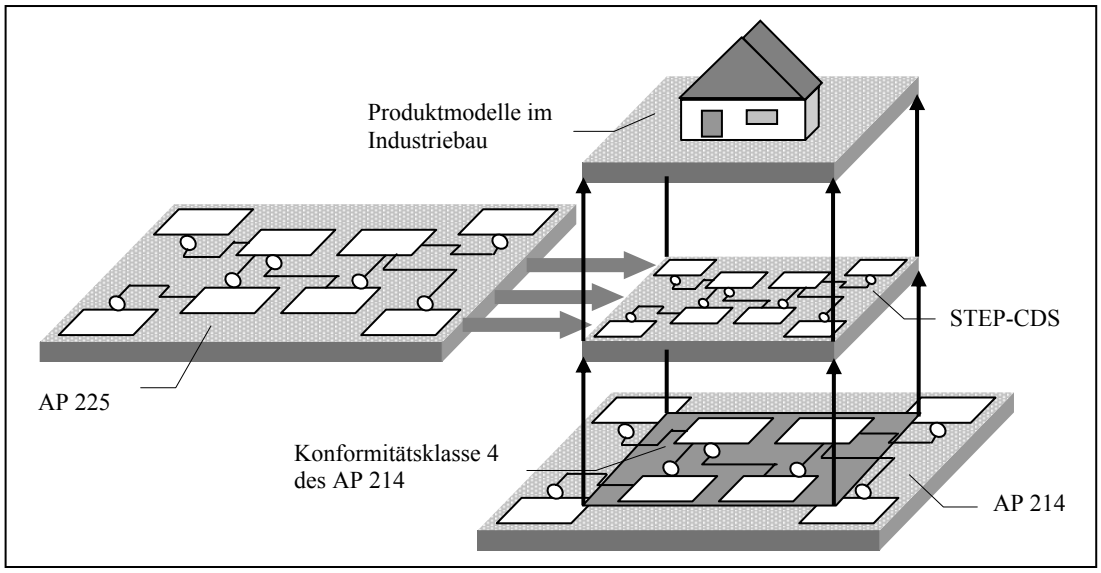


Abb. 2.5: Anwendung von STEP-CDS im Industriebau in Verbindung mit AP 225

In Abbildung 2.5 wird STEP-CDS als Untermenge der Daten der Conformitäts Klasse 4 des AP 214 dargestellt. In dieser Abbildung wird auch der Einsatz von STEP-CDS im Industriebau in Verbindung mit AP 225 dargestellt.

STEP-CDS beinhaltet folgende Informationsklassen:

- 2-D Geometrie als Modelldaten.
- Modellstrukturen wie Gruppen und Layer.
- Strukturdaten zur Erfassung von Anlage- und Gebäudestrukturen.
- Administrative Daten wie Identifikator, Version und Freigabe.
- Sachdaten wie Material, Oberfläche und Leistungswerte.
- Referenzen wie externe Daten und Dokumente.
- Technische Zeichnungen mit Bemessung, Maßtoleranzen, Beschriftung, Schraffur, Farben, Symbole und Zeichnungslayout.

2.5.3 Teil 106: Building Construction Core Model

Teil 106 *(Building Construction Core Model)* [ISO106] bildet innerhalb der "Anwendungsbezogenen integrierten Ressourcen" *(integrated application resources)* von STEP das Kernmodell für den Hochbau. Teil 106 spezifiziert die notwendigen integrierten Ressourcen zur Entwicklung von Anwendungsprotokollen im Hochbau. Teil 106 beschreibt die Datenstrukturen in allen Phasen des Produktlebenslaufs im Hochbau. Innerhalb von Teil 106 sind folgende Begriffe von großer Bedeutung:

- **Produkte** sind Gegenstände, die als Bestandteile eines Bauwerks vorkommen werden.
- **Prozesse** sind die erforderlichen Aktivitäten zur Herstellung der Produkte.
- **Ressourcen** sind für den Bauprozess notwendige Mittel zur Herstellung der Produkte.
- **Kontrollen** sind Bedingungen, die innerhalb des Bauprozesses an Produkten, Prozessen und Ressourcen angewendet werden.

Bei der Entwicklung von Teil 106 wird auf Teil 41 und Teil 42 als „Allgemeine Integrierte Ressourcen" der STEP zurückgegriffen und diese um die erforderlichen Modellkonstrukte des Hochbaues ergänzt.
Die Entwicklung von Teil 106 wurde mittlerweile eingestellt.

2.6 Industry Foundation Classes

Nach der Einstellung der Entwicklung von STEP-Teil 106 *(Building Construction Core Model)* wurde die **Industry Foundation Classes (IFC)** entwickelt.
Die IFC wird als weltweit gültige, plattformübergreifende Objektsprache für rechnergestützte Bauplanung, Bauausführung und Gebäudeverwaltung betrachtet. Die Definition der IFC berücksichtigt alle Lebensphasen eines Gebäudes vom Entwurf bis hin zum Abriss. Diese Objektsprache wurde von der **Industrie Allianz für Interoperabilität (IAI)** entwickelt. Die **IAI** ist ein internationaler Zusammenschluss von Firmen aus dem gesamten Umfeld des Baubereichs und verfolgt das Ziel, die Teilaspekte der AEC/FM-Industrie (Architecture, Engineering and Construction / Facility Management) zu integrieren und in einem Datenmodell zusammenzuführen. Gegenüber dem konventionellen Datenaustausch stellt die IFC dem Anwender ein programmübergreifendes Datenmodell bereit. Dieses Datenmodell baut auf bestehenden Entwicklungen wie STEP-Standard auf.

Insbesondere markiert die IFC in der Version IFC2x den Wechsel der IAI zu einer Plattformentwicklung sowie der Unterstützung der Haustechnik, des Facility Managements und des Projekt-Management. Diese Version enthält auch XML-Definitionen für wichtige Datenaustauschszenarien. So wird die ifcXML als Übertragung der bereits anerkannten Inhalte des IFC2x Standards in die XML Schema Definition, XSD betrachtet. Die ifcXML erlaubt die Übertragung von IFC Inhalten zwischen Softwarelösungen des Bauwesens. Damit werden die international gültigen IFC Beschreibungen der Bauelemente und Produkte auch im neuen Webstandard XML verfügbar. Bekanntlich erlaubt die XML (eXtensible Markup Language) den Austausch von strukturierten Daten und Inhalten über das World Wide Web.

2.7 Objektorientierte- und Produktmodellierung im Bauwesen

Nachdem sich die Objektorientierung als Konzept durchgesetzt hat, sind auf dem Gebiet der Objektorientierten Modellierung im Bauwesen zahlreiche Veröffentlichungen erschienen.
In [Rüpp94] ist das Objektorientierte Management von Produktmodellen der Tragwerksplanung beschrieben.
Rüppel und Meißner [Rüpp96] beschreiben den Einsatz der Produktmodellierung im Bauwesen zur Erzeugung digitaler Bauwerksmodelle. Das Bauwerksmodell besteht demnach aus Objekten, die in semantischen Beziehungen zueinander stehen.

In [Diaz97] wird die Objektorientierte Modellierung von geotechnischen Systemen dargestellt. Zuerst werden die Objektorientierte Analyse und der Objektorientierte Entwurf des geotechnischen Systems nach Rumbaugh [Rumb93] beschrieben. Danach werden u. a. die Semantik und der Einsatz eines geotechnischen Informationssystems erläutert. Anschließend wird die persistente Datenhaltung anhand des Objektorientierten Datenbank-Management-Systems ObjectStore beschrieben.

In [Reym95] wird die Objektorientierte Modellierung von Stahlbetonbauteilen dargestellt. Diese Bauteile werden als Objekte in eine Klassenhierarchie eingeordnet und auf Objektorientierter Basis analysiert. Die Objektorientierte Modellierung wird hier sowohl aus Anwendersicht als auch aus der Sicht des Programmierers dargestellt.

In [Lena90] ist ein Expertensystem zur Unterstützung des Entwurfsprozesses von Baukonstruktionen, dargestellt am Beispiel des Bereiches Treppendesign, beschrieben. Dieses System besteht aus einem Objektmodell, einer Datenbank und einer Benutzerschnittstelle. Das Objektmodell, mit seiner Struktur und dem Wissen über den Entwurf von Baukonstruktionen beschreibt ausführlich die Treppe in ihrer Entwurfsumgebung. Als Komponenten dieses Modells werden die Objekte und die Relationen zwischen diesen Objekten betrachtet. Es werden Klassenstrukturen gebildet, wobei die Klassen Geometrie, Konstruktion, Statik, Material, Gebäude- und Treppenfunktion als Oberklassen angesehen werden.

Gehri [Gehr92] stellt ein Beratersystem für die Baustellenleitung vor, mit dem die erforderlichen Informationen zu erfassen sind. Die Konzeption des Systems umfasst ein Kontroll- und Steuerungssystem, ein Dispositions- und Administrationssystem. Teile des Kontrollsystems wurden mittels der Konzepte der Objektorientierung definiert und anhand der Programmiersprache SMALLTALK 5 entworfen. Das Programmsystem ist mit einer Objekthierarchie ausgestattet, in der das Objekt Job als zentrales Element der Applikation dargestellt wurde. Job ist hier jedoch als ein räumlich oder terminlich abgrenzbares Resümee von Vorgängen und Leistungen definiert.

In [Webe93] ist der Einsatz der Objektorientierten Softwaretechnologie bei der automatischen Nachweisführung im Stahlbau nach DIN 18 800 gezeigt. Das Objektorientierte Schema der DIN 18 800 besteht aus zwei Klassen mit einer gemeinsamen Basisklasse. Die Basisklasse (class DinObj) beinhaltet die gemeinsamen Methoden der abgeleiteten Klassen Einwirkungen und Kombination. Die Deklaration dieser Klassen erfolgt mittels der Objektorientierten Programmiersprache C++.

Auf der Basis der Objektorientierten Konzepte wird in [Casp90] ein Modell eines regelbasierten Nachweissystems für den Stahlbau nach DIN 18800 T2 geschildert.
Ein Produktmodell für den Stahlbau ist in [Hall94] auf der Grundlage der STEP-Technologie dargestellt.

Osterrieder, Haller und Saal [Oste97] erläutern die bauspezifische, fertigungsorientierte Produktmodellierung im Stahlbau und Holzbau. Zur Vermeidung der Datenredundanz bei dem Einsatz von Softwarelösungen im Bauwesen wird hier die Bedeutung der Erstellung von Datenaustauschkonzepten auf ganzheitlicher, projektbezogener Basis betont. Dabei wird besonderes Gewicht auf die Erstellung von redundanzfreien Datenstrukturen, sowohl für die Planungs- als auch für die Fertigungsaspekte gelegt.
In [Höre98] sind Tendenzen und Stand der Technik bei der Produktmodellierung im Stahlbau und Holzbau erläutert.

In [Oste00] ist das Produktmodell DtH für den Datenaustausch bei der Planung und Fertigung im Holzbau dargestellt. Dieses Produktmodell beschreibt die relevanten Datenstrukturen für alle Lebensphasen des Produktes. Eine effiziente und redundanzfreie Strukturierung der Planungs- und Fertigungsprozesse steht hier im Vordergrund.
Um die angestrebte Akzeptanz bei den Softwareentwicklern zu erreichen, wird bei der Beschreibung dieses Produktmodells besonderer Wert auf eine implementierungsfreundliche, bau- und werkstoffspezifische Darstellung in parametrischer Form gelegt.

In [Ohts93] ist ein Konzept zur Finiten Element Modellierung, welches auf Objektorientierter Technologie basiert, beschrieben. Hier wird auch das Prototypprogramm MODIFY *(MODeling tool for Integrated Finite element analYsis)* dargestellt. Die Besonderheit von MODIFY ist u. a. die vollständige Objektorientierte Struktur. Zur Anwendung von MODIFY ist es notwendig, Teilobjekte zu definieren, welche aus Geometrie-Objekten, Objekten in analytischem Zustand und Beziehungsobjekten bestehen. MODIFY generiert automatisch ein passendes FE-Modell für jedes Geometrie-Objekt, welches die Stetigkeitsbedingungen mit angrenzenden Teilen durch Verweis auf Beziehungsobjekte erfüllt.

In [Gabb96] ist die Objektorientierte Integration der Finite-Element-Methode im Produktdatenmodell beschrieben. Hier werden Verwaltungsmodelle für Produktdaten, Integrationsmodelle für Teilmengen von Produktdaten und Standardmodelle für Produktdatenaustausch als Formen der Produktmodelle klassifiziert. Der Produktdatenaustausch zwischen den Anwendungssystemen wird jedoch auf der Basis eines generischen Produktdatenmodells funktionieren. Der Entwurf dieses Integrationsmodells baut auf die Modellierung der Assoziation der Daten, die in den unterschiedlichen Phasen des Produktlebenslaufs erfasst werden, auf. Die Basis des Entwurfs dieser Integration ist ein entwickeltes generisches Produktdatenmodell. Ein solches Integrationsmodell ist nur dann wirkungsvoll, wenn es mit EXPRESS und in Form eines STEP-Anwendungsmodells beschrieben wird.

Eine Methode zur Speicherung von CAD-Gebäudedaten in einer Objektorientierten Datenbank ist in [Busch96] beschrieben. Zuerst werden hier die Objekte zusammen mit ihren Geometriedaten im CAD-Modell identifiziert und danach im CAD-System intern gespeichert. Dann werden diese Objektdaten in einem DXF-File dargestellt und schließlich werden sie aus dem DXF-File gelesen und in ein OODBS eingetragen.

In [Hütt96] wird der Einsatz von Basiskomponenten wie Graph, Zelle und Zellkomplex zur Modellierung von Anwendungen im Bauwesen wie z. B. Gebäudemodelle und Lichtsignalanlagen beschrieben. Hier wird gezeigt, wie die Objektorientierte Modellierung im Bauwesen insbesondere auf die Graphentheorie zurückgreifen kann.

Durch Rückgriff auf die Mengenlehre sind in [Laab98] Methoden zur Modellierung mit Objekten im Bauingenieurwesen dargestellt. Hier werden Methoden zur Modellierung von Ansichten im Bauwesen, die nicht mit der Objektorientierung befriedigend modelliert werden können, entwickelt. Zuerst werden hier die Struktur der Objekte, die Struktur der Mengen sowie die Dynamisierung und Konsistenz eines Systems analysiert. Danach werden Methoden entwickelt und implementiert und an einem Modell zur Visualisierung des physikalischen Verhaltens dreidimensionaler Körper dargestellt.

In [Nies93] sind Ansatzpunkte zur Entwicklung einer Schnittstelle zum Datenaustausch zwischen Tragwerksplaner und Objektplaner beschrieben. Dabei wird die Objektorientierte Technologie als geeignete Methode dargestellt.

Die Bearbeitung fachübergreifender Produktdaten wird in [Schn96] dargestellt. Zuerst werden hier die Struktur des Produktmodells und die Modellierung der Beziehungen der Objekte untereinander beschrieben. Danach wird die persistente Speicherung des Produktdatenmodells sichtbar gemacht.

In [Beet97] sind Basistechnologien zur Entwicklung von Objektorientierten Anwendungen im Bauwesen dargestellt. Den Kern dieser Technologien bildet ein "Dynamic Product Model-Kernel, kurz DPM". Es handelt sich dabei um eine Produktmodellierungstechnologie zur Modellierung eines Ausschnitts der Realität. Der DPM-Kernel erlaubt die Erweiterung und Modifizierung von Modellinformationen während der Laufzeit der Anwendung. Auf der Basis von DPM wurde eine Prototyp-Anwendung entwickelt, in der alle relevanten Daten eines Bauprojektes über seinen gesamten Lebenslauf verwaltet werden.

Die Integration vom STEP-Anwendungsprotokoll 225 in ein rechnergestütztes Facility Managementsystem ist in [Kark97] beschrieben. Hier wird das Datenmodell vom STEP-Anwendungsmodell 225 an ein vorhandenes Facility-Managementsystem angepasst.
Mit EXPRESS wird noch ein Modell zum Austausch der Gebäudedaten zur Verfügung gestellt. Dieses Modell dient zum Lesen und Interpretieren der Daten während des Datenaustausches. Die im AP 225 definierten Strukturen der Objekte werden hier berücksichtigt.

In [Schm96] sind eine Modellierungsmethode und eine Modellierungsplattform als Bestandteile eines Vorgangsmodellierungssystems vorgestellt. Es handelt sich dabei vorrangig um ein System zur Modellierung der Aktivitäten der beteiligten Objekte im Bauwesen. Zuerst werden hier die Prozesse nach der SADT/IDEF0 analysiert. Danach werden die zu modellierenden Bereiche nach [Rumb93] Objektorientiert modelliert. Schließlich werden die Ergebnisse der Modellierung zu einem integrierten Workflowmodell abgeglichen und verknüpft.

Flatscher [Flat93] präsentiert Grundgedanken zum Kalkulieren und Ausschreiben anhand der Prinzipien der Objektorientierung. Durch die Betrachtung der Materialien eines Bauteils als Daten und der Berechnungsfaktoren als Methoden dieses Bauteils, stellt er fest, dass die Bauteile teilweise schon den Grundsätzen der Objektorientierung genügen.

2.8 Rechnergestützte Modellierung auf dem Gebiet der baubetrieblichen Prozesse

In [Flat93] wird der Einsatz der Objektorientierung in der Baukalkulation und Ausschreibung im Bauwesen vorgestellt. Dabei werden die Baustoffe als Attribute und die Bauarbeiten als Operationen dargestellt. In [Flat96] ist der Einsatz der Daten aus CAD-Zeichnungen für die Erstellung von Leistungsverzeichnissen beschrieben. Es wird hier auch die Ermittlung der Maße aus den Zeichnungen dargestellt.

Rüppel [Rüpp98] beschreibt die Methodik eines ganzheitlichen Managements von Bauprojekten aus Auftraggebersicht auf der Basis integrierter Ablauf- und Kostensteuerung. Hier werden zuerst die Grundlagen des Projektmanagements im Bauwesen dargestellt. Dann wird die Methodik zum Management von Bauprojekten beschrieben, wobei die Umsetzung dieser Methodik anhand eines geeigneten Softwaresystems anschließend gezeigt wird.

Beucke und Huhnt [Huhn00] stellen ein Konzept zur Integration der vorhandenen und in der Praxis erprobten Verfahren zur Bearbeitung technischer und betriebswirtschaftlicher Aufgaben im Bauwesen vor. Dadurch kann eine Bearbeitung von Bauprojekten ohne mehrfache Brüche in den Informationsstrukturen erfolgen.

Diaz, Kalantari und Schäffler [Kala00] beschreiben ein Konzept zum Datenaustausch im Bereich AVA durch den Einsatz von XML.

Wassermann und Heck [Wass00] legen einen Entwurf zur Integration von raum- und bauteilorientierten Daten in der Gebäudeplanung in einem zentralen Objektmodell vor. Bei der Bearbeitung dieses Entwurfs liegt der Schwerpunkt in der strukturierten Umsetzung des Problemraums in ein Klassenkonzept. Dieser Entwurf wurde um zwei weitere Objektorientierte Teilmodelle zur Kostenermittlung nach DIN 276 sowie zur Flächen- und Rauminhaltermittlung nach DIN 277 erweitert. Diese Teilmodelle werden dann in das zentrale Objektmodell integriert. Zur Erstellung der Teilmodelle wurden diese DIN-Vorschriften untersucht und danach in entsprechenden Objektorientierten Teilmodellen abgebildet.
Bei der Ermittlung der Flächen und Rauminhalte eines Gebäudes werden diese gemäß DIN 277 für jedes Geschoß getrennt ermittelt. So werden die Hierarchie und die Gliederung eines Gebäudes in diesem Teilmodell durch entsprechende Klassenhierarchien modelliert.
Bei der Ermittlung der Kosten nach DIN 276 werden einige Aspekte behandelt, wie die Art und Weise der Betrachtung, Gliederung und Ermittlung der Kosten. Als weiterer Aspekt wird die Integration des Teilmodells in dem vorhandenen Modell untersucht.

Gehbauer und Lennerts [Gehe00] beschreiben ein Konzept zur Modellierung und Entwicklung eines wissensbasierten, Objektorientierten Systems zur Planung optimierter Baustellen-Layouts. Der Schwerpunkt liegt hier in der Entwicklung eines Objektorientierten Systems zur Planung von dynamischen Baustelleneinrichtungselementen im Hochbau. Die Entwicklung dieses Modells zeigt, dass das Objektorientierte Paradigma auch bei der Modellierung komplexer ingenieurwissenschaftlicher Anwendungen im Bereich Baubetrieb geeignet ist.
Zuerst wird das gesamte System der Baustelle unter dem Gesichtpunkt des Materialflusses Objektorientiert modelliert. Bei dieser Modellierung wird zunächst das Objektstrukturmodell der Baustelle entwickelt. Der Schwerpunkt der dynamischen Modellierung liegt in der Beschreibung der Systemzustände und ihre Übergänge sowie in den nebenläufigen Prozessen innerhalb des Systems.

Ein wissensbasierter Ansatz zur Kostenanalyse von Baustelleneinrichtungen ist in [Sche02] dargestellt. Es handelt sich hier um einen wissensbasierten Ansatz zur Kostenanalyse, dessen Basis ein dynamisch veränderliches Baustellenmodell ist. Als Ausgangspunkt bei der Ermittlung und Kontrolle der Projektkosten werden die DIN 276 und DIN 277 betrachtet. Zuerst werden das Baustellenmodell und seine Elemente beschrieben. Bei dieser Beschreibung wird sowohl auf STEP als auch auf IFC zurückgegriffen. Dann wird auf der Basis der IFC ein Kostenmodell der Baustelleneinrichtung dargestellt.

Scherer, Nollau, Buchwalter und Scheler [Sche00] beschreiben Produktinformationssysteme, die durch dynamische Klassifikation und ähnlichkeitsbasierte Suche unterstützt werden.
Diese Produktinformationssysteme dienen der Bereitstellung von betriebswirtschaftlichem Wissen sowie von Produktdaten innerhalb des Bauprozesses. Der Informationsbedarf besteht hier aber nicht nur aus Produktinformationen. Vielmehr werden hier Informationen und Wissen in Form von Lösungen und Varianten im Bereich des in Frage kommenden Produktes bereitgestellt. Um dieses Ziel zu erreichen, werden einige Schwerpunkte festgelegt. In Mittelpunkt steht hier eine dynamische, Objektorientierte Strukturierung der Produktdaten. Ein weiterer Schwerpunkt liegt in der Integration des Produktmodells der Teilprodukte in ein allgemeines Bauwerksmodell.

Klauer [Klau02] stellt ein Konzept zur Integration von Sach- und Kosteninformationen in das Modellbasierte Projektkommunikationssystem BauKom-Online. Das Ziel dieses Ansatzes ist es, die Erhöhung der Kostensicherheit bei der Planung und Ausführung von Bauprojekten zu erreichen. Durch eine über alle Leistungsphasen durchgängige Strukturierung der Kosten im Hinblick auf Aktualität und Nachvollziehbarkeit werden die unterschiedlichen Sichtweisen auf Kostenzusammenhänge und fachspezifische Kostenbeeinflussungen berücksichtigt.
Es werden semantische Beziehungen zwischen den bisher meist getrennt verwalteten Bereichen Gebäudestruktur, Kosten und Termine gebildet. Dies wird durch die Verknüpfung dokumentbasierter Ansätze bei der Gebäudeplanung mit Fachmodellen auf der Internetplattform BauKom-Online erzielt. Zu diesem Zweck wird ein internetbasierter Framework bereitgestellt, dessen Basis ein gewerkübergreifendes Sachinformations- und Kostenmodell (SIKOM) bildet. Dadurch können die Folgen von Planungsänderungen auf andere Fachinformationen gewerkübergreifend abgeschätzt werden. Dementsprechend können die Konsistenz der Planungsinformationen sowie die Kostensicherheit innerhalb des Systems verbessert werden.

Aus der Analyse der Ansätze auf dem Gebiet der rechnergestützten Modellierung der baubetrieblichen Prozesse können einige Schwerpunkte abgeleitet werden, die der wissenschaftlichen Weiterentwicklung auf diesem Gebiet dienen. Dazu zählt eine umfassende, Objektorientierte Modellierung der baubetrieblichen Prozesse als Basis einer effizienten Softwareentwicklung in diesem Bereich. Insbesondere die Modellierung der dynamischen Aspekte dieser Prozesse sollen näher untersucht werden. In diesem Zusammenhang wird auf die Bedeutung einer ausführlichen Untersuchung dieser Aspekte im System der Baustelle hingewiesen. Auch eine effiziente Integration des Baustellenmodells in das gesamte Modell des Systems des Bauprojektes soll im Hinblick auf den Einsatz neuer Technologien näher untersucht werden.
Objektorientierte Werkzeuge wie die UML sollen auf ihre Tauglichkeit zur Modellierung der Dynamik der baubetrieblichen Prozesse näher untersucht werden.
Eine neue, Objektorientierte Gliederung der DIN 276 und DIN 277 sowie die Anpassung dieser Gliederung an vorhandenen Plattformen, wie Industry Foundation Classes (IFC) kann auch als Schwerpunkt zur wissenschaftlichen Weiterentwicklung betrachtet werden.

3 Grundlagen der Objektorientierten Softwareentwicklung

3.1 Datenorientierte Modellierungskonzepte

▪ Das Entity-Relationship-Model

Das *Entity-Relationship-Model* (ERM) ist ein Werkzeug zur Modellierung von Ausschnitten der Realität. Ziel dieser Modellierung ist, sachlogische Strukturen zu entwerfen, die dann in einem relationalen Datenbankmodell umgesetzt werden. Komponenten des ERM sind Entitäten *(entities)*, die Beziehungen *(relationships)* zwischen den Entitäten und deren Eigenschaften. Durch den Einsatz von ERM kann ein Teil der Realität in Form eines konzeptionellen Datenschemas dargestellt werden. Das konzeptionelle Datenschema beinhaltet die Datenobjekttypen, die Schlüsselreferenzen zwischen diesen Datenobjekttypen, deren Attribute und die Integritätsbedingungen. Die konzeptionelle Datenmodellierung mit ERM umfasst einerseits das Spezifizieren der Entitäts- und Beziehungstypen, deren Attribute, sowie der Kardinalitäten der Beziehungen und anderseits die Darstellung der Ergebnisse in Form eines Entity-Relationship Diagramms.

Nach Chen [Chen92] sind die Konzepte der Objektorientierung längst im ERM ausgeführt. Ferner bringt er zum Ausdruck, dass die Objektorientierung aus Struktursicht eine Untermenge des ERM darstellt. Nach [Balz01] unterscheiden sich die Konzepte der Objektorientierung von den ER-Konzepte dadurch, dass diese kein Verhalten modellieren können. In [Voss98] wird das ERM als ein wichtiges Instrument zur Datenmodellierung bewertet. Die Gründe dafür sind u. a. die Unabhängigkeit dieses Modells von einem bestimmten Datenbanksystem sowie sein starkes und natürliches Ausdrucksvermögen.
Das ERM ist in [Chen76], [Your89], [Flat90], [Chen91] und [Bark92] im Detail erläutert.

▪ Das 3-Ebenenkonzept von ANSI/X3/SPARC

Das 3-Ebenenkonzept wurde vom American National Standards Institute (ANSI) entwickelt, mit dem Ziel, eine effektive Datenbankorganisation zu erreichen. Die Drei-Ebenen dieses Konzeptes bilden die externe, die konzeptionelle und interne Ebene (Abb. 3.1):

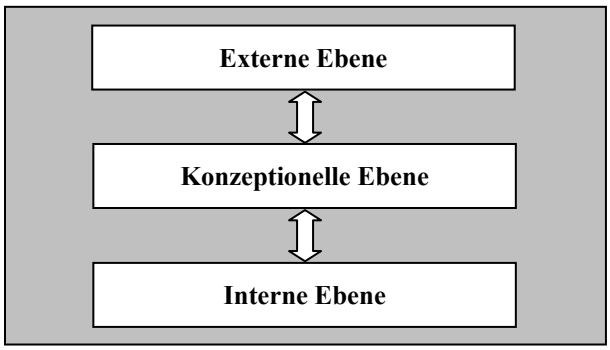


Abb. 3.1: Das 3-Ebenenkonzept von ANSI/X3/SPARC

Die externe Ebene beinhaltet die Beschreibung der Daten aus Benutzersicht. Es handelt sich hier um die Art und Weise, wie der Benutzer auf die Daten zugreift und sie logisch verknüpft. Hier wird eine Datenmanipulationssprache angewendet [Stah89]. Auf der konzeptionellen Ebene wird die logische Struktur der relevanten Daten in Form eines konzeptionellen Schemas beschrieben. Die Entwicklung dieses Schemas ist unabhängig von den Gesichtspunkten der Datenverarbeitung, also von den anderen Ebenen dieses Konzeptes. Die Entwicklung des konzeptionellen Schemas eines Ausschnittes der realen Welt ist eine Analyse dieses Ausschnittes, die eine beachtliche Investition mit sich bringen könnte. Das Entity-Relationship Model wird als ein geeignetes Darstellungsmittel zur Entwicklung des konzeptionellen Schemas betrachtet. Nach [Flat90] kann das konzeptionelle Schema in drei Schritten beschrieben werden:

- Objekte selektieren, Beziehungen feststellen, dann Objekte und Beziehungen durch ihre Eigenschaften (Attribute) darstellen,
- Bezeichnung der selektierten Objekte und Beziehungen und
- Klassifikation der Objekte und Beziehungen.

In der internen Ebene wird die physische Datenorganisation beschrieben. Es handelt sich dabei um die Festlegung der Speicherformen und der Zugriffspfade für die Daten, die bereits im konzeptionellen Schema erfasst wurden [Eign91]. Auf der Basis des 3-Ebenenkonzeptes von ANSI erstellt Thalheim [Thal91] einen 6-Phasen Entwurfsprozess zum Datenbankentwurf. In der ersten Phase dieses Entwurfsprozesses stehen die Anforderungsanalyse und die Spezifikation des Systems im Vordergrund. In der zweiten Phase wird das konzeptionelle Modell entwickelt. In der dritten, vierten und fünften Phase wird die Umsetzung des konzeptionellen Modells im Schema eines Datenbankmodells beschrieben. In der letzten Phase geht es um die physische Datenorganisation.

3.2 Darstellung der Basiskonzepte der Objektorientierte Technologie

▪ Abstraktion

Abstraktion bedeutet, das Wichtige über ein Ding zu berücksichtigen und die geringfügigen Bestandteile zu ignorieren. Datenabstraktion ist nach Abelson und Sussman [Abel91] der analoge Begriff für zusammengesetzte Daten. Abstrakte Datentypen werden nur über ihre Methoden definiert, wobei die anderen Bestandteile der Objekte, nämlich die Attribute und die Algorithmen, zur Realisierung der Methoden verkapselt bleiben. Traditionelle Programmiersprachen unterstützen das Konzept der Datenabstraktion. Ein Beispiel dafür sind die Datenstrukturen und Datentypen, die jedoch nur die Abstraktion von strukturellen Aspekten der Datenorganisation gestatten. Prozeduren und Funktionen sind andere Beispiele, welche die Datenabstraktion von Verhaltensaspekten der Datenorganisation erlauben. Die Objektorientierte Programmierung erlaubt jedoch die Kombination der beiden Abstraktionsarten [Ege92]. Die Realisierung des Konzepts der Datenabstraktion verstärkt die Softwarezuverlässigkeit und gewährleistet eine bessere Umsetzung des Konzeptes Geheimnisprinzip.

▪ Objekte und persistente Objekte

Jell [Jell91] definiert das Objekt als eine Zusammensetzung von Eigenschaften und Methoden die diese Eigenschaften manipulieren. Es ist ein nach außen abgeschlossener Komplex und reagieret mit einem vordefinierten Verhalten auf seine Umgebung. Durch seine Identität wird ein Objekt von allen anderen Objekten differenziert. Während seines Lebenslaufs kann ein Objekt diverse Zustände haben, besitzt aber zu einem bestimmten Zeitpunkt nur genau einen Zustand.

Die Lebensdauer eines Objektes ist an die Laufzeit des Programms gebunden, in dem dieses Objekt erzeugt wurde. Persistente Objekte besitzen eine anhaltende Existenz. Damit ist die Lebensdauer dieser Objekte nicht an die Programmausführung gebunden [Andr87]. Die Bedeutung der Persistenz der Objekte zeichnet sich durch die Notwendigkeit der Weiterbearbeitung von Objekten aus, die bei der Ausführung eines Programms erzeugt werden.

▪ Klassen und Pakete

In [Bolk87] wird eine Klasse als "logische Beschreibung einer Gruppe einzelner ähnlicher Objekte" betrachtet. Damit wird die Klasse als eine Zusammenfassung von Objekten mit gleichartigen Attributen und Methoden betrachtet. In [Fied91] wird die Klasse als "Ausgangspunkt einer Objektorientierten Umgebung" bezeichnet. Breutmann [Breu92] definiert den Begriff Klasse als "abstrakten Datentyp, der in Vererbungsbeziehungen zu anderen abstrakten Datentypen stehen kann". Nach Krause [Krau92] enthalten die Klassen "die Definition für die Struktur und das Verhalten von Objekten, die als Instantiierungen dieser Klasse erzeugt werden".

Pakete sind ein Konzept zur Verallgemeinerung von Einzelheiten, mit dem Ziel der Erstellung von übergeordneten Strukturen innerhalb eines Modells. Dieses Konzept wird angewendet, um bei der Objektorientierten Analyse die Klassen besser zusammenzustellen. Auf diese Weise können Klassen zu Paketen zusammengefasst werden. In UML kann ein Paket neben Klassen auch noch andere Pakete enthalten.

▪ Attribute

Durch Attribute werden Eigenschaften der Objekte definiert. Ein Attribut kann auch ein Objekt einer anderen Klasse sein [Copl92]. Attribute eines Objektes sollen zuerst identifiziert werden. Objekte einer Klasse haben die gleichen Attribute. Diese Attribute können aber unterschiedliche Attributwerte haben. Attribute einer Klasse können nur durch Methoden dieser Klasse manipuliert werden.

▪ Methoden und Botschaften

Die Funktionalität der Objekte wird durch Methoden definiert. Die Methode ist dabei eine Aktivität, die ein Objekt ausführt. Alle Objekte einer Klasse setzen die gleichen Methoden ein, wobei diese Methoden jeweils auf ein einzelnes Objekt angewendet werden. Methoden einer Klasse kennzeichnen ihr Verhalten. Methoden werden in UML 1.3 als Dienstleistungen betrachtet, die von einem Objekt mit einer Nachricht angefordert werden können, um ein bestimmtes Verhalten zu bewirken.

Mit dem Konzept der Überschreibung *(overriding)* können Unterklassen, Methoden, die sie von Oberklassen geerbt haben, unter den gleichen Namen, neu implementieren. Methoden können auch Überladen *(overloading)* werden. Damit kann ein Methodenname, innerhalb einer Klasse, mit unterschiedlichen Parametern benutzt werden.

Die Botschaft *(message)* ist eine Anweisung eines Objekts an ein anderes Objekt, um eine Methode gleichen Namens auszuführen. Die Menge der Botschaften wird als Protokoll der Klasse genannt. In UML können die Botschaften innerhalb der Interaktionsdiagramme dargestellt werden

▪ Assoziation

Assoziation ist ein Konzept zur Modellierung der Beziehungen zwischen Objekten unterschiedlicher Klassen eines Modells. Objekte derselben Klasse können miteinander in einer rekursiven Assoziation stehen. Die Kardinalität der Assoziation definiert die Menge der an der Assoziation mitwirkenden Objekte. Mit der Angabe der Kardinalitäten kann auch definiert werden, ob eine Kann- oder Muss-Assoziation zwischen den Objekten vorliegt. Assoziationen können mit Namen und Rollennamen beschrieben werden. Diese müssen angezeigt werden, falls zwischen zwei Klassen mehr als eine Assoziation vorhanden ist [Balz01]. Mit dem Ziel der Erhöhung ihrer Präzision können Assoziationen um Einschränkungsangaben *(constraints)* und Qualifikationsangaben *(qualifier)* erweitern werden.

Die Aggregation und Komposition werden in der UML als spezielle Formen der Assoziation betrachtet, in den zwischen den Objekten die Rangordnung „besteht aus“ oder „ist Teil von“ existiert. Bei der Komposition besteht zwischen den Objekten ein stärkerer semantischer Zusammenhang als es bei der Aggregation der Fall ist. So kann bei der Komposition ein Objekt der Teilklasse, zu einem bestimmten Zeitpunk, nur ein Teil eines einzigen Objektes der Aggregationsklasse sein. Ferner bezieht sich die dynamische Semantik des Ganzen auf seine Teile. Daher sollen die Teile des Ganzen zwangsläufig kopiert oder gelöscht werden falls das Ganze kopiert bzw. gelöscht wird. In Anhang A sind die unterschiedliche Assoziations- und Kardinalitätsarten gemäß UML 1.3 ausführlich dargestellt.

▪ Vererbung

Das Konzept der Vererbung zeichnet sich dadurch aus, dass Attribute und Methoden einer Oberklasse auf die Instanzen ihrer Unterklassen vererbt werden [Ott91] [Kemp93]. Attribute und Methoden, die vererbt werden sollen, müssen jedoch als *public* oder *protected* deklariert werden. Die Unterklassen können nicht nur die von den Oberklassen geerbten Attribute und Methoden besitzen, sondern auch ihre eigenen. Das Konzept der Vererbung kann nur auf Klassen eingesetzt werden. Die Objekte der Klassen können damit nicht miteinander verknüpft werden.
Vorteile der Vererbung sind u.a. die Förderung der Wiederverwendbarkeit, wobei die Struktur und das Verhalten eines Objekttyps in seinen Subtypen verwendbar sind.
Bei der Vererbung wird zwischen einfachen und mehrfachen Vererbungen unterschieden. Von einer einfachen Vererbung wird gesprochen, wenn eine Klasse nur eine Oberklasse besitzt. Eine mehrfache Vererbung tritt auf, falls eine Klasse über mehr als eine Oberklasse verfügt. Die mehrfache Vererbung führt oft zu Kontroversen von Attributen und Methoden. Zur Vermeidung dieser Kontroversen sind in [Moch93] Lösungsmechanismen dargestellt. Während in C++ sowohl die einfache als auch die mehrfache Vererbung unterstützt wird, unterstützt Java nur die einfache Vererbung.

▪ Geheimnisprinzip und Verkapselung

Das Geheimnisprinzip *(Information Hiding)* ist das absichtliche Verbergen interner Informationen eines Objektes. Damit können der Zustand eines Objektes und die Implementierung der Methoden innerhalb der Klasse geheim gehalten werden.
Verkapselung *(encapsulation)* bedeutet, dass Attribute und Methoden einer Klasse, in dieser Klasse verkapselt sind. Im Gegenzug zum Geheimnisprinzip können bei der Verkapselung der Zustand eines Objektes und die Implementierung der Methoden außerhalb der Klasse sichtbar sein. In Java kann die Verkapselung mit den Sichtbarkeitsstufen *public*, *protected* und *private*, ohne die Einhaltung des Geheimnisprinzips realisiert werden.

▪ Polymorphismus und dynamisches Binden

Polymorphismus beschreibt die Fähigkeit, mehr als eine Form anzunehmen. Polymorphismus als Konzept der Objektorientierung bietet die Möglichkeit, mit einer Nachricht an die Objekte unterschiedliche Klassen unterschiedliche Methoden aufzurufen. Es wird aber erst zur Laufzeit des Programms festgelegt, welche Methode durch die Nachricht aufgerufen wird. Damit können die Konzepte Polymorphismus und das dynamische oder späte Binden nicht getrennt von einander betrachtet werden. Während in C++ Methoden ausdrücklich als polymorph deklariert werden müssen, sind alle Methoden in Java ohne weiteres polymorph.

3.3 Objektorientierte Analyse

Die Objektorientierte Analyse (OOA) ist eine Beschreibung der fachlichen Anforderungen an ein Anwendungssystem in Objektorientierter Form [Sinz91]. Damit stehen Objekte und Klassen im Mittelpunkt dieser Beschreibung [Endr92]. Nach [Pall92] ist die Objektorientierte Analyse ein Konzept zur integrierten Beschreibung von Prozess- und Datensicht eines Ausschnittes der Realität.

Nach Rumbaugh [Rumb93] gliedert sich die OOA in Objektmodellierung, dynamische Modellierung, funktionale Modellierung und die Identifizierung der Methoden. Diese Vorgehensweise zur Objektorientierten Analyse wurde auf Grund ihrer Übersichtlichkeit von vielen Softwareentwicklern favorisiert. Die Objektorientierte Analyse kann in Anforderungsmodellierung und Analysemodellierung gegliedert werden.

3.3.1 Anforderungsmodellierung

Die Anforderungsmodellierung dient einerseits als detaillierte Spezifikation des zu modellierenden Systems und anderseits als Kommunikationsmittel zur Harmonisierung des Verständnisses des Problembereichs zwischen den Entwicklern und den Auftraggebern von Softwareprodukten. Der Kern dieser Modellierung bildet die Anwendungsfallmodellierung, in der die Systemfunktionalität durch der Definition der vom System unterstützten Anwendungsfälle dargestellt wird. Die Anwendungsfälle können sowohl textuell als auch anhand von Aktivitätsdiagrammen beschrieben werden. Bei der textuellen Beschreibung handelt es sich um eine detaillierte Darstellung der einzelnen Anwendungsfälle in Form eines Textes, wobei diese Darstellung eine kurze Beschreibung und die Akteure des Anwendungsfalles sowie Abläufe, Bedingungen und Fehlersituationen innerhalb dieses Anwendungsfalles beinhaltet. Die Anwendungsfallmodellierung kann in UML durch Anwendungsfalldiagramme und Aktivitätsdiagramme beschrieben werden. Im Rahmen der Anforderungsmodellierung wird ein Teil der Objekte des zu modellierenden Systems identifiziert. Diese Objekte werden dann zu Klassen zusammengefasst und in Klassendiagramme dargestellt werden.

Um Vereinbarungen für Anwendungsfälle oder Subsysteme zu spezifizieren, werden Schnittstellen definiert. Eine Schnittstelle *(interface)* in UML ist eine Zusammenfassung von Methoden, die eingesetzt werden, um die Bedienung einer Klasse oder Komponenten zu spezifizieren [Booc99]. Sie wird in UML grafisch als Kreis dargestellt. Schnittstellen werden als Konzept eingesetzt, um Assoziationen und Botschaftenaustausch zwischen Klassen einzurichten, die eine benötigte Schnittstelle implementieren, ohne sie unmittelbar mit einer bestimmten Klasse von Objekten zu verbinden.
Bei der Modellierung der Schnittstellen des Systems handelt es sich um Festlegungen, wie das System mit seiner Umgebung kommunizieren soll. Die Systemumgebung wird durch Akteure dargestellt, die mit dem zu entwickenden System in Interaktionen agieren. Da die Akteure sich in Personen und externe Systeme gliedern, können die Schnittstellen in Benutzerschnittstellen und Systemschnittstellen gegliedert werden. Benutzerschnittstellen sind solche Schnittstellen, die aus den Interaktionen zwischen den Akteuren Personen und den Anwendungsfällen des zu entwickenden Systems resultieren. Benutzerschnittstellen spielen eine wichtige Rolle bei der Beschreibung der Anwendungsfälle und insbesondere bei der Identifikation von relevanten Attributen.

Systemschnittstellen resultieren dagegen aus den Interaktionen zwischen den Akteuren externer Systeme und den Anwendungsfällen des zu modellierenden Systems. Da das entwickelte Softwaresystem mit externen Systemen wie Datenbank- oder CAD-Systemen kommunizieren wird, sollen solche Aspekte der Systemschnittstellen wie die Syntax und Symantik der Daten sowie Kommunikationsmechanismen definiert werden. Als Kommunikationsmechanismen können z. B. Präprozessor, API *(Application Programming Interface)* wie ODBC *(Open DataBase Connectivity)*, JDBC *(Java DataBase Connectivity)* und Java RMI *(Remot Method Invocation)* sowie CORBA *(Common Object Request Broker Architecture)* verwendet werden.
Das Schnittstellenmodell der Anforderungsanalyse des Systems wird in der Implementierungsphase, unter Berücksichtigung der verwendeten Programmiersprache spezifiziert. Die Programmiersprache Java unterstützt stark das Konzept der Schnittstellen.

3.3.2 Analysemodellierung

Bei der Analysemodellierung, die auf der Basis der Anforderungsmodellierung erfolgt, handelt es sich um eine detaillierte Beschreibung des zu modellierenden Systems. Diese Beschreibung wird in Strukturmodellierung und dynamische Modellierung gegliedert. Die Strukturmodellierung und die dynamische Modellierung können jedoch nicht streng getrennt voneinander erfolgen. So sollen deren wechselseitige Einwirkungen berücksichtigt werden.

Ein Aspekt der das gesamte Spektrum der Analysemodellierung deckt, ist die Identifizierung der Methoden. Die Identifizierung der potentiellen nützlichen Methoden ist eine komplizierte Aufgabe, da die Liste solcher Methoden unendlich sein kann. Die Identifizierung der Methoden kann in folgenden Schritten durchgeführt werden:

- **Erfassen der Methoden bei der Strukturmodellierung**
  Methoden, die bei der Strukturmodellierung erfasst werden, sind im Prinzip die eindeutig identifizierbaren Methoden. Diese müssen im Verlauf der dynamischen Modellierung geprüft werden, ob es sinnvoll, sie weiter als Methoden zu betrachten.
- **Erfassen der Methoden aus Ereignissen**
  Ereignisse, die bei der dynamischen Modellierung spezifiziert werden, können als Methoden erfasst werden.
- **Erfassen der Methoden aus Interaktionsdiagrammen**
  Nachrichten die innerhalb der Interaktionsdiagramme auftreten, können als Methoden identifiziert werden.
- **Erfassen der Methoden aus Zustandsdiagrammen**
  Aktionen und Aktivitäten innerhalb der Zustandsdiagramme können als Methoden identifiziert werden.

3.3.2.1 Strukturmodellierung

Nach der Anforderungsmodellierung, in der die Anforderungen ausführlich dargestellt und modelliert werden, werden bei der Strukturmodellierung die statischen Aspekte des Systems beschrieben. Als Ergebnis dieser Modellierung werden die Klassen des Systems, die Assoziationen zwischen diesen Klassen, die Vererbungsstrukturen und die Attribute sowie die in dieser Phase identifizierten Methoden in Form von Klassendiagrammen dargestellt. Dadurch werden die identifizierten Objekte des Systems in Klassen zusammengefasst. Klassen, die gemeinsame Struktur und Verhalten besitzen, werden in Hierarchien eingeordnet. Bei der Strukturmodellierung werden die Basiskonzepte der Objektorientierung wie Klassen, Attribute, Assoziation, Pakete und Vererbung eingesetzt. Die Strukturmodellierung, die den Bezugsrahmen für die dynamische Modellierung bereitstellt, kann in folgenden Schritten durchgeführt werden:

- **Identifizierung der Objektklassen**
  Die Identifizierung der Objekte beginnt mit der Anforderungsanalyse. Dort können bereits bei der Anwendungsfallmodellierung die Basisobjekte identifiziert werden. Ein Ansatz zur Identifizierung der Objekte, der sich in der Praxis bewährt hat, besteht darin, alle für das zu modellierende System relevanten Substantive aus der Problembeschreibung zu betrachten. Damit entsteht eine Liste von Klassenkandidaten, die möglicherweise noch um Begriffe aus der Anwendungswelt ergänzt werden kann. Die Liste von potentiellen Klassen enthält vermutlich unnötige sowie unkorrekte Klassen wie z. B. redundante Klassen, irrelevante Klassen, vage Klassen, Methoden, Rollen anderer Klassen sowie Klassen, die eigentlich Attribute anderer Klassen sein sollen. Derartige Klassen werden aus der Liste entfernt.

- **Identifizierung der Assoziationen**
  Bei der Strukturmodellierung werden die Beziehungen zwischen den Klassen sowie der Verweis von einer Klasse auf eine andere durch Assoziationen modelliert. An dieser Stelle sollte darauf hingewiesen werden, dass Assoziationen in Objektorientierten Programmiersprachen in der Regel als Referenzattribute von Klassen implementiert werden. Bei der Identifizierung der Assoziationen werden zunächst alle potentiellen Assoziationen zwischen vorhandenen Klassen aus der Problembeschreibung extrahiert. Danach werden unnötige sowie falsche Assoziationen eliminiert. Darunter fallen unter anderem irrelevante Assoziationen, Aktionen oder abgeleitete Assoziationen.

- **Identifizierung der Attribute**
  Ein Teil der relevanten Attribute lässt sich direkt aus der Problembeschreibung herauslesen. Bei den Attributen ist jedoch zu beachten, dass diese bei weitem nicht so vollständig in der Problembeschreibung auftauchen, wie es z. B. bei den Objekten der Fall ist.
  Abgeleitete Attribute, ebenso wie abgeleitete Objekte und Assoziationen, sind in der Lage, eindeutige Eigenschaften zu verallgemeinern. Sie sollen deswegen deutlich von den Hauptattributen gesondert, benannt oder gar weggelassen werden. Im nächsten Schritt werden die geeigneten Attribute nach bestimmten Kriterien einbehalten und die ungeeigneten ignoriert.

- **Bereitstellung eines *Data dictionary***
  Hier wird der Bedeutungsgrad der Klassen und die möglichen Annahmen und Beschränkungen in Zusammenhang mit deren Anwendung sowie Assoziationen, Attribute und die bis hier identifizierten Methoden dargestellt.

- **Verfeinern des Strukturmodells durch Vererbung**
  Ziel dieses Schrittes ist, durch das Konzept der Vererbung gemeinsame Strukturen von Klassen zu entwickeln. Obwohl die mehrfache Vererbung die kollektive Nutzung von mehr Daten gestattet, steigert sie aber die Komplexität in allen Phasen der Modellierung und Implementierung eines Systems. Deswegen soll die mehrfache Vererbung nur dann eingesetzt werden, wenn es keine andere Alternative gibt.

- **Überprüfen der Zugriffspfade**
  Hier werden die Zugriffspfade in den Klassendiagrammen untersucht. Dabei wird geprüft, ob sie zweckmäßige Folgen zulassen. Um fehlende Informationen im Strukturmodell auszuschließen, sollen geeignete Fragen gestellt werden und danach Antworten auf diese Fragen gefunden werden.

- **Allgemeine Verfeinerung des Strukturmodells**
  In diesem Schritt sollen die vielleicht noch fehlenden Objekte nach bestimmten Kriterien gesucht werden. Es ist hier auch sinnvoll, das Strukturmodell noch einmal zu validieren, z. B. indem die Bestandteile des Modells genauer nach ihrem Sinngehalt überzuprüfen sind. Allerdings ist eine definitive Verfeinerung des Strukturmodells nur nach der dynamischen Modellierung des Systems möglich.

3.3.2.2 Dynamische Modellierung

Die dynamische Modellierung wird als eine Ergänzung der Strukturmodellierung betrachtet. Während die Strukturmodellierung rein statische Informationen darstellt, dient die dynamische Modellierung der Beschreibung des zeitabhängigen Systemverhaltens. Grundsätzlich beschreibt die dynamische Modellierung den zulässigen Veränderungen der Objekte des Strukturmodells. Objekte wechseln niemals ihre Klassenzugehörigkeit, jedoch ihren Zustand. Dadurch kann die dynamische Modellierung als Verfeinerung des Strukturmodells betrachtet werden. Die Identifizierung der Methoden der Objekte kann nicht vollständig abgeschlossen werden, wenn die dynamische Modellierung beginnt. Methoden können deshalb auch innerhalb der dynamischen Modellierung identifiziert werden.
Die Basis zur dynamischen Modellierung bilden die folgenden zwei Schritte:

- **Entwicklung von Szenarios**
  Szenarios sind Dialoge zwischen dem System und der Außenwelt. Diese Dialoge werden nun aufgeschrieben, beginnend mit den Normalfällen, also denjenigen Interaktionssequenzen, die keine Ausnahmefälle oder Fehler enthalten. Danach werden Benutzerfehler und Ausnahmesituationen festgehalten. Jedes solcher Szenario besteht aus einer Folge von Ereignissen.

- **Identifizierung der Ereignisse**
  Der Informationsaustausch zwischen einem Objekt des Systems und einem Akteure wird als Ereignis betrachtet. Ein Akteur kann in diesem Fall ein Benutzer, ein Anwendungssystem oder auch ein anderes Objekt des Systems sein.

Zur Spezfikation der Dynamik des zu modellierenden Systems, welche auch nur mit einer verbalen Beschreibung der Anwendungsfälle erfolgen kann, bietet die UML (V. 1.3) folgende Diagramme:

- **Interaktionsdiagramme** zur Beschreibung des Verhaltens zwischen Objekten zueinander. Damit werden die dynamischen Aspekte eines Systems anhand von Objekten und deren Beziehungen, sowie die austauschbaren Nachrichten zwischen diesen Objekten analysiert. Bei dem Einsatz der Interaktionsdiagramme wird normalerweise das Verhalten der einzelnen Anwendungsfälle modelliert. Die Interaktionsdiagramme gliedern sich in:
  - **Sequenzdiagramme**, wobei hier der zeitliche Ablauf der Nachrichten von Bedeutung ist.
  - **Kollaborationsdiagramme**, in denen die Struktur der Objekte, die gegenseitig Nachrichten austauschen, relevant ist.

- **Aktivitätsdiagramme** beschreiben die Reihenfolge der Aktivitäten. Sie sind ein Spezialfall der Zustandsdiagramme, wobei fast alle Zustände Aktivitätszustände sind und fast alle Zustandsübergänge durch den Abschluss von Aktivitäten im Ursprungszustand ausgelöst werden. Nach Hruschka [Hrus98] wurden die Aktivitätsdiagramme in die UML hinzugefügt, nicht weil man sie braucht, sondern vielmehr aus Marketingmotiven.

- **Zustandsdiagramme** zur Modellierung der Dynamik der Objekte nicht nur innerhalb eines Anwendungsfalles sondern ihr Verhalten auch im bezug auf andere Anwendungsfälle. Dadurch können Objekte mit außerordentlichem dynamischem Verhalten beschrieben werden. In diesen Diagrammen, die zuerst aus einer einfachen Folge von Zuständen und Ereignissen bestehen, wird ein Intervall zwischen zwei Ereignissen als ein Zustand gezeigt, der einen eindeutigen Namen haben soll. Bei der Erstellung von Zustandsdiagrammen können die beschriebenen Zustände von Objekten zur Verfeinerung ihrer Attribute verwendet werden.

3.4 Objektorientierter Entwurf

Nach [Hurs95] entspricht der Objektorientierte Entwurf *(Object Oriented Design, OOD)* dem Abbilden der Objektorientierten Analyse in einer Programmiersprache. In [Rumb93] wird der Objektorientierte Entwurf als eine Zusammenführung von Objektmodell, dynamischem Modell und funktionalem Modell betrachtet. Hier werden die entwickelten abstrakten Strukturen in softwareorientierte Strukturen wie z. B. Datenstrukturen und Funktionen übertragen. Um den Neuentwicklungsumfang in der Entwurfsphase zu verringern, hat sich bewährt, von bereits entwickelten Objektbibliotheken wie z. B. graphischen Kernsystemen, Anwendungsoberflächen und Datenbanken, im eigenen Projekt Gebrauch zu machen.
Der Objektorientierte Entwurf kann in Systementwurfsmodellierung und Detailentwurfsmodellierung erfolgen.

3.4.1 Systementwurfsmodellierung

Bei der Systementwurfsmodellierung wird die Architektur des Systems modelliert. Hier werden solche Entscheidungen getroffen, die hauptsächlich das gesamte System betreffen. In diesen Rahmen sollen hier folgende Schritte gemacht werden:

- Festlegung der Implementierungsplattform.
- Auswahl der Mechanismen zur Datenhaltung.
- Festlegung der Verteilungs- und Kommunikationsstrukturen in Form von Komponenten- und Verteilungsdiagrammen.
- Im letzten Schritt soll die verwendete Softwarearchitektur festgelegt werden, wobei hier genau festgestellt wird, ob ein Kontrollfluss *(single threads of control)* oder nebenläufige Prozesse *(multiple threads of control)* verwendet werden.

3.4.2 Detailentwurfsmodellierung

Die Detailentwurfsmodellierung befasst sich hauptsächlich mit der Verfeinerung des Strukturmodells und des dynamischen Modells einschließlich der Festlegung der Optimierungen und die Modifikation von bestimmten Modellelementen in Hinblick auf deren künftigen Wiederverwendbarkeit. Bei der Detailentwurfsmodellierung können folgende Schritte unternommen werden:

- Auflösung von Strukturen, welche in der festgelegten Implementierungsplattform nicht direkt implementierbar sind, wie z. B. die Mehrfachvererbung, die in Java nicht ohne weiteres implementierbar ist.
- Erkennen von häufig wiederkehrenden Entwurfsproblemen und der Einsatz von Entwurfsmuster *(design pattern)* als bewährte Lösungen solcher Probleme.
- Verfeinern der Klassendiagramme, wie z. B. durch Ergänzung, um spezifische Attribute und Methoden.
- Festlegung und Eintragung der geeigneten Sichtbarkeiten der Klassen in Hinsicht auf die Implementierungsplattform.

In den Literaturquellen ist es unterschiedlich festgelegt, wo die Grenze zwischen dem Objektorientierten Entwurf und der Objektorientierten Implementierung ist. Wo diese Grenze liegt, soll eigentlich eine Nebenrolle spielen. Hauptsächlich soll das Entwurfsmodell durchgängig verfeinert werden bis es implementierungsfähig ist.

3.5 Objektorientierte Implementierung

3.5.1 Objektorientierte Programmierung

Das Paradigma der Objektorientierung ist im Wesentlichen durch die Entwicklung in höheren Programmiersprachen geprägt. Diese lässt sich in drei Phasen aufteilen:

- Imperative Programmiersprachen wie COBOL und Fortran, in denen Variable, Felder und rudimentäre Kontrollstrukturen verfügbar sind.
- Strukturierte Programmiersprachen wie Algol, Modula-2 und Pascal in denen Blockstruktur und Kapselung vorhanden sind.
- Objektorientierte Programmiersprachen wie Simula, Eiffel, Smalltalk, C++ und Java in denen die Objektorientierte Sicht die Grundlage bildet.

Um die Semantik der realen Welt effizienter auf die Hardware übertragen zu können, wurde die Objektorientierte Programmierung entwickelt, [Bolk87]. Nach [Prin99] soll eine Objektorientierte Programmiersprache die Konzepte der Objektorientierung wie Datenabstraktion, Datenkapselung, Vererbung und Polymorphie unterstützen.

3.5.2 Objektorientierten Datenhaltung

▪ Objektorientierte Datenbankmodelle

Im Relationalenmodell, wo ein System mit Listen-Typen modelliert werden kann, ist eine effektive Modellierung von Ausschnitten der realen Welt nur begrenzt möglich. Misslungen ist auch der Versuch, semantische Datenmodelle einzusetzen, da hier die operationelle Spezifikation fehlt. Objektorientierte Datenbankmodelle scheinen eher geeignet, diese Anforderungen zu erfüllen. Einem Objektorientierten Datenbanksystem (OODBS), liegt ein Objektorientiertes Datenbankmodell (OODBM) zugrunde, dessen Basis Objekte im Sinne der Objektorientierung sind. Es ist sinnvoll an dieser Stelle zu bemerken, dass in manchen Literaturquellen auch der Begriff Objektorientierte Datenmodelle (OODM) als Synonym für Objektorientierte Datenbankmodelle vorkommt.

Nach Dittrich [Ditt90] können die Objektorientierte Datenmodelle als eine Kombination von Konzepten der klassischen Datenmodelle, der Objektorientierten Programmierung und der Wissensrepräsentation betrachtet werden. Nach Vossen [Voss92] muss ein Objektorientiertes Datenmodell die Prinzipien der Objektorientierung erfüllen.

Vossen [Voss99] definiert das OODM als „Übertragung der Deklarationsmöglichkeiten für Strukturen, wie sie in höheren Programmiersprachen üblich sind, und insbesondere abstrakter Datentypen mit der Möglichkeit, Verhalten und Struktur zu kapseln, auf den Datenbankbereich“. Außerdem muss ein OODM die Hauptkonzepte der Objektorientierung umsetzen. Dazu gehört die Realisierung der folgenden Modellierungseigenschaften:

- Darstellbarkeit komplexer Objekte.
- Objektidentität.
- Klassen als Hauptbestandteil des Datenbankschemas.
- Vererbung.
- Verkapselung der Struktur und des Verhaltens.
- Die Möglichkeit zur Überladung der Methodennamen in der Vererbungshierarchie bzw. dem Klassenverband.
- Die Unterstützung von vordefinierten Datentypen.

In [Acht97] wird an dieser Stelle von Objektorientierten Datenbanken gesprochen. Diese müssen neben den Hauptkonzepten der Objektorientierung auch andere Konzepte wie z. B. Integritätsprüfung, Datensicherheit sowie Transaktionssteuerung realisieren. In [Wann98] wird ein OODBM anhand eines Strukturteils und eines Methodenteils dargestellt. Dem Strukturteil liegen Konzepte der Objektorientierung wie Objekte und deren Identität, Klassen und deren Beziehungen, Strukturvererbung und Integritätsbedingungen zugrunde. Der Methodenteil befasst sich mit den Methoden, die auf die Objekte ausführbar sind.
In [Heue97] sind Konzepte, Modelle und Standards der Objektorientierten Datenbanken und -systeme dargestellt. Objektorientierte Datenbanksysteme sind hier nach kommerziellen Systemen und Prototypen eingeteilt. Bei den ausführlich behandelten Systemen wird ihre Systemarchitektur sowie ihre interessantesten Eigenschaften und Komponenten beschrieben.

▪ Objektorientierte Datenbanksysteme

Datenbankunterstützung ist nicht nur im kommerziellen Bereich, wo sich die relationalen Datenbanksysteme durchgesetzt haben, erstrebenswert, sondern auch in vielen anderen Anwendungsgebieten wie im Bauwesen, im Maschinenbau, sowie der Unterstützung von Expertensystemen. In diesen neuen Anwendungsgebieten sind komplexe Objekte zu verwalten. Relationale Datenbanksysteme können diese Anforderungen nicht erfüllen. Diese spezifischen Anforderungen führten zur Entwicklung von Objektorientierten Datenbanksystemen (OODBS), in denen die Konzepte der Objektorientierung einen Niederschlag finden.

Lockemann [Lock91] definiert das OODBS als System zur Verwaltung von Datenbankobjekten. Er definiert das Datenbankobjekt als "Dateneinheit, der in einem gegebenen Anwendungszusammenhang ein Gegenstand der Anschauung in einer physischen oder gedanklichen Umwelt zugeordnet werden kann". Datenbankobjekte, die das Konzept der Aggregation erfüllen, bezeichnet er als komplexe Objekte.

Die wesentlichen Unterschiede der Objektorientierten Datenbanksysteme zu den Objektorientierten Sprachen sind zum einen die persistente Speicherung von Objekten über die Programmlaufzeit hinaus und zum anderen die Transaktionseigenschaften. Neben den Konzepten der Objektorientierung muss ein OODBS grundsätzlich folgende Konzepte realisieren, wobei diese die klassischen Datenbanksysteme charakterisieren:

- Vollständigkeit: Das System ist vollständig im Sinne der Berechenbarkeitstheorie. Alle berechenbaren Funktionen können damit ausgedrückt werden.
- Speicherverwaltung: Daten werden für den Benutzer unsichtbar im Speicher verwaltet.
- Concurrency Control: Mehrere Benutzer können gleichzeitig arbeiten.
- Recovery: Das System kann nach einem Absturz wieder restauriert werden.
- Ad hoc-Anfragen: Anfragen von Benutzer sollen sofort beantwortet werden.

▪ Objektrelationale Datenbanksysteme

Objektrelationale Datenbanksysteme sind relationale Datenbanksysteme, in denen Konzepte der Objektorientierung realisiert sind. Dabei werden diese traditionellen Datenbanksysteme so erweitert, dass sie imstande sind, u. a. komplexe Objekte darzustellen. Durch die Verabschiedung von SQL als Norm, in der Konzepte der Objektorientierung unterstützt werden, wird die Integration solcher Konzepte in den relationalen Datenbanksystemen verankert [ISO/IEC1], [ISO/IEC2], [ISO/IEC9], [ISO/IEC10]. In [Ston99], [Voss99] und [Balz01] sind Objektrelationale Datenbanken näher beschrieben. Ein objektrelationales Datenbanksystem soll u. a. folgende Anforderungen erfüllen [Voss99]:

- Darstellung komplexer Objekte.
- Definition neuer Typen aus gegebenen Basistypen oder neu definierten Typen.
- Definition einer Relation als Spezialisierung einer anderen, wobei diese deren Attribute oder Prozeduren erben kann.
- In der Zeit eines Updates soll eine Integritätssicherung durch Regeln garantiert werden.

In [Wann98] wird der Entwurf eines Objektorientierten Datenbankmodells für relationale Datenbanksysteme beschrieben. Hier werden die Struktur, das Verhalten und die Funktionalität dieses Datenbankmodells dargestellt. Ferner wird die Kopplung dieses Modells mit dem relationalen Datenbanksystem detailliert dargestellt.
Die großen Anbieter relationaler Datenbanksysteme wie z. B. IBM, Informix und Oracle bieten objektrelationale Erweiterungen ihrer Systeme an. Das erste kommerziell verfügbare objektrelationale Datenbanksystem war der Universal Server von Informix, der die Stärken von relationaler und Objektorientierter Datenbanktechnologie in sich vereinigt. Grundsätzlich können Objekte der Datenbank mit Hilfe von Softwarekomponenten, sogenannten *DataBlades*, hinzugefügt werden. Informix hat bereits wichtige *DataBlades* für Text, Bild und räumliche Daten entwickelt.

3.6 UML und die Standardisierung der Objektorientierten Technologie

Die *Object Management Group* (OMG) wurde 1989 mit dem Ziel gegründet, Objektorientierte Technologien zu fördern und zu standardisieren. Ausgangspunkt ist eine heterogene Objektorientierte Umgebung, in der Clients und Server ihre Aufgaben realisieren können.
Dazu wurde die *Object Management Architecture* (OMA) entworfen, welche aus 4 Komponenten besteht:

- *Object Request Broker* (ORB) ist eine Schnittstelle zwischen den Systemkomponenten und bildet die Basis der *Common Object Request Broker Architecture* (CORBA).
- *Object Services* (OS) sind Schnittstellen zu Systemdiensten.
- *Application Objects* sind die Anwendungsobjekte.
- *Common Facilities* stellen allgemeine Dienste zur Verfügung.

Die *Object Database Management Group* (ODMG) als Untergruppe der OMG hat das Ziel, eine Standardisierung der Objektorientierten Datenbanksysteme zu erreichen.
Gegenstand dieser Standardisierung ist u. a. die Entwicklung von einem Objektmodell, Objektdefinitionssprachen wie die *Object Definition Language* (ODL) und die *InterfaceDefinition Language* (IDL).
Weiter wird innerhalb dieser Standardisierung an der Anbindung der Objektorientierten Datenbanksysteme an Objektorientierte Programmiersprachen wie Smalltalk, C++ und Java sowie die Anbindung dieser innerhalb einer CORBA-Umgebung gearbeitet. Ferner wird ein

ODMG-Objektmodell zur Spezifikation der Konzepte, die von einem OODBS unterstützt werden, dargestellt.
Die Basiselemente dieses Objektmodell sind Objekte (mit Identität) sowie Literale (ohne Identität), wobei Objekte und Literale anhand von Typen kategorisiert werden. Der Zustand der Objekte wird anhand ihrer Attribute sowie ihrer Beziehungen zueinander definiert.
Das Verhalten der Objekte wird anhand der Methoden, die auf diese Objekte ausgeführt werden können, definiert.

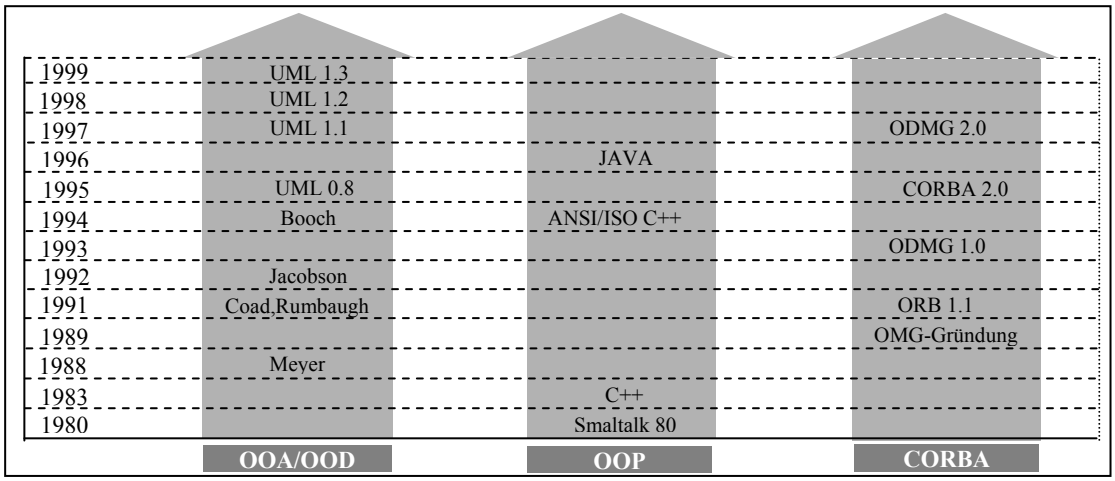


Abb. 2.7: Tendenzen der Entwicklung und Standardisierung der Objektorientierte Technologie

In Abbildung 2.7 sind die Tendenzen der Entwicklung und Standardisierung der Objektorientierten Technologie dargestellt.

Zur Spezifikation von Modellen der Softwaresysteme wurde die *Unified modeling language* (UML) entwickelt. Sie stellt eine evolutionäre Design-Notation dar, die ihren Ausgangspunkt in den Objektorientierten Modellierungsmethoden von Booch, OMT und OOSE hat.
Diese drei Methoden wurden einander angepasst mit dem Ziel, eine einheitliche Modellierungsmethode zu bekommen. UML besitzt die notwendige Flexibilität, um neue Softwarekonzepte erfolgreich einzubeziehen. Die UML hat sich mittlerweile als anerkannter Standard durchgesetzt.

4 Darstellung des Systems der Kosten in Hochbauprojekten

4.1 Allgemeine Beschreibung

4.1.1 Klassifizierung der Datenbasis im Hochbau

Die Menge der Daten, die von den Funktionen eines Informationssystems benötigt bzw. von ihnen erzeugt werden, werden als Datenbasis betrachtet. Dieser Begriff stellt eine Verallgemeinerung des Begriffs Datenbank dar [Blei90].

Als Datenbasis wird die Menge aller permanent verwalteten Datenbestände eines Informationssystems bezeichnet [Trau91].

Die Datenbasis wird anhand von Datenhaltungskonzepten wie z. B. Datenbanksystemen für die unterschiedlichen Bereiche der Produktion zur Verfügung gestellt (Abb. 4.1).

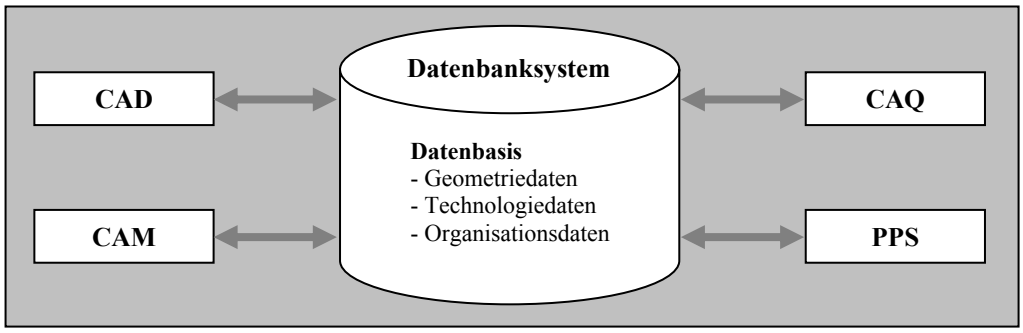


Abb. 4.1: Bereitstellung der Datenbasis anhand von Datenbanksystemen

Bei der Gliederung der Datenbasis unterscheidet man zwischen Daten, die sich als Merkmale von den konkreten Objekten ergeben, und den Stammdaten, die bei der Produktionsplanung und -steuerung häufig wiederholt auftreten [Rist85] [Grab79].

Wissensdaten wie Fakten und Regeln, die bei der Entwicklung von Expertensystemen von großer Bedeutung sind, werden als Stammdaten eingestuft. Bei der Gliederung der Daten in einem Unternehmensdatenmodell wird zwischen allgemeinen und speziellen Daten differenziert [End90].
Daten können auch nach ihrer Lebensdauer, ihrem Aufbau und ihrer Dynamik sowie ihrer Zugriffs- und Speichercharakteristik gegliedert werden.

Daten im Bauwesen sind nach REFA [REFA84] ein Überbegriff für Zahlen, Fakten und Zusammenhänge und werden für Planung, Steuerung, Kontrolle und Entlohnung verwendet. Bei der Baukalkulation werden die Planungsdaten bei der Vorkalkulation und die Kontrolldaten bei der Nachkalkulation eingesetzt.

Abb. 4.2: Relevante Daten im Hochbau

Datenermittlung bedeutet die Ermittlung der Zeit, Menge, Arbeitsbedingungen und Einflussgrößen. Die für die Modellierung im Hochbau relevanten Daten gliedern sich in geometrische, technische organisatorische und betriebliche Daten (Abb. 4.2).

Die für die Kostenermittlung im Hochbau notwendigen Geometriedaten, insbesondere Massen und Zeichnungsdaten aus Grundrissen, werden vor allem aus dem geeigneten CAD-System eingeführt.

4.1.2 Abgrenzung des Systems der Kosten vom Gesamtsystem

Zur Abgrenzung eines Systems der Kosten vom Gesamtsystem ist eine Analyse der Umgebung dieses Systems notwendig. Ziel dieser Analyse ist es, sich einen allgemeinen Eindruck über den zu modellierenden Ausschnitt der realen Welt und seine Umgebung zu verschaffen.

Abb. 4.3: Strategie zur Planung und Errichtung von Bauvorhaben im Hochbau

In Abbildung 4.3 ist eine Strategie zur Planung und Errichtung von Bauvorhaben im Hochbau dargestellt.

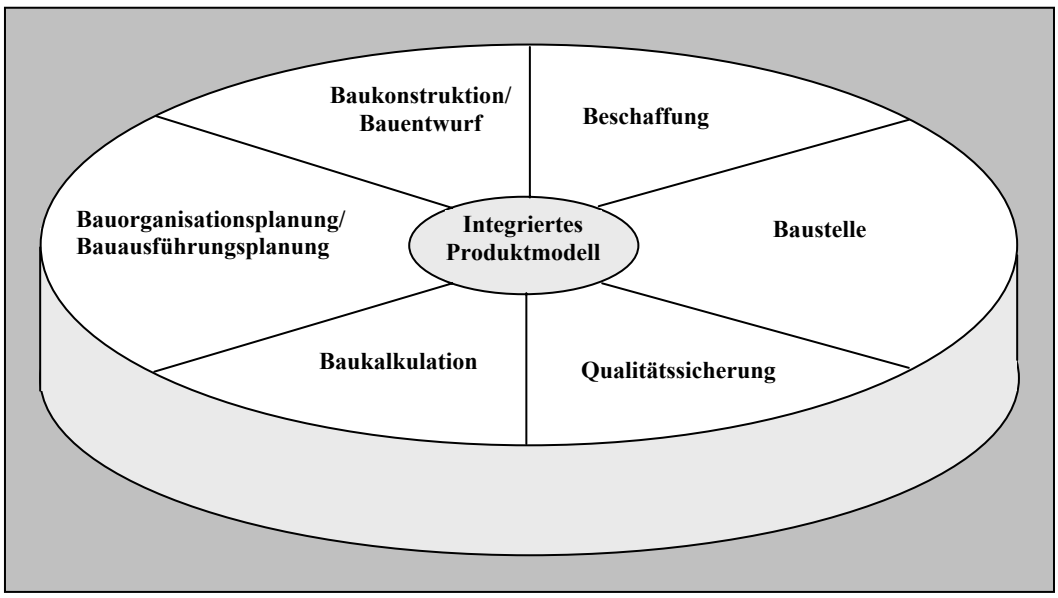


Abb. 4.4: Darstellung des Gesamtsystems Hochbau anhand von Produktmodellsichten

Das Gesamtsystem Hochbau kann durch Produktmodellsichten dargestellt werden. In Abbildung 4.4 sind anwendungsspezifische Sichten eines integrierten Produktmodells im Hochbau dargestellt. Dieses Gesamtsystem ist eine Vereinigungsmenge aller im Bauproduktionsprozess relevanten Daten und deren semantischen Verknüpfungen. Es vereint alle in diesem Prozess beteiligten Komponenten wie z. B. Statik, Baukonstruktion, Bauausführungsplanung, Baustelle und Baubetrieb/Baukalkulation. Diese Komponenten sind besonders in der stationären Industrie stark rechnergestützt.

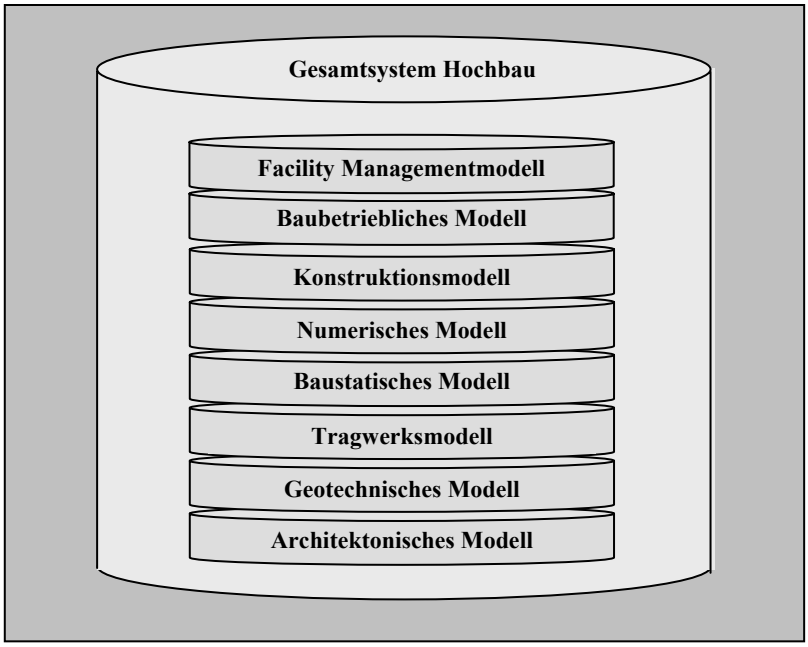


Abb. 4.5: Klassifizierung der Partialmodelle im Hochbau

In Hinsicht auf die Übersichtlichkeit der Darstellung kann das Gesamtsystem Hochbau in Partialmodelle zerlegt werden (Abb. 4.5). Diese Partialmodelle decken den gesamten Produktlebenslauf im Hochbau ab.

In der Planungsphase wird die Gebäudegeometrie durch das **Architektonische Modell** dargestellt.

Das **Geotechnische Modell** beschreibt u. a. die Bodeninformation, die technische Infrastruktur und die relevanten Daten des geographischen Informationssystems. Dieses Modell ist in [Diaz97] auf Objektorientierter Basis beschrieben.

Im **Tragwerksmodell** werden die Zusammenhänge der Geometrie zwischen den Gebäudebauteilen dargestellt. Eine Verknüpfung des architektonischen Modells mit dem Tragwerksmodell ist in [Rüpp94] auf Objektorientierter Basis beschrieben.

Im **Baustatischen Modell** werden die statischen Zusammenhänge der Bauteile dargestellt.

Das **Numerische Modell** beschreibt u. a. die Rand- und Übergangsbedingungen zwischen den Gebäudebauteilen oder zwischen Gebäude und Boden sowie Verformungsgrößen und Lasten.

Im **Konstruktionsmodell** werden u. a. die konstruktive Zusammensetzung und Fertigung der Bauteile unter Berücksichtigung der gültigen Normen dargestellt.

Im **Baubetriebliches Modell** werden u. a. die baubetrieblichen Informationen und die zur Bauausführung notwendigen Maßnahmen beschrieben. Dieses Modell dient der Verbesserung der Wirtschaftlichkeit und Effizienz der Bauausführung. Innerhalb dieses Modells werden die Kosten in Hochbauprojekten kalkuliert.

Das System der Kalkulation in Hochbauprojekten wird vom System des baubetrieblichen Rechnungswesens abgeleitet.

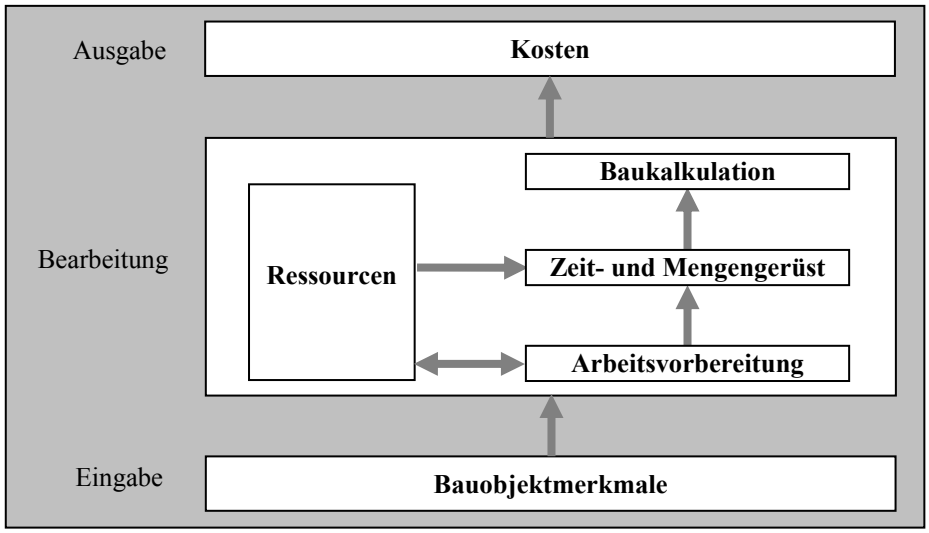


Abb. 4.6: Das Modell zur Bildung der Kosten im Hochbau

Grundsätzlich kann ein Modell zur Bildung der Kosten im Hochbau in drei Phasen gegliedert werden (Abb. 4.6):

- **Eingabe**, wobei hier die Merkmale eines Objektes eingelesen werden.
- **Bearbeitung**, wobei hier die Merkmale der Bauteile anhand der Arbeitsvorbereitung und Ressourcen zu einem Bauobjekt bearbeitet werden, und Zeit- und Mengengerüst daraus abgeleitet und für die Kalkulation bereitgestellt werden.
- **Ausgabe**, wobei in dieser Phase die Kosten als Ausgabeergebnis stehen.

Das **Facility Managementmodell** beinhaltet die notwendigen Daten zur Gebäudeverwaltung. Grundlagen für eine solche effiziente Gebäudeverwaltung sind u. a. eine einheitliche Datenbasis, die man mühelos und preiswert pflegen kann sowie eine preiswerte und effiziente Datenbeschaffung und ein Datenmanagement auf höchstem Aktualitätsgrad.

4.2 Beschreibung des Systems der Baukalkulation in Hochbauprojekten

4.2.1 Grundbegriffe der Kostenrechnung in Hochbauprojekten

Nachfolgend werden die Bestandteile des Systems der **A**usschreibung, **V**ergabe und **A**brechnung des Bauprojektes (AVA) dargestellt.

- Die **Ausschreibung** eines Bauprojektes ist die Initiierung des Auftraggebers, um für sein gewünschtes Bauwerk Angebote zu bekommen. Normalerweise wird die Ausschreibung mit einem Leistungsverzeichnis erläutert. Um den Auftrag zu bekommen, wird seitens der Bauunternehmer eine Angebotskalkulation vorgenommen.

- Bei der **Vergabe** des Bauprojektes wird der Auftragnehmer vom Auftraggeber nach bestimmten Kriterien ausgesucht und dann der Bauvertrag abgeschlossen. Zur Ermittlung der anfallenden Kosten wird innerhalb der Bauzeit seitens des Bauunternehmens eine Arbeitskalkulation vorgenommen.

- Bei der **Abrechnung** des Bauprojektes werden die Vergütungsansprüche des Auftragnehmers dargestellt. Dadurch können mit der **Nachkalkulation** die tatsächlich angefallenen Kosten ermittelt werden.

Kosten in Hochbauprojekten sind Aufwendungen für Güter, Leistungen und Abgaben einschließlich Umsatzsteuer, die für die Planung und Errichtung von Bauprojekten des Hochbaus erforderlich sind. Bei der Kostenermittlung werden die Kosten für gleichartige Aufwendungen in Kostengruppen zusammengefasst. Die Summe aller Kostengruppen wird als Gesamtkosten betrachtet. Kosten können entsprechend der Kostenart (z. B. Fertigungslöhne) den Kostenstellen (z. B. Maurerkolonne) und Kostenträger (z. B. Wände) zugeordnet werden.

Bei der Zuordnung der Kosten ist die Kostenrechnung in [Götz99] nach dem Informationsziel in drei Bereiche gegliedert:

- **Kostenartenrechnung**, wobei hier im Mittelpunkt steht, welche Kosten angefallen sind oder anfallen werden. Die **Kostenartenstruktur** wird als wichtiges Ordnungssystem im baubetrieblichen Rechnungswesen betrachtet. Die Strukturierung der Kostenarten wird von Bauunternehmen, entsprechend der Anforderungen aus baufachlicher Sicht, unterschiedlich dargestellt. Die Kostenartenstruktur beinhaltet Informationen, die bei einem EDV-Einsatz als Stammdaten abgespeichert und bei Bedarf von da aus abgerufen werden.

- **Kostenstellenrechnung**, wobei hier im Mittelpunkt steht, wo die Kosten angefallen sind oder anfallen werden. Als Grundlage für die Kostenstellenrechnung bilden die **Kostenstellen** im Bauunternehmen, auf denen Kosten verursacht bzw. erfasst werden. Als mögliche Kostenstellen können Kostenstellen der Verwaltung, Hilfsbetriebe und Verrechnungskostenstellen sowie Baustellen betrachtet werden.

- **Kostenträgerrechnung**, wobei hier im Mittelpunkt steht, wofür die Kosten angefallen sind oder anfallen werden. Bei der Baubetriebsrechnung werden die anfallenden Kosten den Kostenstellen zugeordnet. So können die Kostenstellen, die sich auf eine Baustelle beziehen, als **Kostenträger** betrachtet werden.

Nachfolgend wird der Unterschied zwischen der Kostenermittlung in frühen Leistungsphasen auf Planerseite und der unternehmensseitigen Kalkulation verdeutlicht.
Auf Planerseite ist die **Kostenermittlung** ein Oberbegriff für Verfahren, mit deren Hilfe die Kosten in Bauprojekten entsprechend dem Stand der Planung und Durchführung festgestellt werden.

Die hierarchische Kostengliederungsstruktur als eine Zusammenfassung der Kostengruppen nach DIN 276 bildet die Basis zur Kostenermittlung nach dieser Norm. Eine Kostengruppe wird nach DIN 276 als die Zusammenfassung einzelner nach Kriterien der Planung oder des Projektablaufs zusammengehörenden Kosten betrachtet.

Der Ausgangspunkt eins Bauprojektes, welches von einem Bauherren veranlasst wird, bilden:
- ein **Kostenvoranschlag**, der aufgrund eines Vorentwurfes erstellt wird.
- ein **Leistungsverzeichnis**, dessen Grundlage die Entwurfspläne der Architektur, der Statik, der technischen Ausbauwerke und eine ergänzende Baubeschreibung sind [Keil88].

Der Kostenvoranschlag eines Gebäudes, der vom Bauherrn als Basis für die Finanzplanung genutzt wird, gilt für das Ingenieurbüro als Orientierung bei dem Prozess der Ausschreibung. Die Kostengruppen nach DIN 276 Teil 2 bilden die Grundlage zur Erstellung des Kostenvoranschlags.

DIN 276 legt folgende Arten von Kostenermittlungen fest:

- **Kostenschätzung**
  Die Kostenschätzung wird vor dem Baubeginn und vor der Annahme von Angeboten von Bauleistungen erstellt und ist als Orientierungsmittel für Finanzierungsüberlegungen gedacht. Bedarfsplanung und Zeit- und Kapazitätsplanung können als Grundlagen der Kostenschätzung betrachtet werden. Bei der Kostenermittlung durch Kostenschätzung werden die einzelnen Kostengruppen bis zur 2. Spalte der Kostengliederung nach DIN 276 Teil 2 beachtet.

- **Kostenberechnung**
  Die Kostenberechnung wird zur Untersuchung der angenäherten Gesamtkosten genutzt. Abhängig von der Kostenberechnung wird entschieden, ob die Baumaßnahmen planmäßig durchgeführt werden sollen. Hier werden die einzelnen Kostengruppen bis zur 3. Spalte der Kostengliederung nach DIN 276 Teil 2 berücksichtigt.

- **Kostenanschlag**
  Mit der Kostenermittlung durch Kostenanschlag werden die tatsächlich zu erwartenden Kosten durch das Zusammenstellen von mehreren Berechnungen festgestellt. Grundlage des Kostenanschlags sind die genaue Bedarfs-, Zeit- und Kapazitätsplanung und die dazu gehörigen Unterlagen. Leistungen werden hier in Leistungspositionen aufgeschlüsselt und in den Kostengruppen in der Aufeinanderfolge des Produktionsvorgangs geordnet. Bei dem Kostenanschlag werden die Kostengruppen bis zur letzten Spalte der Kostengliederung nach DIN 276 Teil 2 berücksichtigt.

- **Kostenfeststellung**
  Ein Nachweis über die entstandenen Kosten wird durch eine Kostenfeststellung durchgeführt. Die notwendigen Unterlagen für eine Kostenfeststellung sind Kostennachweise, Planungsunterlagen und der Fertigstellungsbericht.

Die **Kalkulation** als bedeutender Bestandteil des baubetrieblichen Rechnungswesens dient dazu, die Kosten für Bauleistungen im Bauunternehmen zu ermitteln [Petz95]. Das baubetriebliche Rechnungswesen wird nach REFA [REFA84] im Rahmen der Aufbauorganisation der Bauunternehmung der kaufmännischen Leitung angegliedert; die Nachkalkulation aber wird der technischen Leitung zugeordnet.

Das System des baubetrieblichen Rechnungswesens gliedert sich nach Keil [Keil88] in:
- **Unternehmens- und Finanzrechnung** sowie
- **Kosten- und Leistungsrechnung**.

Kosten- und Leistungsrechnung gliedern sich ihrerseits in:
- **Bauauftragsrechnung** und
- **Baubetriebsrechnung**.

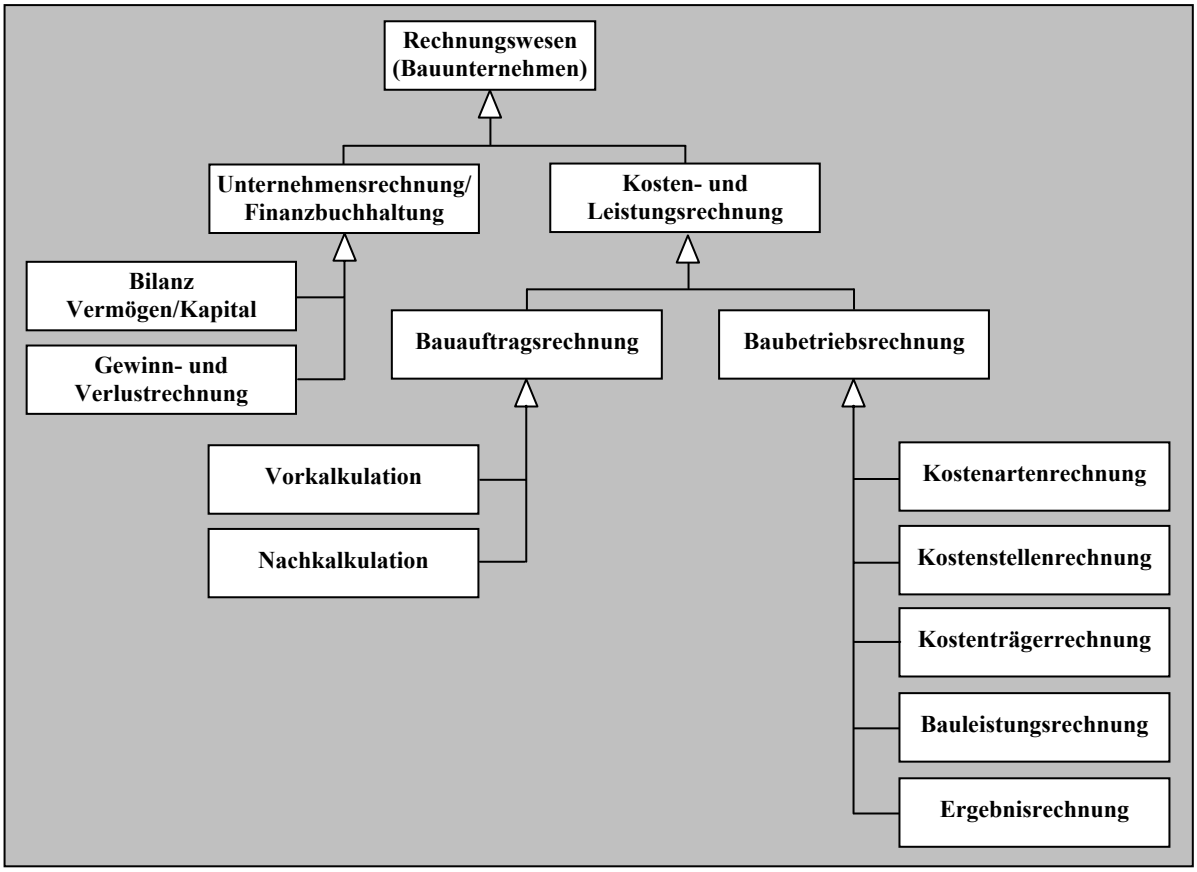


Abb. 4.7: Klassendiagramm des Rechnungswesens einer Bauunternehmung

In Abbildung 4.7 sind die Hauptkomponenten des baubetrieblichen Rechnungswesens anhand eines Klassendiagramms dargestellt.

Die Aufgabe der **Kalkulation** als Auftragsbezogene Kostenermittlung besteht darin, die bei der Realisierung der Bauleistungen entstehenden Kosten der Mengen und Werte der erforderlichen Materialien und Arbeitszeit zu ermitteln. Bei der Ermittlung der Mengen und Werte steht ein **Leistungsverzeichnis** mit Einzelbeschreibungen der Positionen oder eine Baubeschreibung zur Verfügung.

Gemäß dem Zeitpunkt ihrer Erstellung kann die Kalkulationen in Vor- und Nachkalkulation gegliedert werden. Die Vorkalkulation wird vor der Auftragserteilung vorgenommen. Nach der Auftragserteilung erfolgt die Arbeitskalkulation. Die Nachkalkulation wird dann nach dem Bauende vollzogen.

Die **Vorkalkulation** als Kostenermittlung von Bauleistungen gliedert sich nach Prange, Leimböck und Klaus [Pran95] in:

- **Angebotskalkulation** zur Kostenermittlung von Bauleistungen zur Erstellung eines Angebotes. Die Angebotskalkulation wird im weiteren Verlauf der vorliegenden Arbeit näher beschrieben.
- **Auftragskalkulation** zur Kostenermittlung von Bauleistungen während der Auftragsverhandlungen. Durch diese Kalkulation können die sich aus dem Bauvertrag ergebenden Abweichungen gegenüber den Verdingungsunterlagen in ihren Kosten überprüft und mit der Angebotskalkulation verglichen werden, um Auswirkungen auf das kalkulierte Baustellenergebnis festzustellen.
- **Arbeitskalkulation** zur Kostenermittlung von Bauleistungen auf der Grundlage des Bauausführungsplanes. Diese Kalkulation wird mit dem Ziel erstellt, eine maximale Wirtschaftlichkeit bei der Bauausführung zu erreichen.
- **Nachtragskalkulation** zur Kostenermittlung von Bauleistungen, die nicht im Bauvertrag vereinbart wurden sowie solche, für die sich die Basis der Preisermittlung geändert hat.

Die Basis der Vorkalkulation ist die nach Einzelpositionen durchzuführende Baupreiskalkulation. Dafür wird eine Gliederung nach **Kostengruppen** vorgenommen:

- Lohnkosten
- Gerätekosten
- Baustoffkosten
- Nachunternehmerkosten /Fremdleistungen

Die **Nachkalkulation** ist die Ermittlung der bei der Bauausführung tatsächlich entstandenen Kosten. Ergebnisse der Nachkalkulation dienen zur Überprüfung der Ansätze der Vorkalkulation und zur Ermittlung von Richtwerten für die Angebotskalkulation ähnlicher Bauprojekte.

4.2.2 Beschreibung des Systems der Angebotskalkulation

Bei der Angebotskalkulation handelt es sich um ein Verfahren zur Ermittlung der Angebotssumme. Hier wird die Preisbildung, die dem Angebot eines Bieters zugrunde liegt, gezeigt.
Die Erteilung von Bauaufträgen erfolgt normalerweise anhand von Angeboten, die von den Auftragnehmern auf eine Ausschreibung des Bauherrn hin ausgearbeitet werden. Dies bedeutet, dass es äußerst wichtig ist, die Kalkulation für das Angebot gründlich und sorgfältig zu ermitteln.

Bei zu hohen Angebotssummen scheidet die Unternehmung in der Regel aus dem Wettbewerb aus. Wenn der Angebotspreis zu niedrig ist, können die für das Bauunternehmen entstehenden Kosten nicht mehr ersetzt werden. Aus diesem Grund spielt die Angebotskalkulation im Bauunternehmen eine bedeutende Aufgabe.

Die Angebotskalkulation setzt eine gründliche Analyse der Ausschreibungsunterlagen voraus. Als Ausschreibungsunterlagen kommen hier das Leistungsverzeichnis mit den zugehörigen Vertragsbedingungen und Ausschreibungsplänen in betracht. Das Leistungsverzeichnis beinhaltet die einzelnen Teilleistungen eines Bauvorhaben, die nach geordneten Positionen dargestellt sind.

Zur Modellierung der betriebswirtschaftlichen Prozesse innerhalb der Angebotskalkulation ist es notwendig, die Strukturen der Angebotskalkulation in der Bauwirtschaft darzustellen.
Nachfolgend werden die wichtigsten Bestandteile des Systems der Angebotskalkulation beschrieben:

- **Einzelkosten** sind die direkt einzuordnenden Kosten auf ein Produkt oder eine Dienstleistung. Sie werden als Bestandteil der Herstellkosten einer Baumaßnahme betrachtet.

- **Einzelkosten der Teilleistungen (EKT)** sind die ermittelten Kosten der einzelnen Positionen eines LV. Im Rahmen der Ermittlung der Herstellungskosten steht die Ermittlung der Einzelkosten der Teilleistungen an erster Stelle. Kostensummen einer Baumaßnahme, die in einer Kalkulation als EKT abgebildet wurden, lassen sich bei der Kostenartenrechnung verdichtet darstellen.

- **Gemeinkosten der Baustelle (BGK)** sind Kosten, die durch das Betreiben einer Baustelle entstehen, die sich jedoch keiner Teilleistung direkt zuordnen lassen. Diese Kosten können in zwei Kostenarten aufgeteilt werden, nämlich die **zeitabhängigen** wie z. B. Bauleitungskosten und die **zeitunabhängigen** Kosten wie z. B. die Transportkosten der Baustelleneinrichtung.

- **Herstellkosten** sind die auf das herzustellende Bauobjekt zuzurechnenden Kosten. Sie enthalten die Summe der Einzelkosten der Teilleistungen und die Summe der Gemeinkosten der Baustelle.

- **Allgemeine Geschäftskosten (AGK)** sind Kosten, die dem Unternehmen durch den Betrieb als Ganzes entstehen. Hierzu zählen insbesondere sämtliche Kosten der Unternehmensverwaltung wie z. B. Gehälter und Miete.

- **Selbstkosten** sind die auf das herzustellende Produkt zuzurechnenden tatsächlichen Kosten. Sie enthalten die Summe der Herstellkosten und die Summe der Allgemeinen Geschäftskosten.

- **Wagnis** und **Gewinn** bilden die Differenz zwischen den Selbstkosten und der Angebotssumme. Hiermit sollen Unternehmerwagnisse wie Konjunkturrisiken und unvorhergesehene Kosten abgedeckt werden.

Innerhalb der Angebotskalkulation werden die Herstellkosten als Summe der Einzelkosten der Teilleistungen und der Baustellengemeinkosten ermittelt.
Die Angebotssumme ergibt sich aus der Summe der Herstellkosten zuzüglich einem Anteil an den allgemeinen Geschäftskosten und einem Anteil für Wagnis und Gewinn als Prozentsatz der Endsumme.

Zur Vereinfachung der Übernahme von Aufwandswerten aus der KLR Bau, ist es empfehlenswert, bei der Kalkulation dieselbe Kostenartengliederung zu verwenden, die bei KLR Bau eingesetzt wird. Unter dieser Kostenartengliederung sind Lohnkosten, Stoffkosten, Rüst- und Schalungskosten, Gerätekosten, Baustellen-, Betriebs- und Geschäftskosten, Fremdleistungen und Sonstige Kosten zu berücksichtigen.
Die Kostenartengliederung richtet sich vor allem nach der Art des Bauunternehmens und seiner Schwerpunkte. Während beim Hochbau die Lohnkosten relativ größer sind, bilden im Straßenbau die Geräte- und Materialkosten den Hauptanteil.

Eine realistische Erfassung der Kosten sowie deren eindeutige Zuordnung zu bestimmten Bereichen sind die wichtigsten Anforderungen an ein effizientes Kalkulationsverfahren. In diesem Zusammenhang ist zu berücksichtigen, dass unterschiedliche Kalkulationsverfahren bei der Kalkulation von Angeboten eingesetzt werden:

- **Kalkulation über die Angebotssumme**
  Das Verfahren der Kalkulation über die Angebotssumme baut auf die Kalkulation der Einzelkosten der Teilleistungen zuzüglich der Summe der Gemeinkosten der Baustelle sowie einem Zuschlag für allgemeine Geschäftskosten, Wagnis und Gewinn.

- **Kalkulation mit vorberechneten Zuschlägen**
  Dieses Verfahren basiert auf die Kalkulation der Einzelkosten der Teilleistungen zuzüglich eines vorgegebenen Zuschlags für die Gemeinkosten der Baustelle sowie einem Zuschlag für allgemeine Geschäftskosten, Wagnis und Gewinn.
  Bei diesem Verfahren werden die Einzelkosten der Teilleistungen, wie bei der Kalkulation über die Angebotssumme berechnet. Aus den Werten der Betriebsbuchhaltung wird der Zuschlag für die Gemeinkosten der Baustelle und der Verwaltung ermittelt und im Laufe der Zeit aktualisiert.
  Dieses Verfahren wird zumeist bei kleineren Bauunternehmen oder Handwerksfirmen angewendet, die für ihre Bauaufträge mit gleichen oder ähnlichen BGK und allgemeinen Geschäftskosten rechnen können. Bei größeren Bauvorhaben ist dieses Verfahren nicht zweckmäßig.

5 Objektorientierte Analyse des Systems der Kosten in Bauprojekten

5.1 Vorgehensweise bei Modellierung des Systems

Als Methodik zur Modellierung Kosten in Bauprojekten des Hochbaus wird die *Unified modeling language* (UML) Version 1.3 eingesetzt. In UML wird die Architektur von Softwaresystemen in Form von fünf überlappenden Sichten dargestellt (Abb. 5.1).

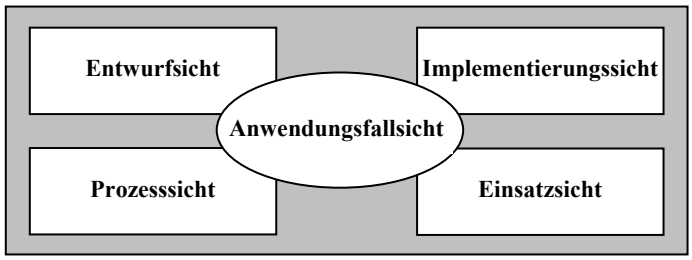


Abb. 5.1: Modellieren einer Systemarchitektur (nach [Booc99])

Diese Struktur dient vor allem der Visualisierung, Spezifikation und Konstruktion von Softwaresystemen. Es werden dabei Aspekte zur Festlegung der Organisation, Zusammensetzung und Verhalten von Softwaresystemen sowie das Zusammenfügen von unterschiedlichen Elementen in diesen Systemen betrachtet.

Die erste der o. g. Sichten ist die Anwendungsfallsicht *(use case view)*. Die statischen Gesichtspunkte dieser Sicht werden in Form von Anwendungsfalldiagrammen und die dynamischen Gesichtspunkte anhand von Interaktions-, Zustands- und Aktivitätsdiagrammen beschrieben. Die Basis dieser Diagramme bilden die identifizierten Anwendungsfälle *(use cases)*.

In der Entwurfssicht *(design view)* werden Klassen, Schnittstellen und Kollaborationen eines Systems beschrieben. Die statischen Gesichtspunkte dieser Sicht werden in Form von Klassen- und Objektdiagrammen und die dynamischen Gesichtspunkte anhand von Interaktions-, Zustands- und Aktivitätsdiagrammen dargestellt.

In der Prozesssicht *(process view)* eines Systems werden die Nebenläufigkeit und Synchronisierung anhand von Threads und Prozessen beschrieben. Zur Darstellung der statischen und dynamischen Gesichtspunkte dieser Sicht dienen die Diagramme der Entwurfssicht, wobei hier die aktiven Klassen den Kern dieser Darstellung bilden.

In der Implementierungssicht *(implementation view)* eines Systems werden die zur Zusammenstellung und Freigabe des physischen Systems notwendigen Komponenten und Dateien ausgearbeitet.
Die statischen Gesichtspunkte dieser Sicht werden in Form von Komponentendiagrammen und die dynamischen Gesichtspunkte anhand von Interaktions-, Zustands- und Aktivitätsdiagrammen dargestellt.

In der Einsatzsicht *(deployment view)* eines Systems wird die Hardwaretopologie des Systems anhand von Knoten dargestellt. Hier steht die Verteilung der Teile eines physischen Systems im Mittelpunkt. Die statischen Gesichtspunkte dieser Sicht werden in Form von Einsatzdiagrammen und die dynamischen Gesichtspunkte anhand von Interaktions-, Zustands- und Aktivitätsdiagrammen dargestellt.

Um die Anschaulichkeit des Systemmodells zu erhöhen wird in der UML das Konzept Pakete angewendet, um Elemente des Systems zu Gruppen zusammenzufassen. Die Regelung der Sichtbarkeit der Elemente des Systems wird in der UML beschrieben. Diese Regelung ist mit der in Java angewendeten Regelung der Sichtbarkeit vereinbar.

In der UML sind die folgenden zwei Spezialisierungen von Paketen dargestellt:

- **Subsysteme** zur Spezifizierung des Inhaltes von Paketen. Hier werden Attribute, Operationen und Schnittstellen spezifiziert. Subsysteme werden wie Pakete gezeigt, wobei das Schlüsselwort <<*subsystem*>> über dem Namen des Subsystems notiert wird.
- **Modelle** die ebenso wie Pakete gezeigt werden, wobei das Schlüsselwort <<*model*>> über dem Namen des Modells notiert wird.

In Abbildung 5.2 wird die Vorgehensweise zur Modellierung des Systems der Kosten in Bauprojekten dargestellt. Demgemäß kommen folgende Modelle in Betracht:

- Anforderungsmodell
- Analysemodell
- Entwurfsmodell
- Implementierungsmodell

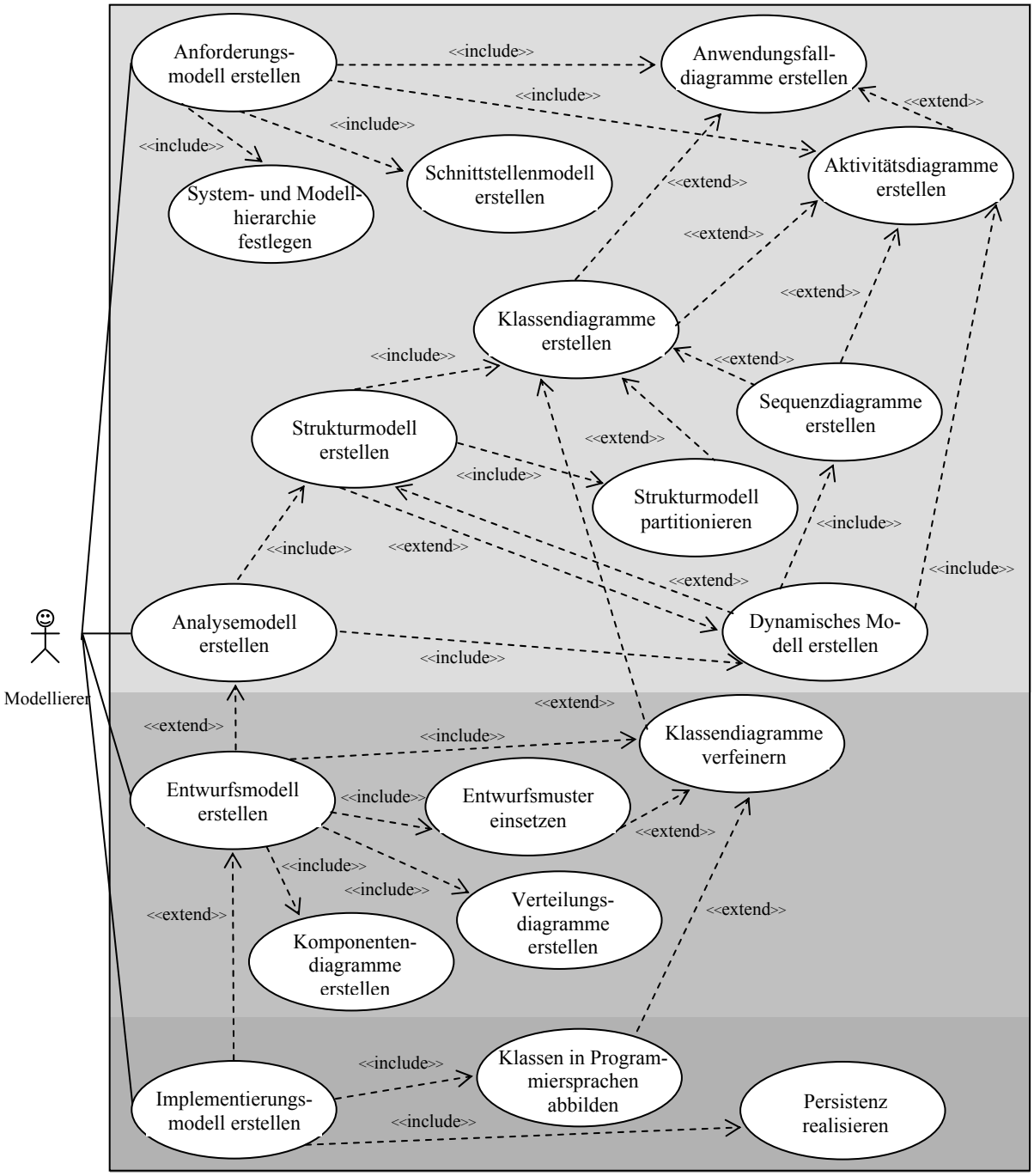


Abb. 5.2: Vorgehensweise zur Modellierung des Systems der Kosten in Bauprojekten mit UML

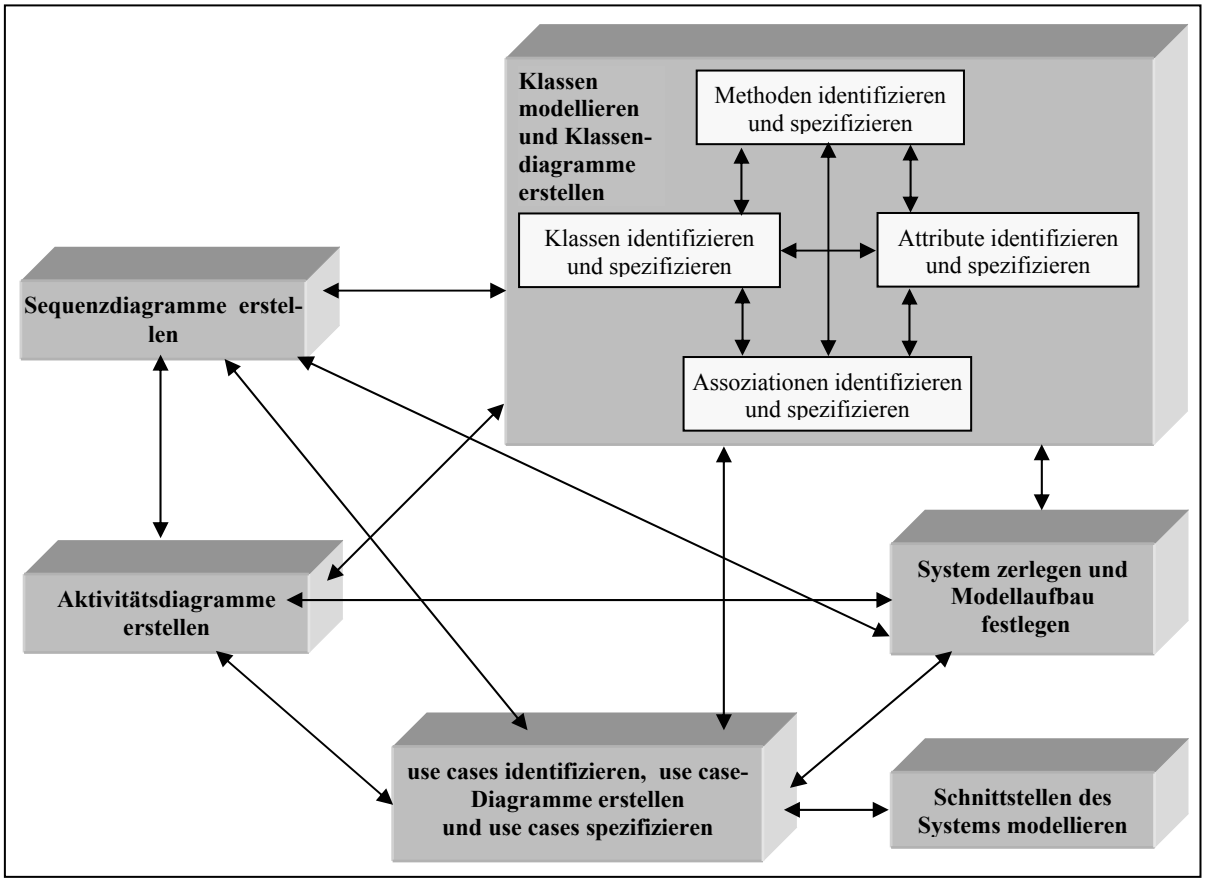


Abb. 5.3: Vorgehensweise zur Modellierung des Systems mit UML in der Anforderungs- und Analysephase

In Abbildung 5.3 ist die Vorgehensweise zur Modellierung des Systems in der Anforderungs- und Analysephase dargestellt, wobei hier das Zusammenspiel zwischen den Komponenten dieser Teilmodelle detailiert beschrieben ist. Es werden hier nur die UML-Modellierungskonzepte eingesetzt die zweckmäßig gebraucht werden.

5.2 Das Anforderungsmodell

Bei der Entwicklung von Softwarelösungen spielt eine bewährte Anforderungsanalyse eine wichtige Rolle. Der Aufwand zur Entwicklung einer solchen Analyse ist viel günstiger als die Einbuße, die durch eine mangelhafte Anforderungsanalyse entsteht.

Die UML stellt einige Konzepte für Anforderungsanalysen zur Verfügung. Als Basis wird die Anwendungsfallmodellierung betrachtet. Darüber hinaus gibt es viele Anforderungen, die durch UML-Konzepte nicht entsprechend erfasst werden können. Damit sind hier die nicht-funktionalen Anforderungen und die Randbedingungen für Entwicklungsprojekte gemeint. Bei der Anforderungsanalyse im Hochbau gliedern sich die Anforderungen folgendermaßen:

- **Funktionale Anforderungen** in welchen die gewünschten funktionalen Leistungen, Abläufen, Algorithmen und Daten des zu modellierenden Systems beschrieben werden.
- **Nicht-funktionale Anforderungen** in welchen Handhabungseigenschaften, die von dem zu modellierenden Systems gewährleistet werden müssen, wie z. B. Performanz und Benutzbarkeit, beschrieben.
- **Produkt- und Systemrandbestimmungen** legen Bedingungen und Einschränkungen fest, die für das Produkt bzw. für das ganze System gelten. In diese Kategorie gehören die Definition**,** der Zweck des Produktes, die Festlegung der Nutzer und der Zweck des zu modellierenden Systems.

Im Anforderungsmodell wird das System der Kosten im Hochbau durch Anwendungsfall- und Aktivitätsdiagramme beschrieben. Die Beschreibung durch diese UML-Konzepte decken teilweise die funktionalen Anforderungen. Anwendungsfalldiagramme beschreiben auch die Systemabgrenzung. Hier werden der Inhalt des Systems der Kosten in Bauprojekten und seine Systemumgebung aufgezeichnet. Darüber hinaus werden die Schnittstellen des Systems modelliert. Hier werden die Kommunikationswege zwischen dem System und seiner Umgebung beschrieben.

Zu funktionalen Anforderungen gehören auch die Spezifikationen der Klassen, Attribute und Assoziationen des zu modellierenden Systems, sowie deren Darstellung anhand von UML-Klassendiagrammen. Es soll auch nicht ausgeschlossen werden, dass im Laufe der Modellierung des Systems neue Anforderungen seitens des Auftraggebers kommen. Dieser Aspekt soll durch den Einsatz von bestimmten Konzepten der Entwurfsmuster berücksichtigt werden.

5.2.1 Beschreibung des Systems der Baukalkulation

Bei der Gliederung der Baukalkulation unterscheidet man zwischen Vorkalkulation und Nachkalkulation [Pran95]. Die Vorkalkulation, die sich in Angebots-, Auftrags-, Arbeits- und Nachtragskalkulation gliedert, deckt das gesamte Spektrum von der Angebotserstellung durch die Auftragskalkulation bis hin zur Nachtragskalkulation einschließlich der Arbeitskalkulation, die nach der Erteilung des Bauauftrags ausgeführt wird (Abb. 5.4).

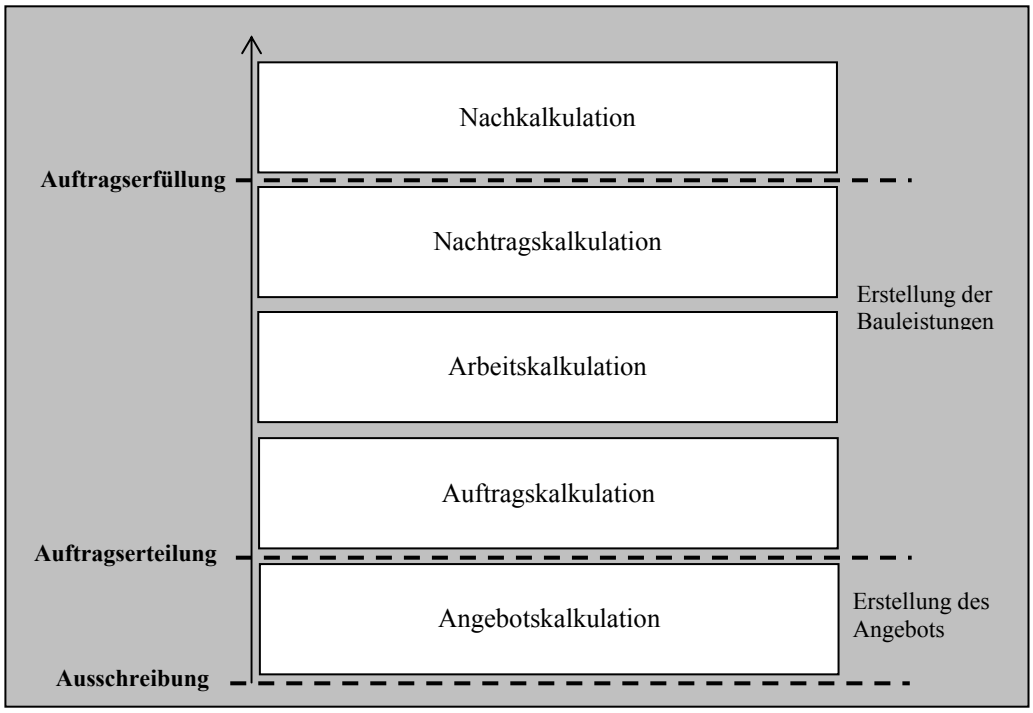


Abb. 5.4: Detaillierte Darstellung der Strukturierung der Baukalkulation

Die Struktur der Baukalkulation wird in Abbildung 5.5 anhand eines Klassendiagramms dargestellt. Hier wird die Klasse Kalkulation als Generalisierung der Klassen Vorkalkulation und Nachkalkulation betrachtet. Die Klasse Vorkalkulation wird ihrerseits als Generalisierung der Klassen Angebotskalkulation, Auftragskalkulation, Arbeitskalkulation und Nachtragskalkulation spezifiziert. Innerhalb dieser Arbeit werden hauptsächlich nur die Strukturen der Angebotskalkulation modelliert.

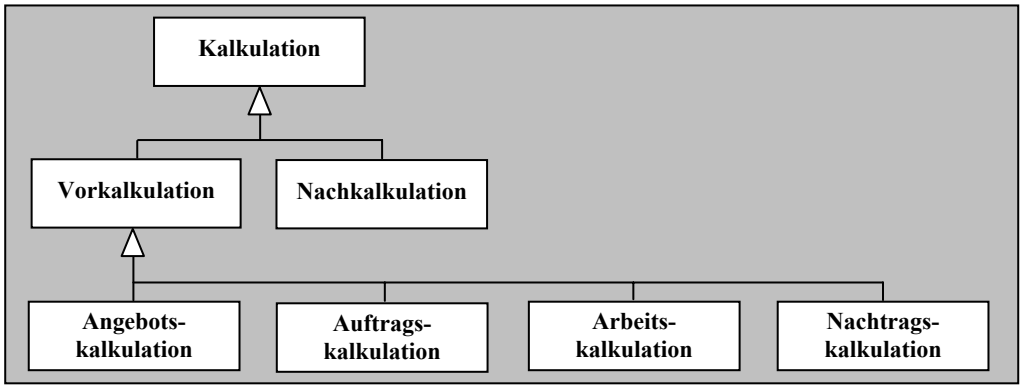


Abb. 5.5: Klassendiagramm der Struktur der Kalkulation

5.2.2 Anwendungsfalldiagramme

Das zu modellierende System wird hier anhand von Anwendungsfalldiagrammen *(use case diagrams)* bezüglich seiner Funktionalität beschrieben. Bei diesem System handelt es sich um eine ganzheitliche Betrachtung des Bauprojektes, also einschließlich der Aktivitäten von Bauherren und Architekten. Anwendungsfalldiagramme zeigen die Beziehungen zwischen Akteuren und Anwendungsfällen.

Um Übersichtlichkeit zu gewährleisten, können Anwendungsfalldiagramme in Pakete gekapselt werden. Ein Anwendungsfall beschreibt eine Menge konsistenter und zielgerichteter Interaktionen von Akteuren mit einem System, an deren Ende ein definiertes Ergebnis entstanden ist. Diese Beschreibung ist ähnlich der Beschreibung von Systemen durch Prozessketten bei der Produktmodellierung mit STEP-Technologie.

Zuerst werden die mit dem System interagierenden Akteure identifiziert. Diese können sowohl menschliche Akteure als auch externe Systeme sein. Es wird zwischen primären und sekundären Akteuren unterschieden. Primäre Akteure sind Hauptanwender des Systems und sind in der Lage, Anwendungsfälle zu aktivieren. Sekundäre Akteure werden lediglich eingesetzt, um eine einwandfreie Funktionalität des System zu gewährleisten. Akteure können auch gemäß ihrer Rolle als aktive oder passive klassifiziert werden, wobei primäre Akteure immer als aktive betrachtet werden. Sekundäre Akteure allerdings können entweder aktive oder passive sein.

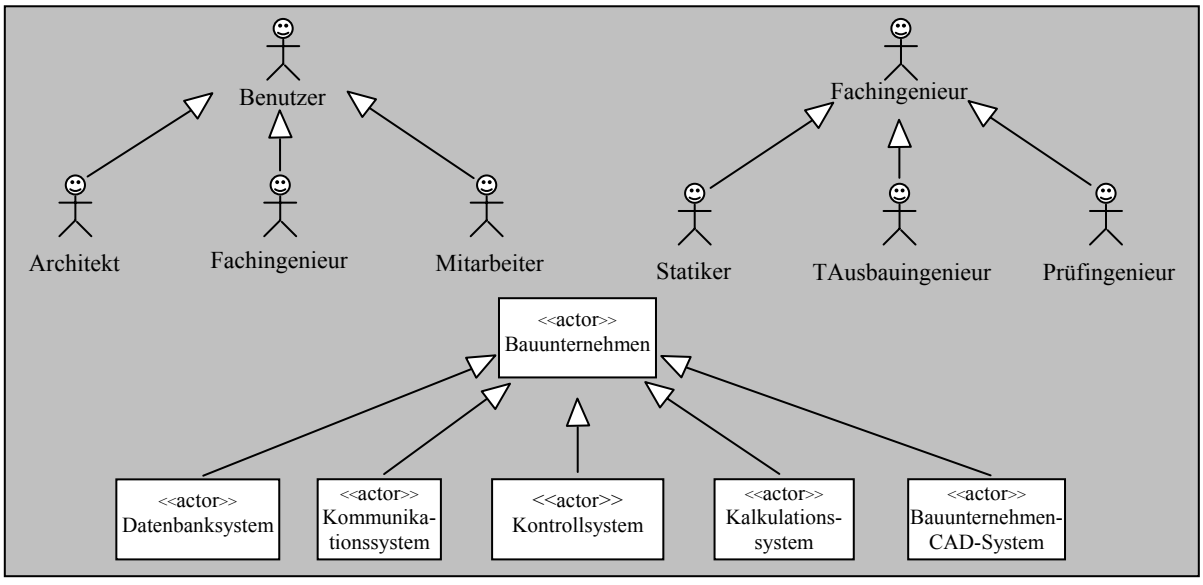


Abb. 5.6: Generalisierungsbeziehungen der Akteure des Systems der Baukalkulation

Als Akteure im System der Baukalkulation werden der Bauherr, der Architekt, der Fachingenieur und die externen Systeme Bauunternehmen und CAD-System identifiziert.
In Abbildung 5.6 werden Generalisierungs- und Spezialisierungsbeziehungen dieser Akteure dargestellt.

Bei der Identifizierung der Anwendungsfälle des Systems der Baukalkulation werden die Aktivitäten innerhalb dieses Systems analysiert.

Die erste Aktivität des Bauunternehmens im System der Baukalkulation beginnt mit dem Anwendungsfall Angebotskalkulation erstellen. Dieser Anwendungsfall ist eine Erweiterung der Anwendungsfälle Bauvorhaben ausschreiben und Ausschreibungsunterlagen erstellen.

Der Anwendungsfall Auftragskalkulation erstellen wird als Erweiterung der Anwendungsfälle Auftragsverhandlungen durchführen, Änderungen berücksichtigen und Angebotskalkulation erstellen betrachtet.

Der Anwendungsfall Arbeitskalkulation erstellen ist eine Erweiterung der Anwendungsfälle Arbeitsvorbereitungen durchführen und Auftragskalkulation erstellen. Der Anwendungsfall Arbeitsvorbereitungen durchführen, als Erweiterung des Anwendungsfalls Auftrag bestätigen, beinhaltet die Anwendungsfälle Ausführungsmethoden berücksichtigen, Herstellverfahren berücksichtigen, Baustellenverhältnisse berücksichtigen, Geräteleistungen berücksichtigen und Leistungs- und Akkordlöhne berücksichtigen.
Der Anwendungsfall Nachtragskalkulation erstellen, als Erweiterung der Anwendungsfälle Bau durchführen und Grundlagen der Preisermittlung ändern, bildet die Basis für den Anwendungsfall Zwischenkalkulation erstellen.
Der Anwendungsfall Zwischenkalkulation erstellen wird als Erweiterung der Anwendungsfälle Arbeitskalkulation erstellen und Nachtragskalkulation erstellen betrachtet.

Der Anwendungsfall Istwerte der Betriebsabrechnung berücksichtigen, als eine Erweiterung des Anwendungsfalls Abrechnung erstellen wobei dieser Anwendungsfall seinerseits als eine Erweiterung des Anwendungsfalls Bau durchführen betrachtet wird.

Schließlich wird der Anwendungsfall Nachkalkulation erstellen als eine Erweiterung der Anwendungsfälle Nachtragskalkulation erstellen, Zwischenkalkulation erstellen und Istwerte der Betriebsabrechnung berücksichtigen.

Um die Übersichtlichkeit des Anwendungsfalldiagramms zu erhöhen, kann bei komplexen Problemstellungen das UML-Abstraktionsmechanismus Paket eingesetzt werden. In diesem Sinne werden solche Anwendungsfälle des Systems, die einen gemeinschaftlichen Zielgedanken haben, in einem eigenen Paket zusammangefasst .
In Abbildung 5.7 ist das Anwendungsfalldiagramm der Baukalkulation im Hochbau dargestellt.

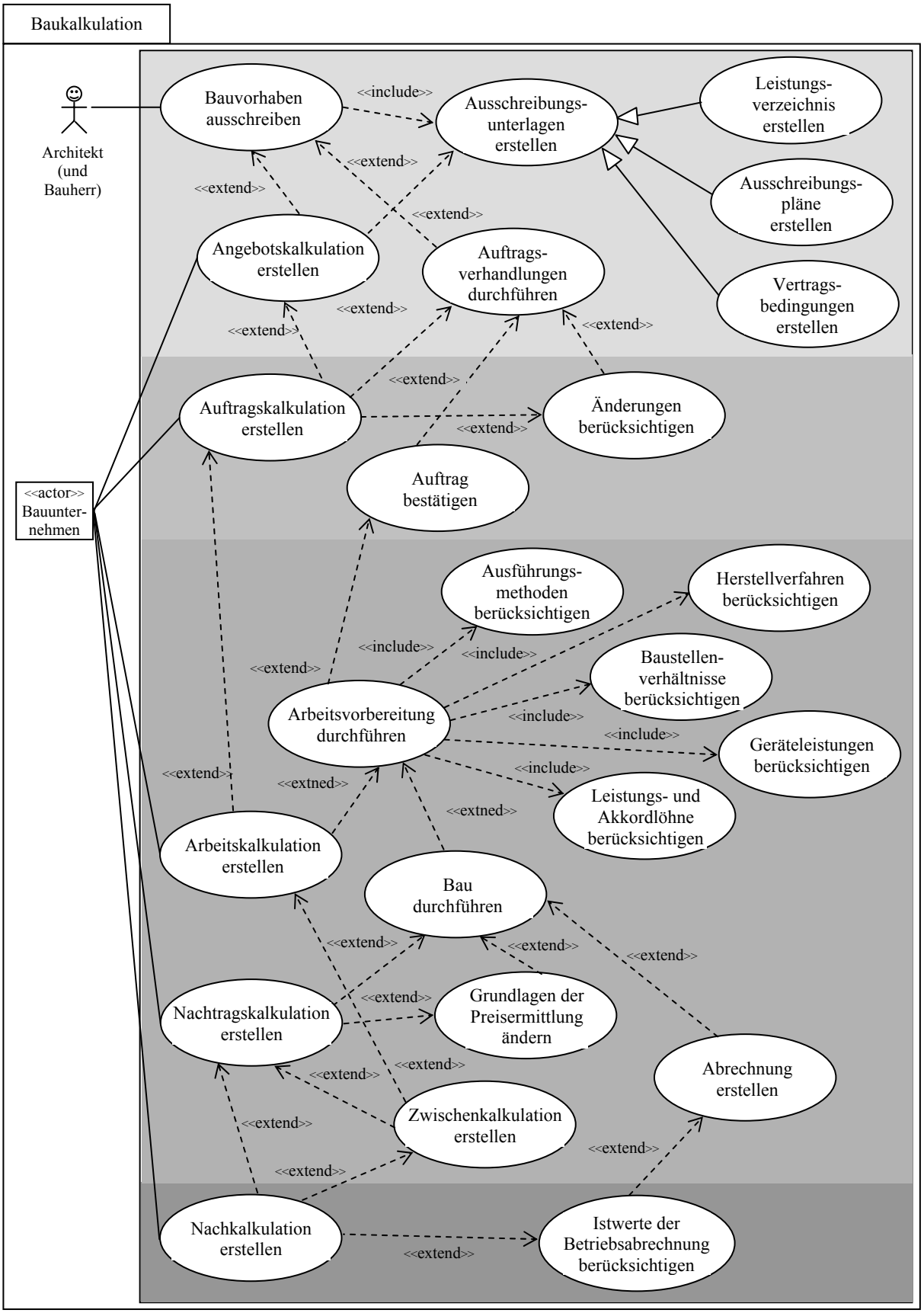


Abb. 5.7 Anwendungsfalldiagramm der Baukalkulation

5.2.3 Festlegung der System- und Modellhierarchie

Sowohl Systeme als auch Teilsysteme können mit UML als Ganzes modelliert werden [Booc99]. Bei der Objektorientierten Analyse der Kosten in Bauprojekten wird das zu modellierende System in Subsysteme nach UML zerlegt.

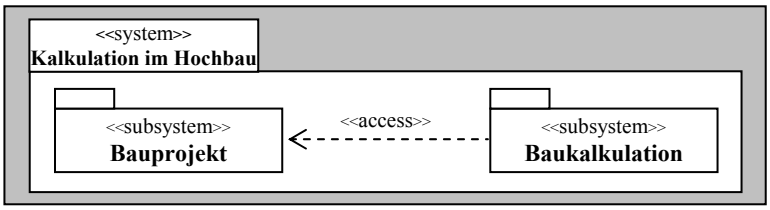


Abb. 5.8: <<system>>-Paket Kalkulation im Hochbau

Dieses System kann als Paket mit dem vordefinierten Stereotyp <<system>> und den eingebetteten Paketen Bauprojekt und Baukalkulation modelliert werden, wobei diese ihrerseits als Subsysteme betrachtet werden (Abb. 5.8).

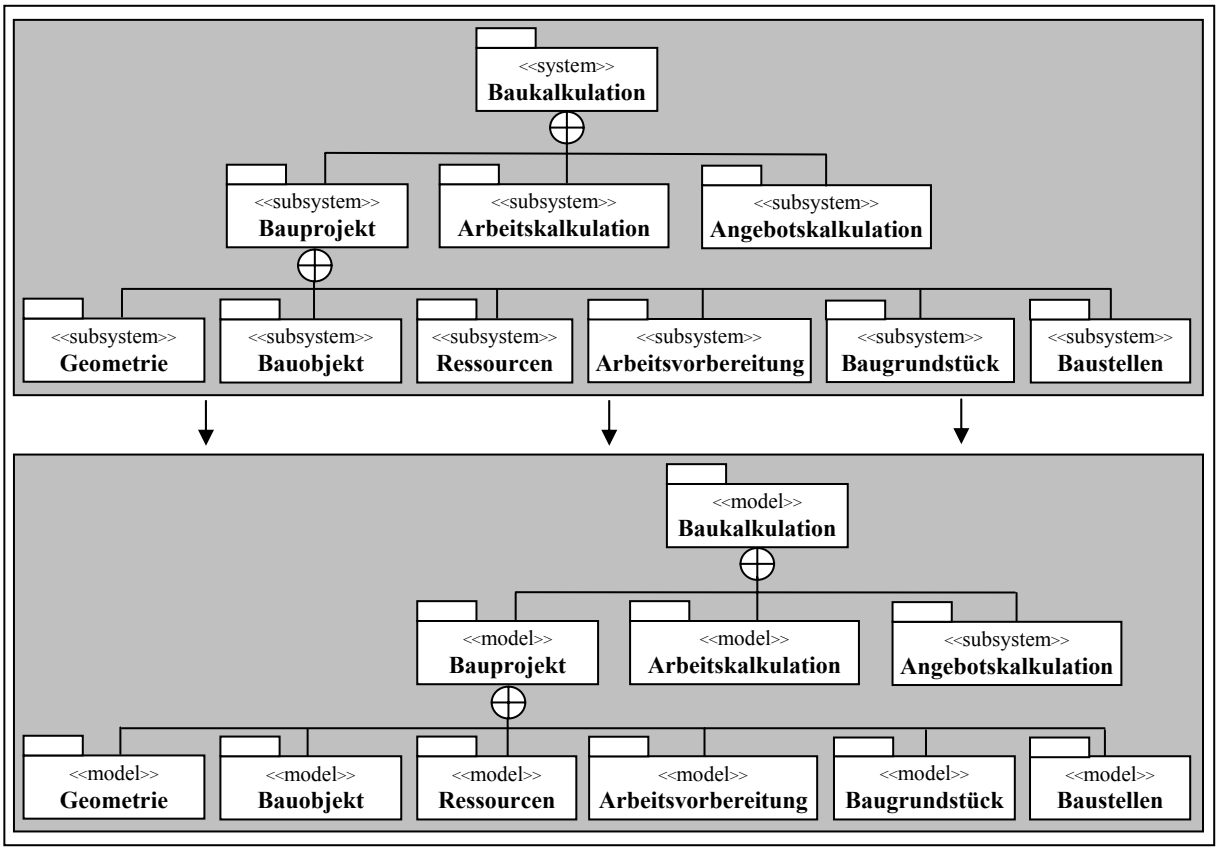


Abb. 5.9: Die Symmetrie zwischen Systemhierarchie und Modellhierarchie der Baukalkulation

Die weitere Modellierung in der vorliegenden Arbeit bezieht sich nur auf das Subsystem Baukalkulation, wobei dies als System betrachtet wird. Das System der Baukalkulation besteht aus den Subsystemen Angebotskalkulation, Arbeitskalkulation und Bauprojekt. Das Subsystem Bauprojekt besteht seinerseits aus den Subsystemen Arbeitsvorbereitung, Baugrundstück, Bauobjekt, Baustellen, Geometrie und Ressourcen.

Bei der Festlegung der Modellhierarchie des Systems der Baukalkulation ist von einer Symmetrie zwischen der Systemhierarchie und der Modellhierarchie auszugehen (Abb. 5.9). So können die Subsysteme modelliert werden, und danach zu einem gesamten Modell des Systems der Baukalkulation integriert werden.

5.2.4 Modellierung der Schnittstellen des Systems

Das Konzept der Schnittstellen spielt eine wichtige Rolle besonders bei der Modellierung von großen Systemen. Hier werden die Schnittstellen nicht nur zur Trennung der Spezifikation und Implementierung von Klassen oder Komponenten eingesetzt, sondern auch um die Außensicht von Paketen oder Subsystemen zu spezifizieren.
So werden Schnittstellen in der UML auch eingesetzt, um die Umgebung eines Systems zu modellieren. Damit spezifiziert die Schnittstelle eine Vereinbarung, nach der die Benutzer sich unabhängig voneinander ändern können, solange sie nur den Verpflichtungen dieser Vereinbarung nachkommen.

Bei der Modellierung der Schnittstellen des Systems der Kosten in Bauprojekten des Hochbaus muß für jeden Akteur und jeden Anwendungsfall in den er einschließt geringstenfalls eine zweckmäßige Schnittstelle definiert werden.

Die Benutzerschnittstellen des Systems der Kosten in Bauprojekten können, neben ihrer Hauptaufgabe als Schnittstelle zum Benutzer des Systems, eine beträchtlichere Klarheit über die einzelnen Anwendungsfälle geben. Die Benutzerschnittstellen werden anhand von konkreten Betrachtungen der Anwendungsfälle durch die Akteure des Systems der Kosten in Bauprojekten modelliert. Hierzu werden die Anwendungsfälle konkretisiert und ihre Beziehungen zu einander spezifiziert.

Bei der Modellierung der Systemschnittstellen werden Vereinbarungen getroffen wie externe Systeme mit dem System der Kosten in Bauprojekten des Hochbaus kommunizieren. Als Akteure kommen hier u. a. die CAD-Systeme in Betracht. Zum Einsatz diese Systeme werden Systemschnittstellenbeschreibungen bereitgestellt. Dabei wird festgelegt, ob produktspezifische Schnittstellen verwendet werden sollen sowie die Syntax und Semantik der Daten des Systems.

5.2.5 Betrachtungen zur Modellierung der zeitabhängigen Aspekten

Bei der Modellierung der zeitabhängigen Aspekte eines Systems wird erst untersucht, in wie weit der Zeitaspekt in diesem System von Bedeutung ist. Dabei werden insbesondere die Antwortzeiten auf bestimmten Aufgaben analysiert.
Falls die Nichteinhaltung von Zeitvorgaben durch das System zu gravierenden Fehlern führt, wie es z. B. bei den ABS-Systemen in der Autoindustrie der Fall ist, werden die Zeitanforderungen als hart betrachtet. Bei Systemen, bei denen es ausreichend ist, dass Zeiten im wesentlichen eingehalten und manchmal auch geringfügig überschritten werden dürfen, wie dies z. B. im System der baubetrieblichen Prozesse der Fall ist, wird die Zeitanforderung als weich bezeichnet.

Facility Management als umfassendes Konzept beinhaltet sowohl das Projektmanagement als auch das Nutzungsmanagement. Dieser Begriff umfasst die folgenden drei Aspekte:

- Raumaspekt in dem die Facilities eingeführt werden. Dieser Aspekt kann durch Gebäude- und Geländemodelle dargestellt werden.
- Zeitaspekt in dem der gesamte Bauwerkslebenszyklus betrachtet wird.
- Nutzeraspekt wobei der Nutzer sich während der Lebenszyklusphasen eines Bauwerkes in einem Raum-Zeit-System fortbewegt.

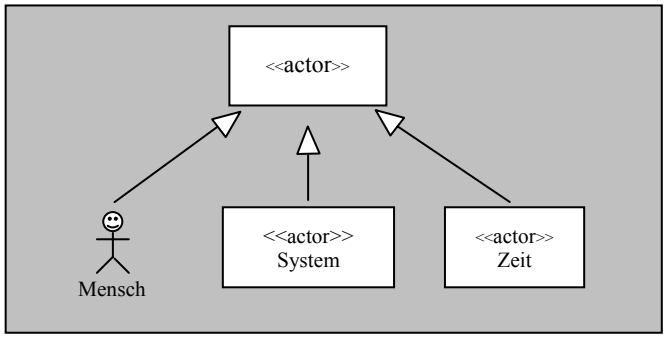


Abb. 5.10: Darstellung des Zeitaspektes als Akteur

Die Betrachtungen der zeitabhängigen Aspekte sind insbesondere bei der Modellierung von Echtzeitsystemen von großer Bedeutung. Bei diesen Systemen werden die Prozesse häufig an definierten Zeitpunkten ausgelöst. Diese Reaktion des Systems ist vordefiniert und vorgeplant. Bei der Modellierung solcher Systeme wird die Zeit als Akteur betrachtet (Abb. 5.10).

Die zeitabhängigen Aspekte können bei der Modellierung des Systems der baubetrieblichen Prozesse durch Interaktionsdiagramme beschrieben werden. Insbesondere eignen sich hier die Aktivitätsdiagramme.

5.3 Das Analysemodell

5.3.1 Das Strukturmodell

Die Erstellung des Strukturmodells bezieht sich auf die Modellierung der Klassen der Subsysteme und deren Darstellung in Klassendiagrammen. Bei der Klassenmodellierung in UML unterscheidet man zwischen Klassendiagrammen und Objektdiagrammen. Klassendiagramme sind Diagramme zur Beschreibung einer Menge von Klassen, Schnittstellen, Kollaborationen und ihren Beziehungen.

Objektdiagramme sind Diagramme zur Beschreibung einer Menge von Objekten als Instanzen von Klassen zu einem bestimmten Zeitpunkt. Das Strukturmodell wird durch Klassendiagramme dargestellt. Um Klassen optimal zu modellieren, werden ihre Verantwortlichkeiten und ihre Beziehungen zueinander definiert. Mit Verantwortlichkeiten wird gemeint, welche Aufgaben die Klassen innerhalb eines Anwendungsfalls haben.

5.3.1.1 Klassenmodellierung des Subsystems Bauprojekt

Als Bauprojekt wird hier ein komplexes Bauvorhaben betrachtet. An der Planung und Steuerung dieses Vorhaben sind normalerweise mehrere Bereiche eines Unternehmens oder mehrere Unternehmen beteiligt.

Das Subsystem Bauprojekt spezifiziert die verschiedenen Aspekte, die bei der Erstellung eines Bauwerkes betrachtet werden sollen. Bestandteile dieses Subsystems sind das Baugrundstück, die Ressourcen, die Arbeitsvorbereitung, das Bauobjekt und die Geometrie.

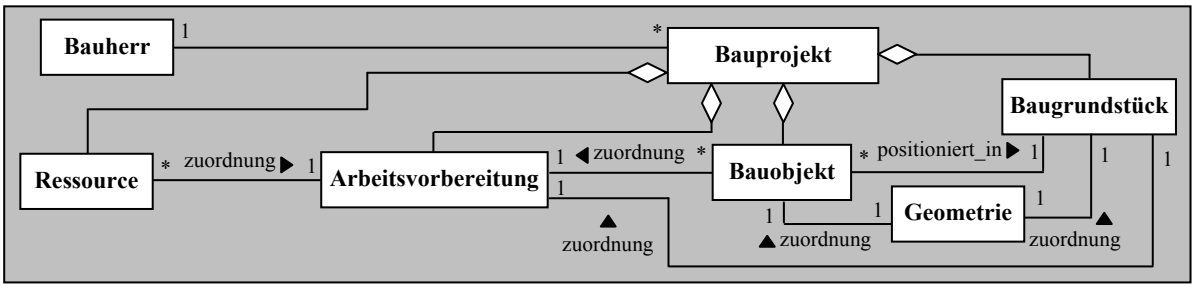


Abb. 5.11: Das Klassendiagramm des Bauprojektmodells

Das Klassendiagramm des Bauprojektmodells im Hochbau ist in Abbildung 5.11 dargestellt. Hier wird die Klasse Bauprojekt als Aggregation der Klassen Arbeitsvorbereitung, Baugrundstück, Bauobjekt, und Ressource betrachtet. Die Klasse Geometrie steht in einer geordneten Assoziation mit den Klassen Baugrundstück und Bauobjekt.

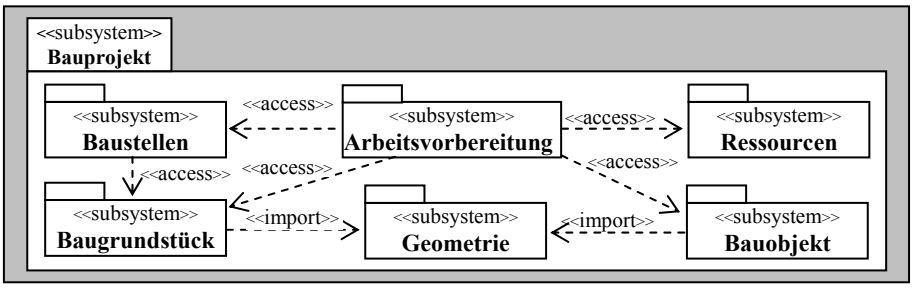


Abb. 5.12: Darstellung des Bauprojektmodells anhand von Paketen

In Abbildung 5.12 ist das Subsystem Bauprojekt weiter in mehrere Subsysteme untergegliedert und durch Pakete dargestellt. Diese Subsysteme, die eine wichtige Rolle bei der Modellierung des Problembereiches spielen, werden nachfolgend spezifiziert.

■ Das Ressourcenmodell

Das Ressourcenmodell dient vorwiegend der Darstellung der Struktur der notwendigen Ressourcen in Bauprojekten des Hochbaus sowie ihrer Zuordnung zur Arbeitsvorbereitung.

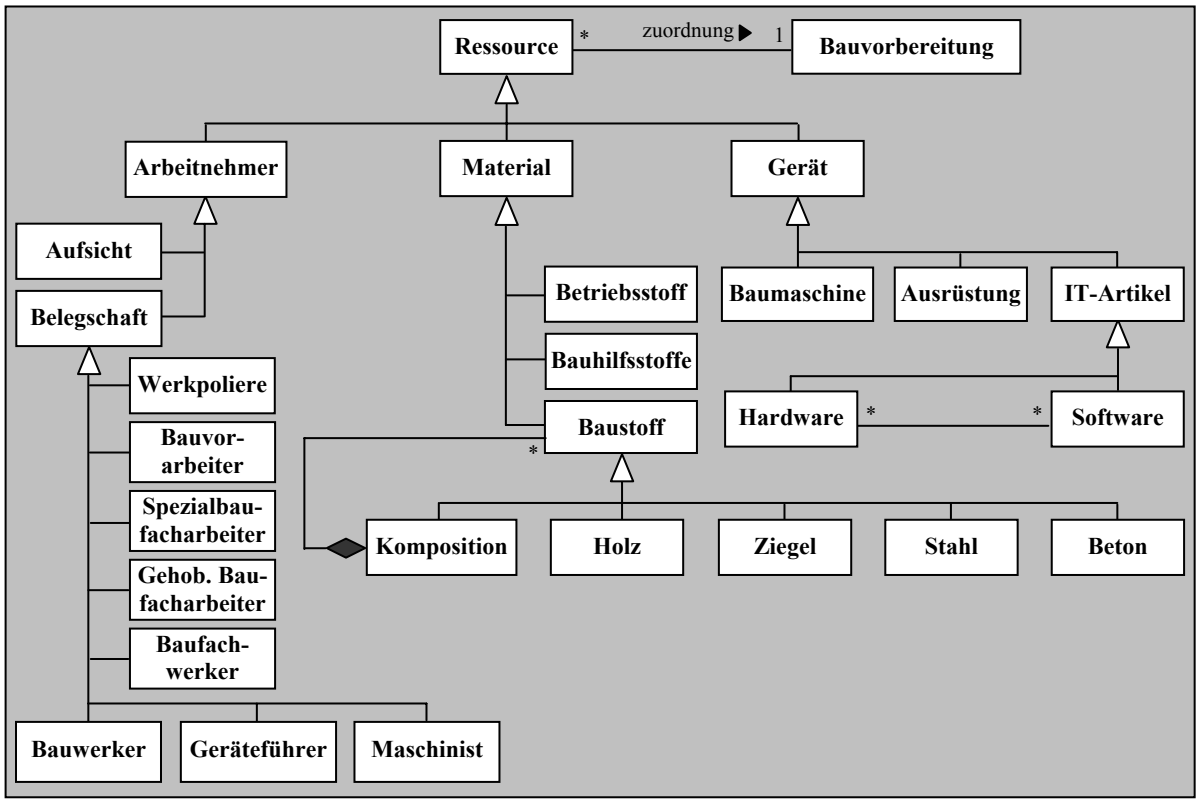


Abb. 5.13: Das Klassendiagramm des Ressourcenmodells

Die Ressourcenplanung hat zur Aufgabe, den Bedarf der verschiedenen Arbeitsmittel, Arbeitskräfte und Arbeitsgegenstände hinsichtlich ihres zeitlichen Einsatzes festzulegen. Im Ergebnis entstehen:

- Maschineneinsatzpläne
- Arbeitskräftepläne
- Materialbedarfspläne.

In Abbildung 5.13 ist das Klassendiagramm des Ressourcenmodells dargestellt. In diesem Diagramm wird die Klasse Ressource als Generalisierung der Klassen Arbeiter, Gerät und Material betrachtet. Die Klasse Ressource steht in einer geordneten Assoziation mit der Klasse Bauvorbereitung. Die Klasse Material wird als Generalisierung der Klassen Baustoff, Energie und Betriebsstoff modelliert. Die Klasse Gerät wird als Generalisierung der Klassen Baumaschine, Ausrüstung und IT-Artikel identifiziert, wobei die Klassen Hardware und Software als Spezialisierung des Objektes IT-Artikel identifiziert werden.

■ Das Bauobjektsmodell

Als Bauobjekt wird hier ein Begriff für Gebäude, sonstige Bauwerke und raumbildende Ausbauten angewendet. Im Rahmen dieser Arbeit wird jedes Bauteil auch als Bauobjekt betrachtet.

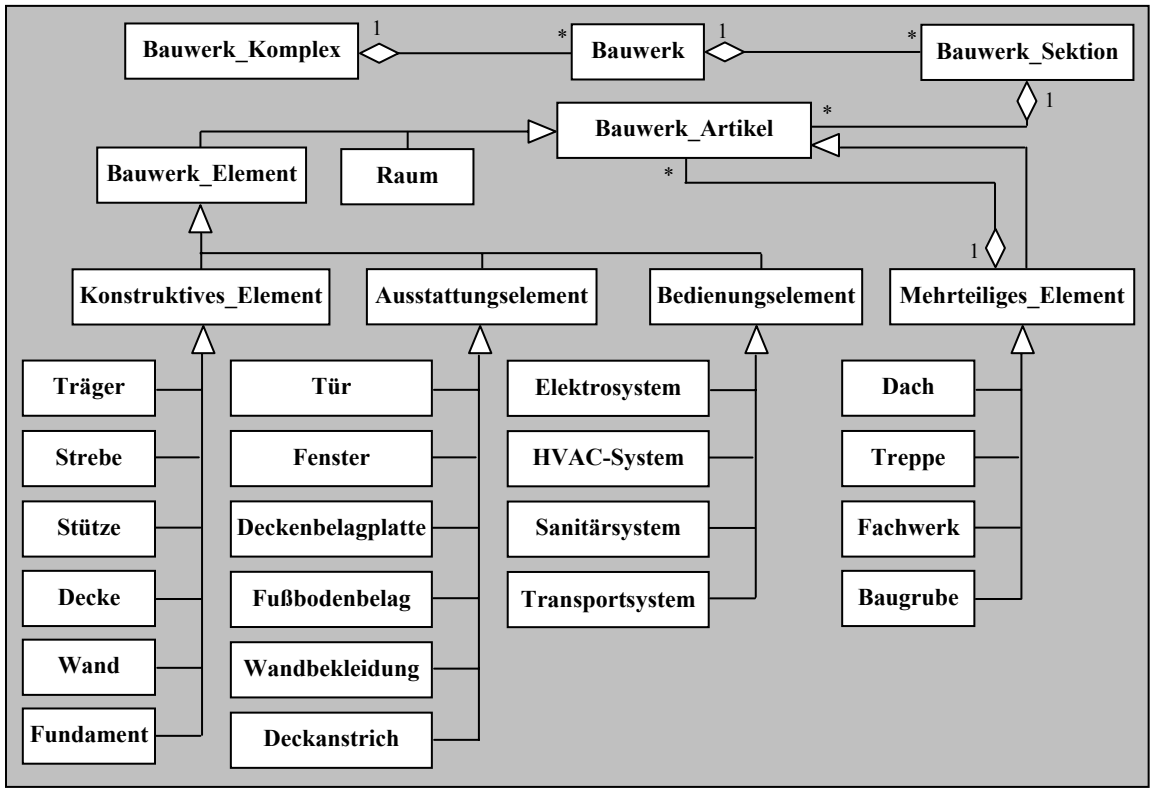


Abb. 5.14: Das Klassendiagramm der Struktur des Bauobjektsmodells

Grundsätzlich kann das Bauwerk aus unterschiedlichen Sichten modelliert werden. Aus der Sicht seiner Konstruktion kann das Bauwerk modelliert werden, wobei hier die Analyse der konstruktiven Struktur des Bauwerkes im Mittelpunkt steht. Das Bauwerk kann auch aus der Sicht seiner Funktion modelliert werden, in dem die Analyse der Benutzung des Bauwerkes im Vordergrund steht.

Eine effiziente Modellierung der Konstruktion des Bauwerkes ist der Ausgangspunkt zu einer Integration der Planung und Ausführung, weil dadurch eine einheitlich strukturierte Darstellung des Bauwerks und seiner Bauteile vorhanden ist. Solche Modellierung ist auch von großer Bedeutung für die Baukalkulation, denn aufbauend auf diese modellierte Struktur können die Kosteninformationen als Attribute der Objekte dieser Struktur gespeichert werden.

Zu einem späteren Zeitpunkt können diese Kosteninformationen für die Erstellung von Angebotskalkulationen anderer gleichartiger Bauwerke angewendet werden. Insbesondere die Strukturierung des Bauwerkes in Sektionen und Elemente, sowie seiner Darstellung als Teil eines Bauwerkkomplexes erleichtert diese Aufgabe, in der eine detaillierte Vergleichmöglichkeit zur Verfügung gestellt wird.

Das Bauobjektsmodell spezifiziert in der vorliegenden Arbeit die konstruktive Struktur eines Bauwerkes in Klassen, die (in Anlehnung an ISO-10303 AP225) in einem Klassendiagramm dargestellt werden (Abb. 5.14).
In diesem Diagramm wird die Klasse Bauwerk_Komplex als Aggregation der Klasse Bauwerk identifiziert.
Die Klasse Bauwerk wird als Aggregation der Klasse Sektion betrachtet, wobei die Klasse Sektion ihrerseits als Aggregation der Klasse Bauwerk_Artikel analysiert wird. Die Klasse Bauwerk_Artikel wird als Generalisierung der Klassen Bauwerk_Element, Raum und Mehrteiliges_Element identifiziert, wobei die Klasse Mehrteiliges_Element als eine Aggregation der Klasse Bauwerk_Artikel betrachtet wird. Die Klasse Bauwerk_Element wird als Generalisierung der Klassen Konstruktives Element, Ausstattungselement und Bedienungselement spezifiziert.

Die technischen Anlagen, die im Rahmen der Modellierung in der vorliegenden Arbeit als Bedienungselemente betrachtet werden, werden insbesondere aus der Sicht des Facility Management als Teil des Bauwerkes betrachtet.

■ Das Geometriemodell

Zur Abbildung der topologischen und topografischen Form der Objekte im Hochbau werden u. a. verschiedene 3D geometrische Objekte eingesetzt, deren Basis die Geometrieobjekte Punkt, Linie, Fläche und Volumen sind.

In Abhängigkeit vom notwendigen Detaillierungsgrad werden die Objekte des Objektmodells geometrisch beschrieben. Für die einfachste Stufe des Detaillierungsgrades ist die Darstellung eines Objektes als Hüllkörper grundsätzlich ausreichend. Für eine höhere Genauigkeit der Darstellung sind komplexere Modelle wie Randflächendarstellung (B-Rep) oder konstruktive Körpergeometrie (CSG) notwendig, möglicherweise auch Hybride zwischen diesen Modellen.

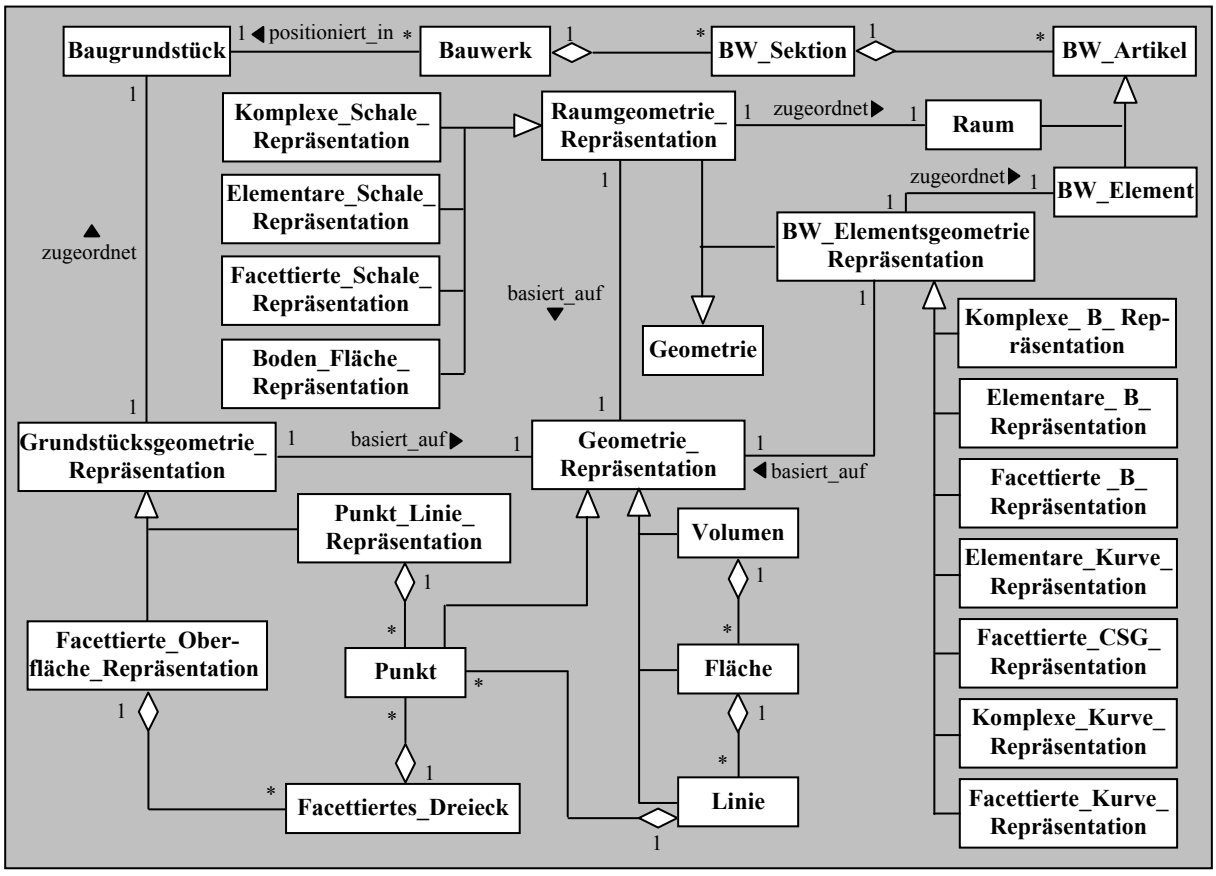


Abb. 5.15: Das Klassendiagramm des Geometriemodells

In Abhängigkeit vom notwendigen Detaillierungsgrad werden die Objekte des Objektmodells geometrisch beschrieben. Für die einfachste Stufe des Detaillierungsgrades ist die Darstellung eines Objektes als Hüllkörper grundsätzlich ausreichend. Für eine höhere Genauigkeit der Darstellung sind komplexere Modelle wie Randflächendarstellung (B-Rep) oder konstruktive Körpergeometrie (CSG) notwendig, möglicherweise auch Hybride zwischen diesen Modellen.

Das Geometriemodell, welches in Abbildung 5.15 in Form eines Klassendiagramms dargestellt ist, spezifiziert die geometrischen Objekte, die zur Beschreibung der Objekte des Hochbaus verwendet werden. Eine Beschreibung der Geometrie der Objekte des Hochbaus ist in [STEP AP 225] detailliert dargestellt.

Zur geometrischen Beschreibung von Objekten im Hochbau, die aus geraden Liniensegmenten bestehen, sowie solche Elemente, die durch flache Flächen begrenzt sind, werden u. a. facettiert geometrische Elemente wie z. B. facettierte Kurven und facettierte Dreiecke eingesetzt.

Die geometrische Beschreibung der Klasse Grundstück kann sowohl durch eine facettierte Repräsentation als auch durch die einfache Beschreibung mit Punkten und Linien durchgeführt werden. Zur geometrischen Beschreibung der im Geometriemodell definierten Klasse Raum wird die B-Rep eingesetzt. Grundsätzlich werden bei der geometrischen Beschreibung der kompletten Form von Bauwerkselementen folgende geometrische Objekte eingesetzt:

- elementare geometrische Objekte wie Kurven und Flächen
- komplexe geometrische Objekte, die aus jeder Art von Kurven und Flächen bestehen
- elementare und facettierte CSG-Objekte.

■ Das Bauvorbereitungsmodell

Im Bauvorbereitungsmodell werden die Objekte, die bei der Bauvorbereitung relevant sind, in Klassen zusammengefasst und in einem Klassendiagramm dargestellt (Abb. 5.16).

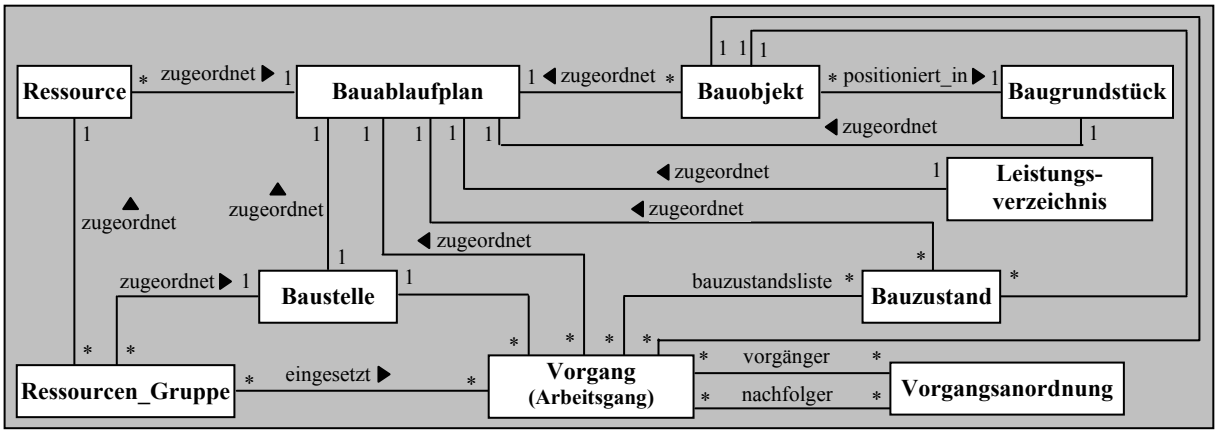


Abb. 5.16: Das Klassendiagramm des Bauvorbereitungsmodells

Die Basis des Bauvorbereitungsmodells bildet die Klasse Bauablaufplan. Diese Klasse steht mit den Klassen Leistungsverzeichnis, Bauobjekt, Baugrundstück und Ressource in einer Assoziation. Einem Bauablaufplan werden Vorgänge zugeordnet, in die Ressourcengruppen eingesetzt werden. Ressourcengruppen werden den Ressourcen zugeordnet und bilden die Grundlage zur Messung von Bezugsgrößen.

Prinzipiell werden Bezugsgrößen zur Messung der bei der Bauausführung geleisteten Arbeit und verbrauchter Ressourcen angewendet. Als Bezugsgrößen können die verschiedenen Maßnahmen der Bauausführung wie z. B. Fertigungszeit, Schalungszeit, Betonierungszeit, bearbeitete Flächen, Raumflächen, Energieverbrauch sowie unmittelbar berechenbare Kosten wie Lohn- und Materialkosten betrachtet werden.

Zwischen den Vorgängen bestehen technisch bedingte Reihenfolgebeziehungen. Diese Reihenfolgebeziehungen werden im Klassendiagramm durch die Klasse Vorgangsanordnung modelliert. Die Festlegung der Reihenfolge ist notwendig. Es ist aber vorteilhaft, diese Festlegung erst so spät wie möglich zu treffen, um möglichst mehr Freizügigkeit bei der Bauablaufplanung bis kurz vor der Baudurchführung zu gewährleisten. Die Klasse Vorgang präsentiert die Struktur einer rekursiven Assoziation. Demzufolge hat diese Klasse eine rekursive Struktur, die eine beliebige Anzahl von untergeordneten und übergeordneten Stufen (Nachfolger und Vorgänger) gestattet. Diese rekursive Struktur wird durch die Klasse Vorgangsanordnung im Diagramm dargestellt.

Mit einer optimierten Baudurchführung kann eine baudurchführungsbegleitende Kalkulation der Kosten in Bauprojekten realisiert werden. Zur optimierten Baudurchführung gehört auch der Einsatz von Verfahren der Netzplantechnik zur Bauablaufsteuerung wie z. B. die Vorgangsorientierten Modelle *Critical Path Method* (CPM) und *Metra Potential Method* (MPM) oder das Ereignisorientierte Modell *Program Evalution and Review Technique* (PERT).

5.3.1.2 Klassenmodellierung des Systems der Baukalkulation

Die Modellierung der Baukalkulation innerhalb dieser Arbeit bezieht sich aus Platzgründen hauptsächlich auf die Angebotskalkulation.
Zur Identifikation der Klassen wird auf der Basis des Anforderungsmodells sowie in Bezug auf Klassenmodelle des Bauprojektmodells eine Liste der wichtigen Begriffe des Problembereiches erstellt. Diese Liste wird dann gefiltert, wobei Begriffe von scheinbaren Klassen entfernt werden oder als Attribute oder Beziehungen zwischen Klassen erfaßt werden.
Aus dieser Liste werden auch Attribute und Methoden von Klassen identifiziert. Nach dieser Filterung entsteht die Liste der Basisklassen des zu modellierenden Systems.
Zuerst werden hier Namen der Objekte (in der realen Welt sehen wir nur Objekte, die dann zu Klassen zusammengefaßt werden), ihre allgemeinen Attribute und ihre statischen Beziehungen miteinander identifiziert. Hier wird die Identifikation der Methoden und dynamischen Beziehungen nicht im Mittelpunkt stehen. Es ist aber sinnvoll, Methoden, die im Laufe dieser Modellierungsphase identifiziert werden können, auch hier zu erfassen.

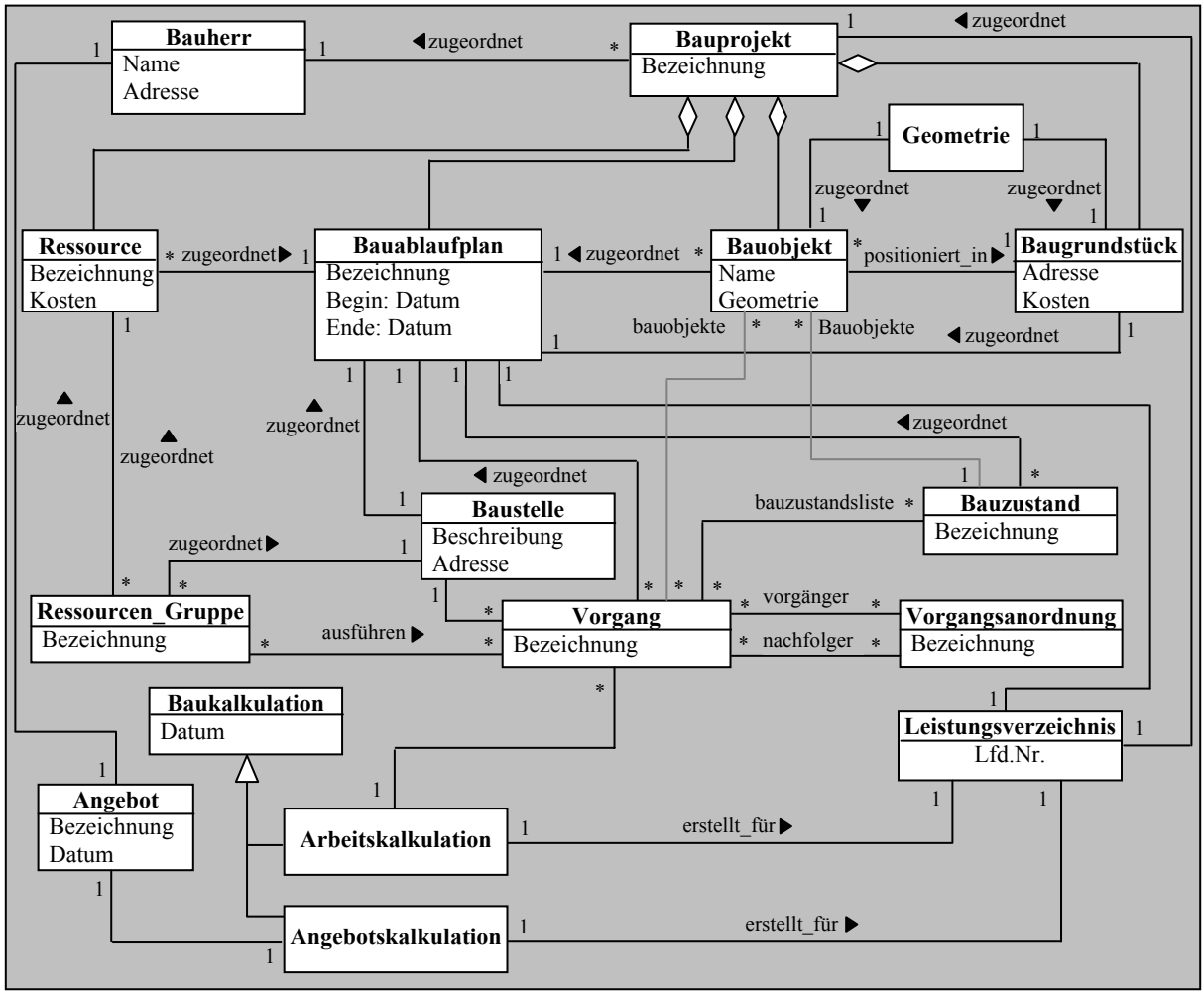


Abb. 5.17: Das Klassendiagramm der Baukalkulation

Im Rahmen der Modellierung der Baukalkulation werden sowohl Klassendiagramme in der Analysephase als auch Klassendiagramme in der Entwurfsphase erstellt. Klassendiagramme der Analysephase beinhalten die Klassen mit ihren Namen und ein Teil der Attribute sowie die Beziehungen zwischen diesen Klassen. Klassendiagramme der Entwurfsphase sind eine Verfeinerung der Klassendiagramme der Analysephase und beinhalten möglichst komplette Attribute der Klassen und zusätzlich die Methoden der Klassen.

In Abbildung 5.17 sind die identifizierten Klassen in einem Klassendiagramm dargestellt. Die Basis des Objektmodells der Baukalkulation bilden die Klassen Bauobjekt, Ressource, Bauablaufplan und Leistungsverzeichnis. Diese Klassen präsentieren die Struktur einer rekursiven Aggregation. Demzufolge haben diese Klassen eine rekursive Struktur, die eine beliebige Anzahl von untergeordneten Stufen gestattet. Anhand dieser rekursiven Aggregation können unterschiedliche anpassungsfähige Strukturen dargestellt werden.
Die Grundlage für die Angebotskalkulation sind Bauleistungen, welche im Leistungsverzeichnis beschrieben sind. Es ist hier zu berücksichtigen, dass ein Leistungsverzeichnis mehrere Positionen beinhaltet.

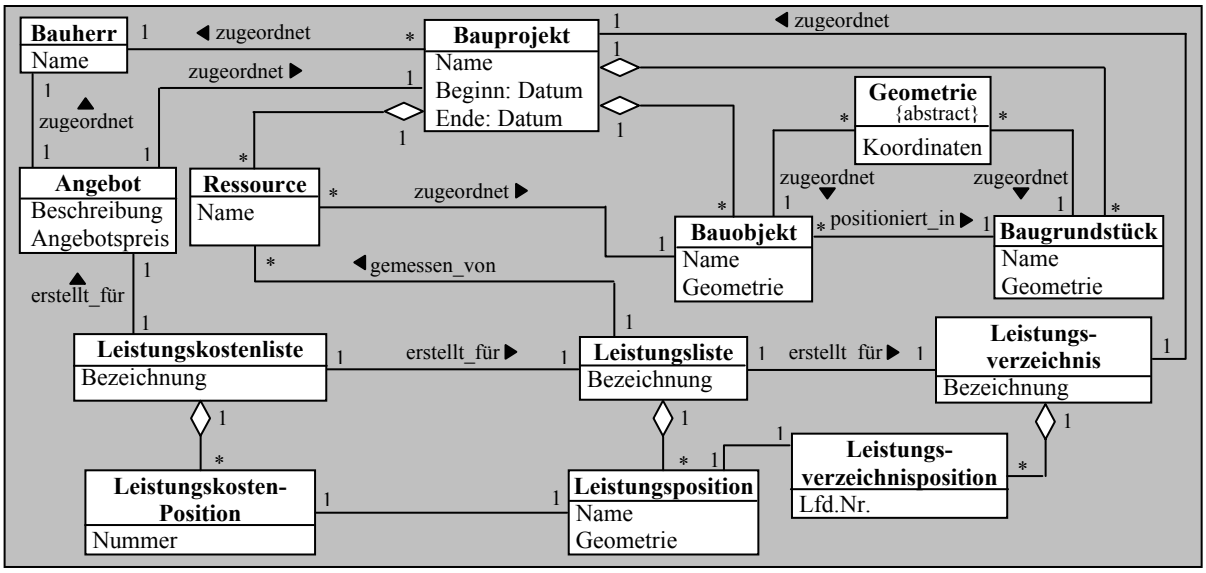


Abb. 5.18: Das Klassendiagramm der Angebotskalkulation

In Abbildung 5.18 ist das Subsystem Angebotskalkulation anhand eines Klassendiagramms dargestellt. Dieses Diagramm wird in der Entwurfsmodellierung verfeinert. Das Klassendiagramm der Arbeitskalkulation ist in Abbildung 5.19 gezeigt.

Es ist von großer Bedeutung an dieser Stelle zu betonen, dass bei der Modellierung der Teilsysteme der Kosten in Hochbauprojekten darauf zu achten ist, dass eine Kooperation zwischen diesen Teilsystemen und den Akteuren dieser Systeme unumgänglich ist. Es wäre sinnvoller, diese Tatsache noch bei der Modellierung zu berücksichtigen und Schnittstellen zwischen den Teilsystemen zu entwerfen um damit ein kooperatives Arbeiten zu erleichtern.

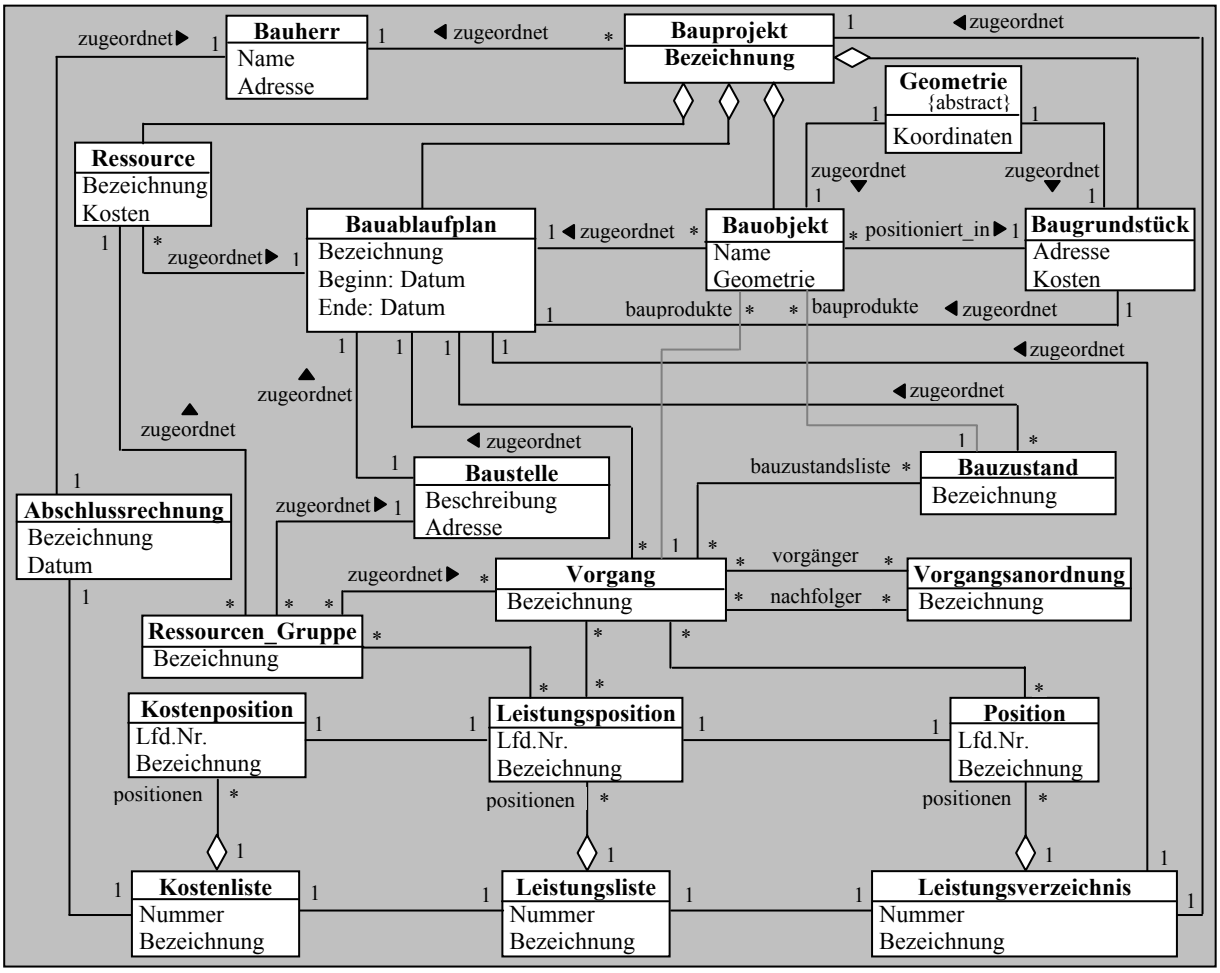


Abb. 5.19: Das Klassendiagramm der Arbeitskalkulation

Grundsätzlich wird der Überblick über das System nach der Analysemodellierung übersichtlicher. Daher ist es notwendig, die Anwendungsfalldiagramme zu verfeinern.
Das Wasserfallmodell als Vorgehensmodell zur Softwareentwicklung, wonach die Software für den ganzen Problembereich entwickelt wird, bringt viele Nachteile mit sich [Balz00]. Nach diesem Vorgehensmodell wird das Softwaresystem unter der Annahme entwickelt, dass die Anforderungen gleich bleiben. Dies ist aber meistens ein Fehlurteil. Ein weiteres Fehlurteil ist die Annahme, dass ein System komplett und fehlerfrei modelliert werden kann, ehe mit der Implementierung angefangen wurde. Daher ist es optimal, Software iterativ zu entwickeln. Dementsprechend sollen die Phasen der Entwicklung nicht für das ganze System sondern nur für ein Subsystem ausgeführt werden, und dies im weiteren Verlauf der Entwicklung verbessert und ergänzt und auf das ganze System ausgedehnt werden. In Hinblick auf diese Feststellungen und nach der bisherigen Beschreibung des Systems der Baukalkulation stellt sich heraus, dass es sinnvoller ist, dieses System in Subsysteme und nicht als Ganzes zu modellieren. Diese Subsysteme, welche unterschiedlichen Zwecken dienen, können auch als Systeme modelliert werden, wobei die Modellierung des Subsystems der Arbeitskalkulation als Erweiterung der Modellierung des Subsystems Angebotskalkulation betrachtet werden kann. Im weiteren Verlauf der Modellierung in der vorliegenden Arbeit wird nur das Subsystem Angebotskalkulation berücksichtigt.

5.3.2 Das dynamische Modell

Bei der anwendungsfallorientierten Vorgehensweise zur Modellierung des Systems der Baukalkulation, wie es in der vorliegenden Arbeit der Fall ist, umfasst die Modellierung des Verhaltens des Systems die Spezifikation der Dynamik auf der Basis der spezifizierten Anwendungsfälle. Zu solcher Spezifikation, welche auch nur mit einer verbalen Beschreibung der Anwendungsfälle erfolgen kann, werden sowohl Aktivitätsdiagramme als auch Sequenzdiagramme eingesetzt.

Die Bedeutung der Aktivitätsdiagramme zur Beschreibung von Abläufen und Prozessen bei der Objektorientierten Modellierung wurde bereits im OMT [Rumb93] mit dem Einsatz von Datenflussdiagrammen erkannt. Aktivitätsdiagramme als UML-Konzept beschreiben das Verhalten des Systems anhand der Verantwortlichkeit und die Reihenfolge des Ablaufs sowie das Wirken der einzelnen Gänge dieses Ablaufs innerhalb des Systems. Aktivitätsdiagramme erweisen sich als hilfreiches Werkzeug bei der Identifikation von Klassen, deren Attribute und Operationen.

In der Bauablaufplanung sind die zur Errichtung eines Bauwerkes geforderten Leistungen, die grundsätzlich in einem Leistungsverzeichnis beschrieben sind, in ihrer zeitlichen Folge darzustellen, um daraus den terminlichen Einsatz der Ressourcen abzuleiten. Bei der Bauablaufplanung werden unterschiedliche Methoden wie z. B. Balkendiagramme und Netzplantechnik eingesetzt. Die Netzplantechnik, deren Grundlage die Graphentheorie ist, ist in der DIN 69900 genormt. Basiselemente der Netzplantechnik sind Ereignisse, Vorgänge und Anordnungsbeziehungen.

Um das System der Kosten in Bauprojekten deutlicher zu spezifizieren, wird die Anwendungsfallbeschreibung durch Aktivitätsdiagramme dargestellt. Bei dieser Darstellung können sowohl einzelne Anwendungsfälle als auch das gesamte System oder dessen Subsysteme durch Aktivitätsdiagramme dargestellt werden. In den Aktivitätsdiagrammen werden Objekte und Objektflüsse identifiziert. Hier wird auch der Zustand der Objekte in eckigen Klammern unter dem Objektnamen angezeigt.

Das in der Abbildung 5.7 dargestellte Anwendungsfalldiagramm der Baukalkulation wird in Abbildung 5.20 durch ein Aktivitätsdiagramm gezeigt. Innerhalb dieses Aktivitätsdiagramms werden die Verantwortlichkeitsbereiche Bauunternehmen und Bauherr (zusammen mit dem Architekt) festgestellt.

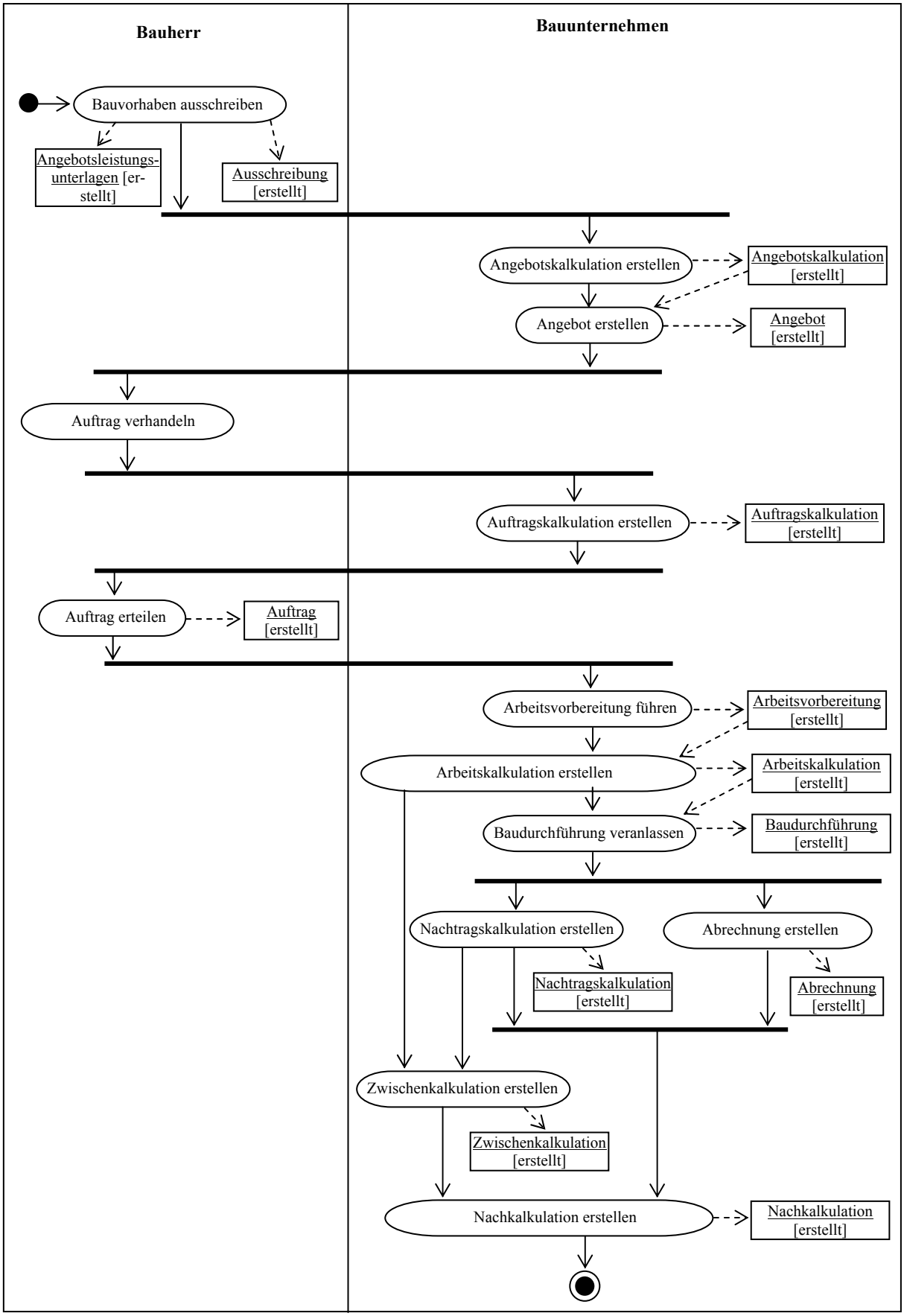


Abb. 5.20: Aktivitätsdiagramm zur Darstellung der zeitlichen Einordnung der verschiedenen Baukalkulationsarten

Die Bearbeitung eines Angebotes setzt mehrere Aktivitäten voraus, die in Abbildung 5.21 durch ein Aktivitätsdiagramm dargestellt werden, wobei hier der Zeitraum vom Zugang der Ausschreibungsunterlagen bis zur Bekanntgabe des Angebots berücksichtigt wird.

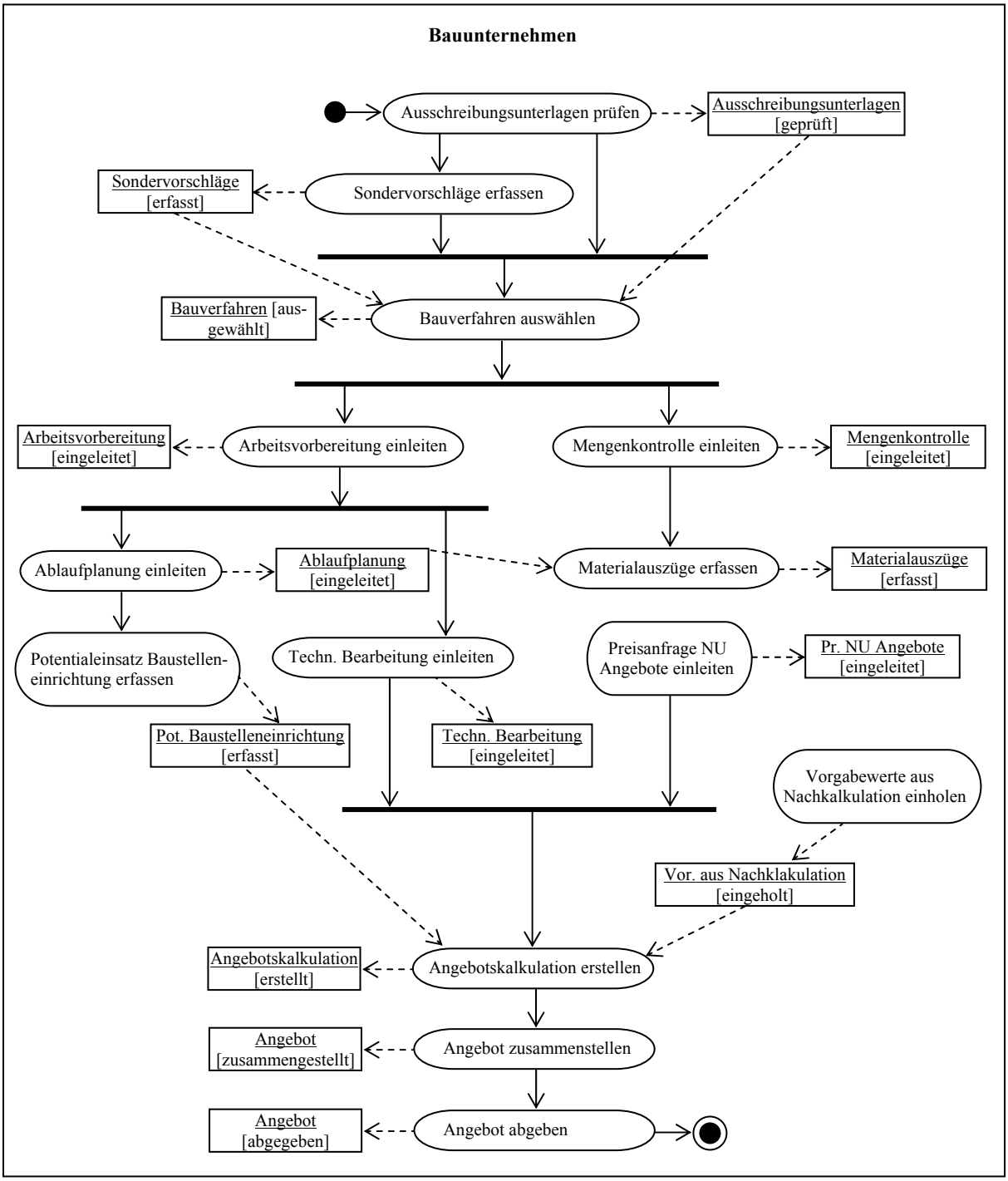


Abb. 5.21: Aktivitätsdiagramm der Bearbeitung eines Angebots

Die Arbeitsmaßnahmen bei der Angebotskalkulation sind in der Abbildung 5.22 durch ein Aktivitätsdiagramm dargestellt.

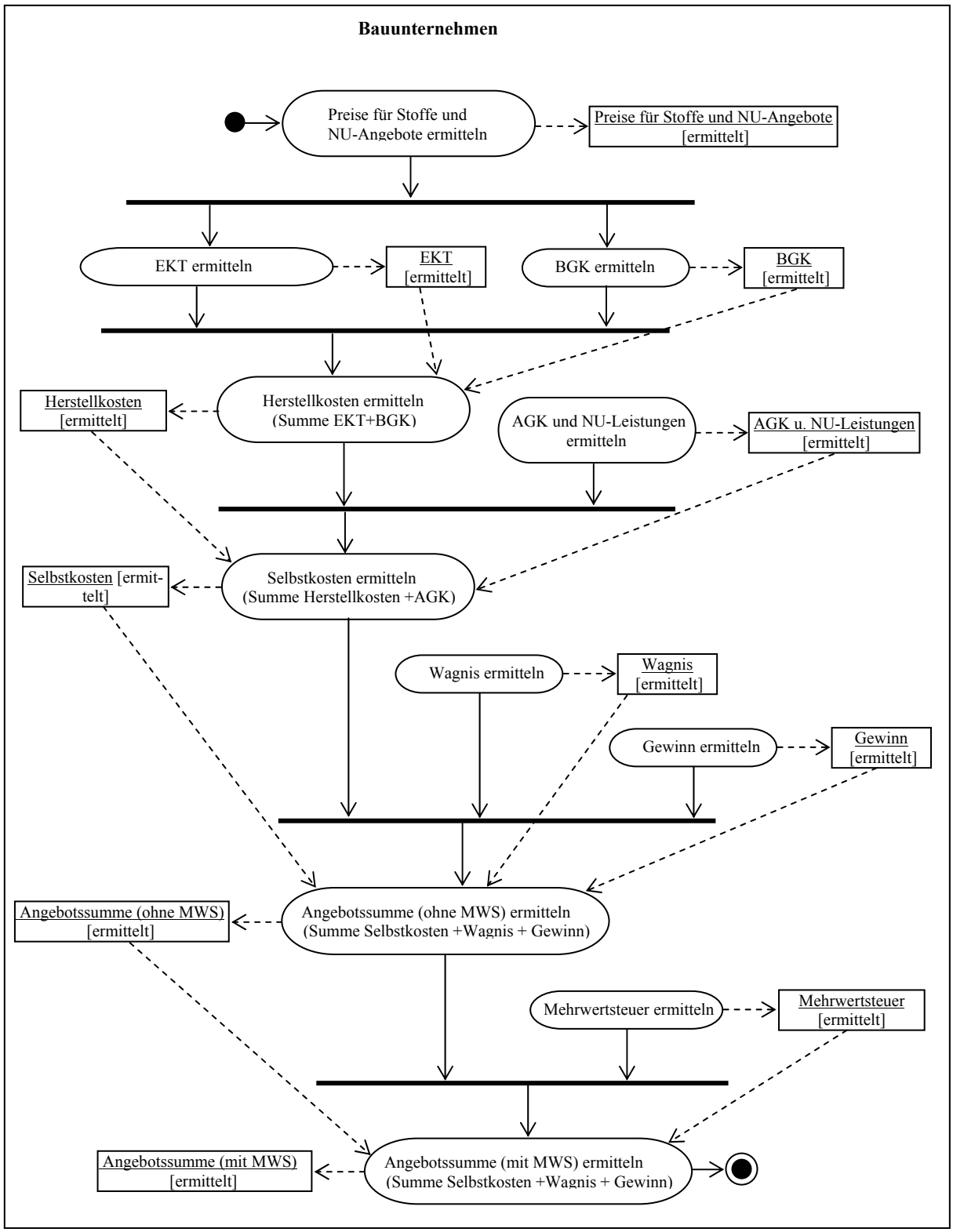


Abb. 5.22: Aktivitätsdiagramm der Angebotskalkulation

Die Arbeitsmaßnahmen zur Angebotskalkulation, die in der Abbildung 5.22 dargestellt sind, gliedern sich in:

- Ermittlung der Preise für Stoffe und Nachunternehmerleistungen (NU-Leistungen)
- Berechnung der Einzelkosten der Teilleistungen
- Berechnung der Gemeinkosten der Baustelle
- Ermittlung der Zuschläge für Allgemeine Geschäftskosten und NU-Leistungen
- Berechnung der Zuschläge für Wagnis und Gewinn
- Ermittlung der Umlagen

Die Modellierung des dynamischen Verhaltens des Systems dient auch dazu, sich einen besseren Überblick bei der Definition der Operationen der Klassen zu verschaffen.
Die Sequenzdiagramme dienen der Spezifikation des Ablaufs des Nachrichtenaustausches in einem Szenario innerhalb eines Anwendungsfalls. Damit soll festgestellt werden, welche Objekte wann aktiv sind. Hauptelemente eines Sequenzdiagramms sind aktive Objekte, die exemplarisch das Verhalten aller Objekte einer Klasse beschreiben.

Ein Objekt wird als ein Kasten oberhalb einer gestrichelten senkrechten Linie abgebildet. Diese Linie wird Objektlebenslinie genannt, wobei eine Nachricht mit einem Pfeil zwischen zwei Objektlebenslinien abgebildet wird. Die Reihenfolge der ausgetauschten Nachrichten und die Dauer der verursachten Aktionen werden entlang der Objektlebenslinie von oben nach unten gezeigt. Falls ein Objekt eine Nachricht an sich selbst sendet, wird dies durch einen Pfeil, der zurück auf die gleiche Objektlebenslinie gehen wird, gekennzeichnet.

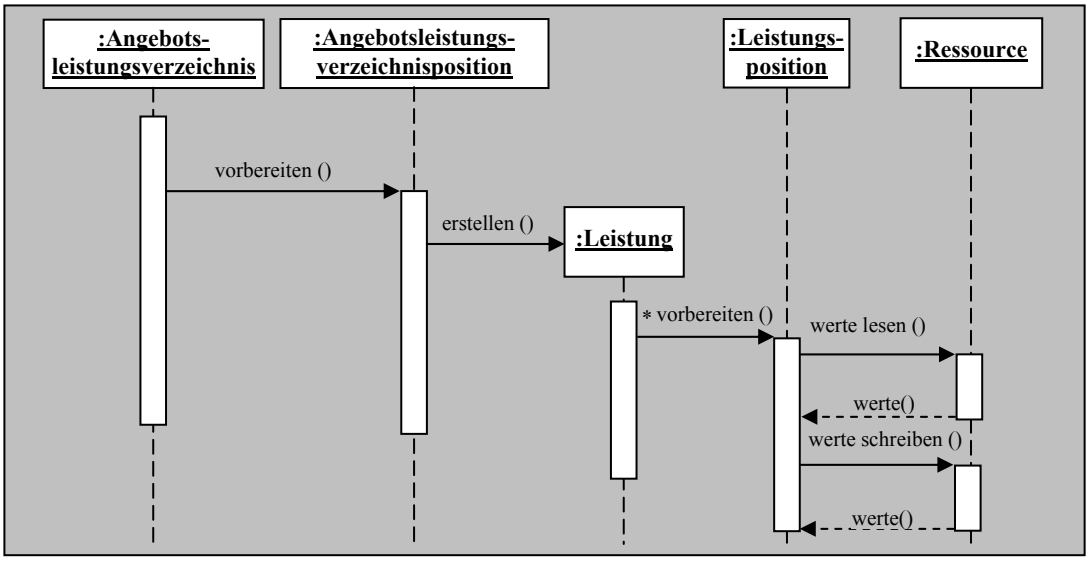


Abb. 5.23: Das Sequenzdiagramm des Anwendungsfalls Angebotskalkulation erstellen (Szenario 1)

Sequenzdiagramme beinhalten unterschiedliche Steuerinformationen wie Einschränkungen und Iterationsmarkierungen. Durch Einschränkungen wird angezeigt, wann eine Nachricht gesendet wird. Iterationsmarkierungen werden eingesetzt, wenn eine Nachricht mehrfach an mehrere Empfängerobjekte gesendet wird. Sequenzdiagramme können auch zur Modellierung von nebenläufigen Prozessen eingesetzt werden. Dies kann durch mehrere Aktivitätszonen zur gleichen Zeit dargestellt werden. Im Klassendiagramm werden Objekte, die miteinander kommunizieren, durch Assoziationen dargestellt. Um diese Assoziationen bei der dynamischen Modellierung nützlich zu machen, wird das Klassendiagramm anhand von Sequenzdiagrammen abgeglichen.

Zur Spezfikation der Dynamik im Anwendungsfall Angebotskalkulation erstellen werden einige Szenarios ausgewählt, in denen diverse Objekte dieses Anwendungsfalls beteiligt sind. Das erste Szenario beinhaltet die Objekte Angebotsleistungsverzeichnis, Angebotsleistungsverzeichnisposition, Leistung und Leistungsposition. Das Sequenzdiagramm dieses Szenarios ist in Abbildung 5.23 gezeigt.

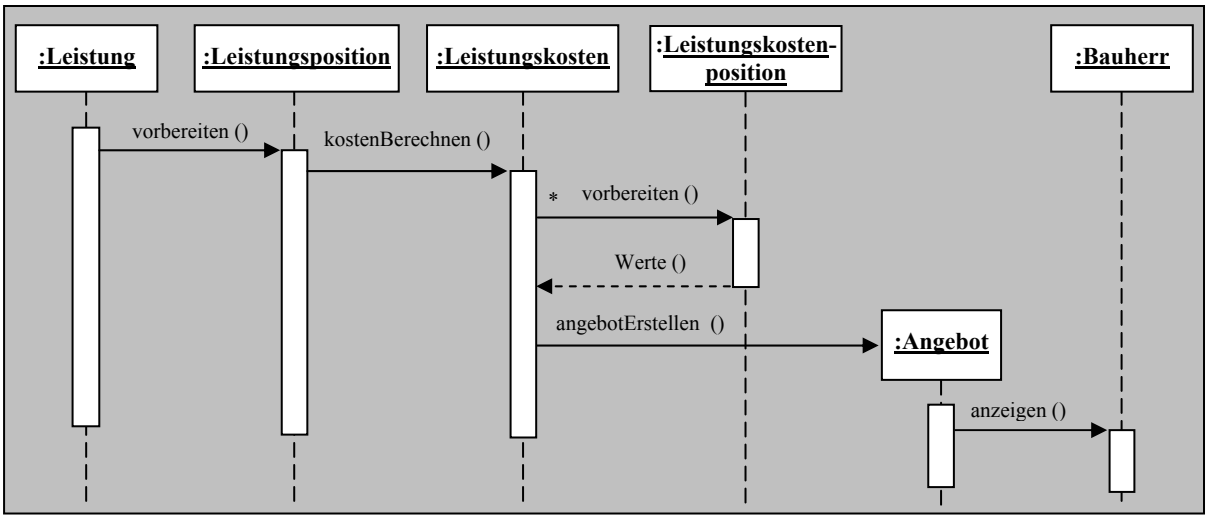


Abb. 5.24: Das Sequenzdiagramm des Anwendungsfalls Angebotskalkulation erstellen (Szenario 2)

Das zweite Szenario umfasst die Objekte Leistung, Leistungsposition, Leistungskosten Leistungskostenposition, Angebot und Bauherr. Das Sequenzdiagramm dieses Szenarios ist in Abbildung 5.24 dargestellt.

6 Objektorientierter Entwurf des Systems der Kosten in Bauprojekten

6.1 Beschreibung der angewendeten Konzepte

In der Entwurfsphase wird festgelegt, welche Implementierungsumgebung eingesetzt wird, wobei in der vorliegenden Arbeit die Programmiersprache Java wegen ihrer positiven und fortschrittlichen Eigenschaften verwendet wird.
Java ist verhältnismäßig einfach, konzeptionell eindeutig, Objektorientiert, plattformunabhängig, portabel und netzwerkfähig. Java unterstützt das Konzept *Garbage Collection* zur automatischen Bereinigung der Speicher von Daten, die nicht mehr benötigt werden. Java unterstützt auch das Internet und die nebenläufige Programmierung *(multi threading)* und besitzt eine reichhaltige Klassenbibliothek.

Ergebnisse der Analysemodellierung eines Systems werden in der Entwurfsphase mit der gewählten Implementierungsumgebung abgestimmt. Es handelt sich dabei um die Verfeinerung der Modelle der Analysephase hinsichtlich dieser Implementierungsumgebung und durch den Einsatz von bestimmten Regeln, insbesondere der Entwurfsmuster *(Design Patterns)*. Entwurfsmuster sind verlässliche Lösungen für ständig wiederholte Entwurfsprobleme [Gamm96]. Bei der Verfeinerung der Klassendiagramme geht es u. a. darum, die Konzepte der Objektorientierung optimal anzuwenden. So werden die Klassen der Klassendiagramme des Strukturmodells überprüft, ob sie in der Tat weiter als Klassen betrachtet werden oder sinnvoller als Attribute oder Methoden von anderen Klassen modelliert werden.

Das Konzept der Vererbung ist außerordentlich hilfreich bei der Implementierung von Klassen, die in Klassenbibliotheken vorhanden sind. Mit dem Einsatz der Vererbung kann eine Spezialisierungsklasse zur Verfügung gestellt werden, die die Rolle eines Adapters zwischen inkompatiblen Klassen übernehmen kann. Mit inkompatiblen Klassen wird hier gemeint, dass eine bestehende Klasse (Bibliotheksklasse) zusammen mit anderen Klassen des Modells eingesetzt werden, die aber nicht die geforderte Schnittstelle besitzen. Am Anfang des Einsatzes der Objektorientierung wurde das Konzept der Vererbung gern angewendet. In den letzten Jahren hat sich aber herausgestellt, dass dieses Konzept auch Nachteile mit sich bringt, nämlich die Schwächung der Kapselung, die als Basiskonzept der Objektorientierung betrachtet wird. Grundsätzlich kann das Konzept der Vererbung eingesetzt werden, wenn u. a. eine Basisklasse erweitert werden soll und wenn diese keine Rolle, sondern eine besondere Prägung der Subklassen darstellt. Dazu soll sich die Vererbung innerhalb des Problembereiches befinden.

In der Entwurfsphase wird auch die Auflösung der Modellierungskonzepte, die in der eingesetzten Programmiersprache nicht unterstützt werden, bereitgestellt. Hier wird das Konzept der Mehrfachvererbung als Beispiel genannt, das von der in der vorliegenden Arbeit eingesetzten Programmiersprache Java nicht unterstützt wird. Manchmal ist es aber zweckmäßig, die Mehrfachvererbung zu verwenden, wie z. B. beim Einsatz des Adapter-Entwurfsmusters. Dieses Entwurfsmuster hat die Aufgabe, die Harmonisierung von zwei Klassen, die kooperieren sollen, obwohl ihre Schnittstellen nicht zusammenpassen. In Java kann dieses Entwurfsmuster anhand des Interface-Konzeptes aufgelöst werden [Seem00]. Dies bedeutet aber nicht, dass dieses Konzept als Werkzeug zur Realisierung der Mehrfachvererbung in Java betrachtet werden kann. In Java können Methoden von Klassen als Interfaces deklariert werden und diese auch gemeinsam von einer Klasse implementiert werden. Dies hat aber mit dem Konzept

der Mehrfachvererbung keine Gleichheit, da die Vererbung das Ziel hat, die Codewiederverwendung zu gewährleisten, und dieser Code, der in mehreren Klassen benötigt wird, wird nur in der Superklasse aufgeführt und mit der Vererbung an die Subklassen weitergeleitet. Nachfolgend werden einige Konzepte, welche bei der Entwurfsmodellierung im Rahmen der vorliegenden Arbeit eingesetzt werden, dargestellt.

- **Interface**

Eine Klasse umfasst sowohl die Schnittstelle (*Interface*) als auch die Implementierung. Ein *Interface*, welches nur aus einer Auflistung von Methoden besteht, kann in Java als eine Klasse ohne Implementierung betrachtet werden. Eine Klasse kann nur dann ein Interface implementieren, wenn sie alle in der Deklaration des Interface vorgegebene Methoden besitzt. Durch dieses Konzept wird in Java die Benutzungsbeziehung realisiert. So kann eine Klasse eine andere Klasse benutzen, in dem sie spezielle Methoden dieser Klasse anwendet, um ihre Funktionalität zu gewährleisten. Um die Klassen flexibler zu machen und überflüssig angewendete Vererbung zu beseitigen, ist es im Sinne eines korrekten Klassenentwurfs bedeutend, das Interface-Konzept einzusetzen.

- **Abstrakte Klasse**

Mit der Deklaration von einer Klasse als abstrakte wird beabsichtigt, dass von dieser Klasse keine Instanzen gebildet werden können. Abstrakte Klassen unterscheiden sich von anderen Klassen dadurch, dass bei den abstrakten Klassen einige der Methoden nicht implementiert sind, also ihre Realisierung wurde auf später verschoben. Damit wird man andere Klassen von der abstrakten Klasse erben lassen und dabei die abstrakten Methoden einsetzten. Die Deklaration der abstrakten Klassen in Java wird durch das Schlüsselwort *abstract*, direkt vor die Klasse notiert, erfolgen. Wenn dieses Schlüsselwort direkt vor eine Methode notiert ist, wird diese Methode als abstrakt definiert und damit wird die umgebende Klasse automatisch als abstrakt deklariert.

- **Generische Klasse**

Mit dem Konzept der generischen Klasse *(parametrized class)* können Klassen mit formalen Parametern beschrieben werden. Damit kann eine Sammlung von Klassen definiert werden. Generische Klassen sind solche Klassen, deren Methoden auf generischen Datentypen operieren. Unter generisch wird verstanden, dass der Datentyp, auf dem die Methoden agieren, noch nicht festgelegt ist. Die generischen Datentypen werden zur Übersetzungszeit gebunden. Generische Klassen können zur Realisierung von Containerklassen verwendet werden. Eine Containerklasse ist bei der Verwaltung von Instanzen einer anderen Klasse nützlich. So stellt eine Containerklasse Methoden zum Zugriff auf diese verwalteten Instanzen bereit. Die Containerklasse *Java.util.Vector* wird in der vorliegenden Arbeit mehrfach eingesetzt, weil ihre Länge nach der Erzeugung nach Bedarf wachsen und abnehmen kann. Zur Umsetzung des Konzepts der generischen Klasse in Java und die Erweiterung von Java um generische Konzepte sei auf [Eise98] verwiesen.

6.2 Verfeinerung der Klassendiagramme

Zur Verfeinerung der Klassendiagramme werden einige Ausschnitte des Klassendiagramms der Angebotskalkulation in der Analysephase (Abb. 5-18) betrachtet. Diese Ausschnitte werden detailliert spezifiziert und mit dem Einsatz von geeigneten Konzepten, wie z. B. Entwurfsmuster, verfeinert. Diese Ausschnitte werden dann zu einem Klassendiagramm zusammengefasst, das als Klassendiagramm der Entwurfsphase betrachtet wird.

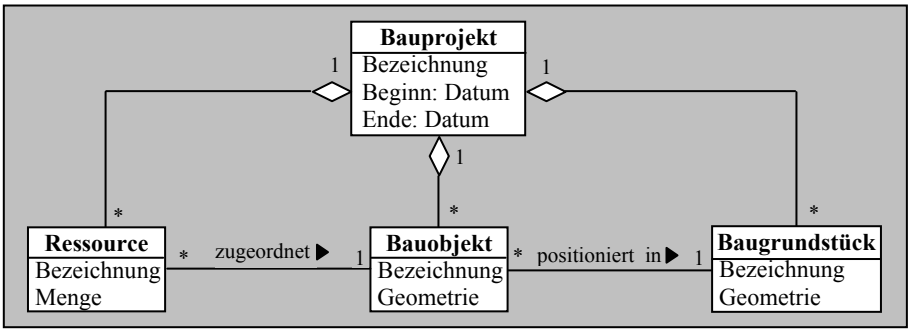


Abb. 6.1: Ausschnitt 1 des Klassendiagramms der Angebotskalkulation

Im Ausschnitt 1 des Klassendiagramms der Angebotskalkulation wird die Klasse Bauprojekt als Aggregation der Klassen Ressource, Bauobjekt und Baugrundstück dargestellt (Abb. 6.1). Es ist jedoch sinnvoller, die Klasse Bauprojekt als Komposition der abstrakten Klasse Bauprojektkomponente zu modellieren. Damit wird die Klasse Bauprojektkomponente als Schnittstelle zum Rest des System betrachtet. Diese Klasse wird dann als Generalisierung der Klassen Ressource, Bauobjekt und Baugrundstück spezifiziert (Abb. 6.2). Sie enthält Methoden, die durch diese Klassen realisiert werden.

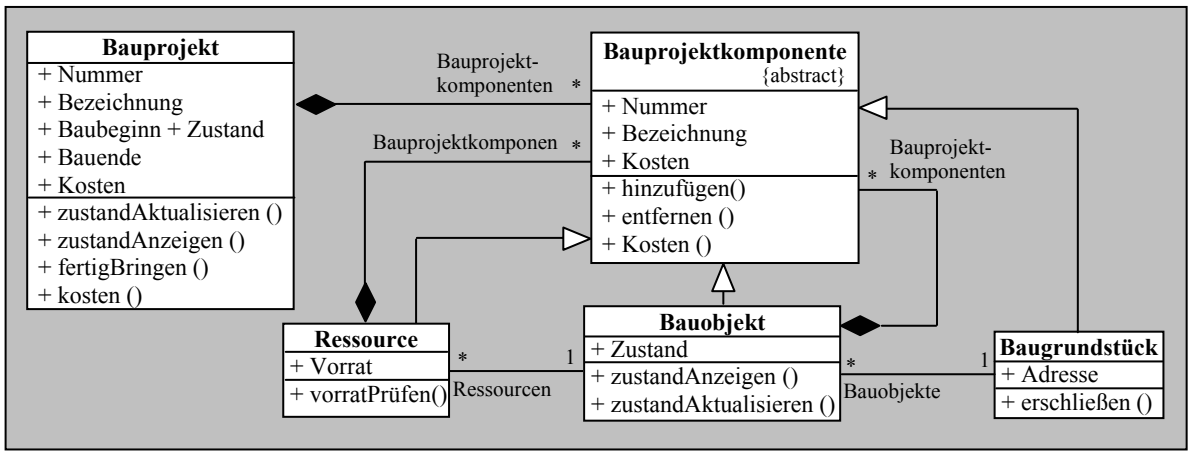


Abb. 6.2: Verfeinerung des Klassendiagramms des Ausschnitts 1

Die Klasse Bauprojektkomponente wird damit als abstrakte Klasse deklariert. Sie stellt eine Methode zur Berechnung der Kosten zur Verfügung. Diese Methode funktioniert so, dass zuerst die bisher eingesetzten Ressourcen ermittelt werden, dann wird die Berechnungsfunktion der Kosten aufgerufen und aus dem zurückgelieferten Wert die bisher angefallenen Kosten ermittelt. Von der abstrakte Klasse Bauprojektkomponente werden die Subklassen Bauobjekt, Baugrundstück und Ressource abgeleitet, die jeweils ihre eigene Berechnungsfunktion der Kosten besitzen.

Generell können die Objekte Bauobjekt und Ressource aus mehreren Bauprojektkomponenten bestehen. Dies bedeutet, dass der Einsatz des Kompositum-Entwurfsmusters zur Modellierung dieses Sachverhaltes sinnvoll ist (Abb. 6.2). Bei dem Einsatz dieses Entwurfsmusters wird erreicht, dass die Klassen des Systems nur durch die Schnittstelle Bauprojektkomponente auf die Komponenten des Bauprojektes zugreifen können. Dadurch wird die Zusammensetzung der Komponenten des Bauprojektes unsichtbar und auf diese Weise die Abhängigkeiten zwischen den Klassen des Systems verringert. Mit dem Einsatz dieses Entwurfsmusters wird auch die Klassenstruktur des Systems leicht erweiterbar. Die zusammengesetzten Bauprojektkomponenten werden durch die Klassen Bauobjekt und Ressource verwaltet.

Diese Klassen können die Komposition durch die Containerklasse *Vector* realisieren. So wird zuerst ein Containerobjekt durch den Konstruktor der Klasse angelegt. Die Methoden hinzufügen () und entfernen () überschreiben die entsprechenden Methoden in der Oberklasse Bauprojektkomponente und fügen Objekte der Klasse Bauprojektkomponente in den Container hinzu bzw. entfernen sie. Die Einzelkosten der im Container enthaltenen Objekte der Klasse Bauprojektkomponente werden durch die Methode kosten () summiert und dann zurückgeliefert. Bei einfachen Bauprojektkomponenten wie Baugrundstück werden die Kosten direkt zurückgeliefert.

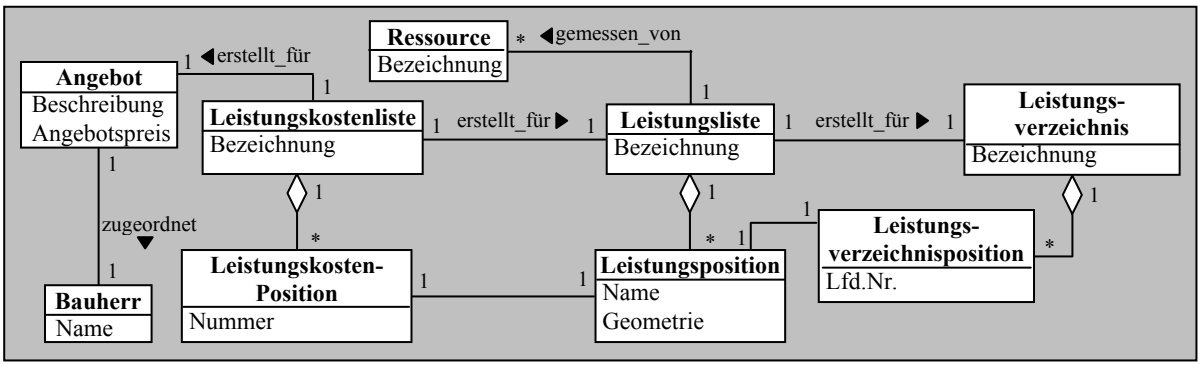


Abb. 6.3: Ausschnitt 2 des Klassendiagramms der Angebotskalkulation

In Ausschnitt 2 werden die Klassen Leistungsverzeichnis, Leistungsverzeichnisposition, Leistungskostenliste, Leistungskostenposition, Leistungsliste, Leistungsposition und Angebot, bezüglich ihrer Nützlichkeit im System spezifiziert (Abb. 6.3).

Bei der Erstellung der Angebotskalkulation wird in der Praxis eine Liste der Einzelkosten der Teilleistungen erstellt. Diese Liste beinhaltet eine Beschreibung der Einzelkostenentwicklung und Mengenangaben laut Leistungsverzeichnis. In dieser Liste werden auch die Einzelkosten der Ansätze je Einheit sowie die erforderlichen Ressourcen dargestellt. Dann wird die Zusammenstellung der Stunden, Lohn- und Materilalkosten sowie der Kosten für Nachunternehmer-

leistungen in einer anderen Liste dargestellt. Eine weitere Liste wird zur Ermittlung der Einheitspreise, der Positionspreise und der Angebotssummen erstellt.
Da diese drei Listen für dieselben Objekte erstellt werden, können sie zu einer Liste zusammengefasst werden, die in der vorliegenden Arbeit als Leistungskostenliste bezeichnet wird (Abb. 6.4). Auf der Basis dieser Liste kann eine Angebotsliste erstellt werden. Diese Angebotsliste wird durch die Klasse Angebot verwaltet, wobei sie als Komposition der Klasse Angebotsposition dargestellt wird.

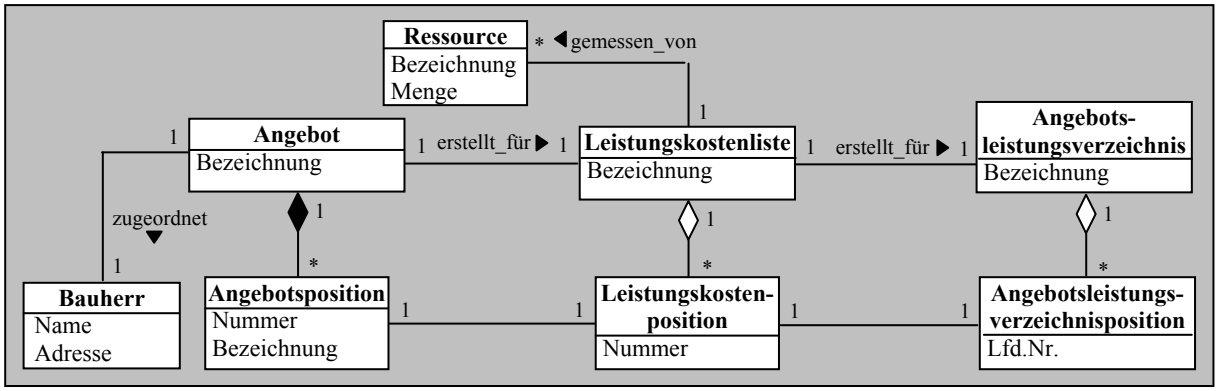


Abb. 6.4: Verfeinerung des Klassendiagramms des Ausschnitts 2

Bei der Verfeinerung dieses Ausschnittes werden solche Aspekte wie die Delegation der Verwaltung von Objekten an Container spezifiziert. Im Klassendiagramm der Angebotskalkulation wird eine 1:n-Beziehung zwischen einigen Klassen spezifiziert, die durch Aggregation modelliert werden kann. Es ist sinnvoller, die Verwaltung von n Elementen, die aus einer 1:n-Beziehung resultieren, durch eine Containerklasse abzuwickeln. So wird die Verwaltung der Positionen des Angebotsleistungsverzeichnisses, der Leistungskostenliste und des Angebots durch die Containerklasse *Vector* realisiert.
Containerklassen des Paketes *java.util* verwalten Elemente der Klassen, die aus der Java-Klasse *Object* abgeleitet sind. Bei dem Java-Einsatz befinden sich alle Klassen in der Hierarchielinie der Klasse *Object*. Dies gilt in Java implizit für alle Klassen, selbst wenn sie nicht als *extends Object* deklariert sind.

Die verfeinerten Ausschnitte werden dann zu einem Klassendiagramm der Entwurfsphase der Angebotskalkulation zusammengefasst (Abb. 6.5). Auf der Grundlage dieses Klassendiagramms wird der Java-Code in der Implementierungsphase erstellt.

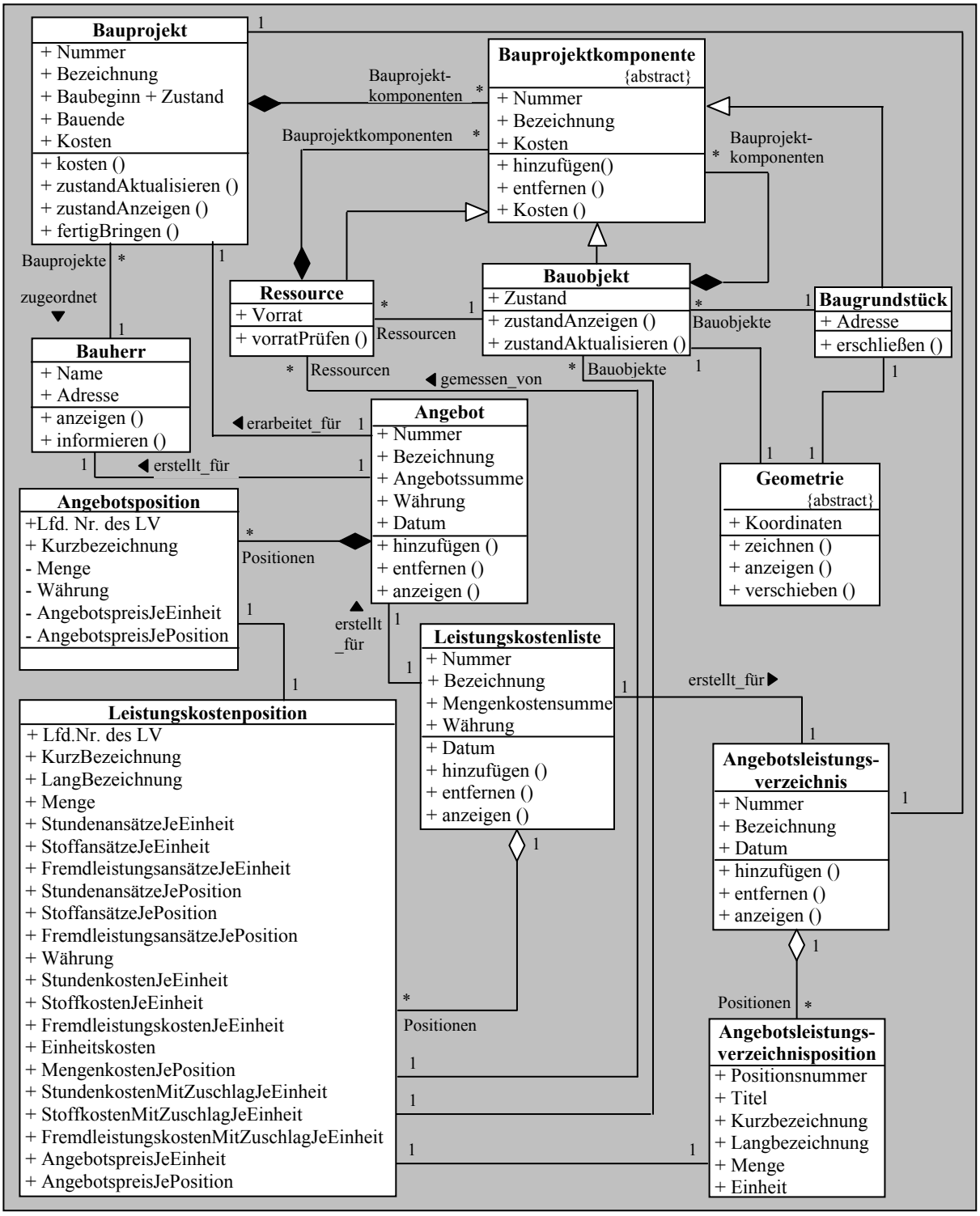


Abb. 6.5: Klassendiagramm der Entwurfsphase der Angebotskalkulation

6.3 Überarbeitung und Partitionierung des Strukturmodells

Hier wird das Strukturmodell der Analysephase in Hinsicht auf eventuelle Erweiterungen und Änderungen der Funktionalität des Systems überarbeitet. Das bedeutet, dass das Strukturmodell so überarbeitet wird, dass Erweiterungen und Änderungen der Funktionalität des Systems möglichst begrenzte Wirkung haben soll. Zu diesem Zweck werden die Klassen des Systems in Informationsklassen, Schnittstellenklassen und Steuerungsklassen klassifiziert [Jaco92]. Informationsklassen (*Entity-Classes*) sind solche, die Informationen darstellen, die nicht nur auf einzelne Anwendungsfälle Einwirkung haben. Schnittstellenklassen (*Boundary-Classes*) sind solche, die Verhalten und Struktur von Benutzer- und Systemschnittstellen abbilden. Steuerungsklassen (*Control-Classes*) sind solche, die insbesondere die Koordinationsaufgaben der Anwendungsfälle erfüllen. Bei der Überarbeitung des Strukturmodells wird das Klassendiagramm der Angebotskalkulation als Ausgangspunkt betrachtet. Aus diesem Klassendiagramm weisen die Klassen Angebot, Bauherr, Geometrie und Angebotsleistungsverzeichnis Schnittstellen auf.

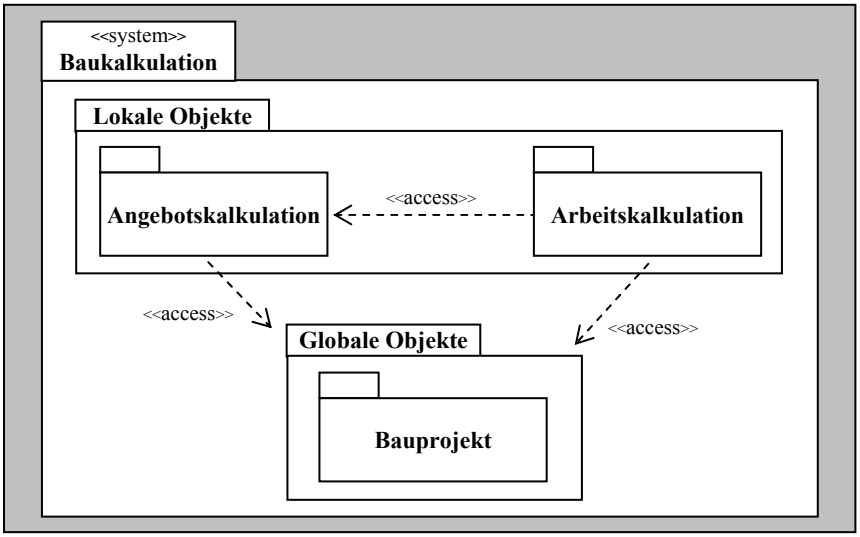


Abb. 6.6: Partitionierung des Strukturmodells

Bei der Partitionierung des Strukturmodells handelt es sich um dessen Darstellung in Paketen durch den Einsatz des Paket-Konzeptes von UML. Dieser Schritt wird durchgeführt, weil dieses Modell nicht mehr als Ganzes dargestellt werden kann. Da sich die Tendenz der Softwareentwicklung im Konstruktiven Ingenieurbau nach einem Vernetzt-kooperativen Konzept fortbewegt, ist es sinnvoll, bei der Partitionierung des Strukturmodells festzulegen, welche Objekte lokal sind und welche global verteilt werden sollen. Diese Gliederung der Objekte spielt eine große Rolle sowohl bei der Aufteilung des Strukturmodells, als auch bei der Definition der Aspekte der physischen Verteilung der Applikationen über das Netzwerk. In Abbildung 6.6 ist die Partitionierung der Pakete des Strukturmodells sowie die *<<acces>>*- Abhängigkeiten zwischen diesen Paketen dargestellt.
Bei dieser Gliederung des Strukturmodells kommt es darauf an, welche Objekte über lokale Formen verfügen und welche globale, die für alle Clients vorhanden seien sollen, wobei sie in einer zentralen Datenbank gespeichert werden. Während Informationsobjekte als globale Objekte betrachtet werden können, werden Benutzerschnittstellenobjekte als lokale Objekte eingeordnet. Objekte des Subsystems Bauprojekt werden als globale Objekte betrachtet. Schließlich sollen sie für alle Beteiligten im Bauprozess zugänglich gemacht werden

6.4 Darstellung der Verteilungs- und Kommunikationskonzepte

Hier werden einige geeignete Verteilungs- und Kommunikationskonzepte, die den Einsatz der verteilten Anwendungen unterstützen, dargestellt. In diesem Zusammenhang wird der Einsatz solcher Technologien wie Java und CORBA in einem verteilten System beschrieben. Dann wird die Tauglichkeit dieser Konzepte zum Einsatz in den Systemen des Konstruktiven Ingenieurbaus erläutert. Der Einsatz von DCOM *(Distributed Component Object Model)* als verteiltes Objektverwaltungssystem kann nur als eine Alternative betrachtet werden, wenn *MS-Windows* als Plattform eingesetzt wird.

6.4.1 Einsatz der Java-Technologie in verteilten Systemen

Die Plattform Java 2, welche eine umfangreiche Architektur zur Entwicklung und Einrichtung netzwerkbasierter Anwendungen präsentiert, wird in drei Versionen angeboten. Bei der ersten Version handelt es sich um die *Java 2 Standard Edition* (J2SE). J2SE bietet als Plattform eine zuverlässige Grundlage zur Entwicklung und Implementierung netzwerkbasierter Unternehmensanwendungen im Bereich des Desktop-PC und Workgroup-Server. J2SE wird durch das Java 2 Software Development Kit (SDK), Standard Edition, und Java 2 Runtime Environment, Standard Edition, implementiert. Als zweite Version wird die Java 2 Micro Edition (J2ME) betrachtet. Dabei handelt es sich um eine stark optimierte Java Laufzeitumgebung, die bei ressourcenkritischen Anwendungen in Consumer-Geräten, wie z. B. Bildschirmtelefonen, digitalen Set-Top-Boxen, Mobiltelefonen und Autonavigationssystemen angewendet wird. Bei der dritten Version handelt es sich um das Java 2 Enterprise Edition, welche nachfolgend gesondert dargestellt wird.

6.4.1.1 Java 2 Enterprise Edition

Java 2 Enterprise Edition (J2EE) wird als Plattform für unternehmensweite Anwendungen zur Realisierung einer effektiven Integration von Java-Technoligien betrachtet. Bei unternehmensweiten Anwendungen handelt es sich hier um verteilte sowie internet-, intranet- und extranetbassierte Anwendungen im Unternehmen. Damit dient J2EE als Plattform für serverbasierte Anwendungen, sowie zur Bereitstellung komplexer Unternehmenslösungen. In der Softwareentwicklung bietet J2EE die Sicherheit, Portabilität und Performance der Java 2 Plattform, sowie plattformübergreifende Kompatibilität und Skalierbarkeit im Bereich der Netzwerkanwendungen. Die J2EE beinhaltet eine Reihe von Java-Technologien innerhalb einer Architektur mit einem ausgedehnten *Application Programming Model* (APM), sowie einer *Compatibility Test Suite*. In Abbildung 6.7 ist die J2EE-Architektur und Interoperabilität dargestellt.

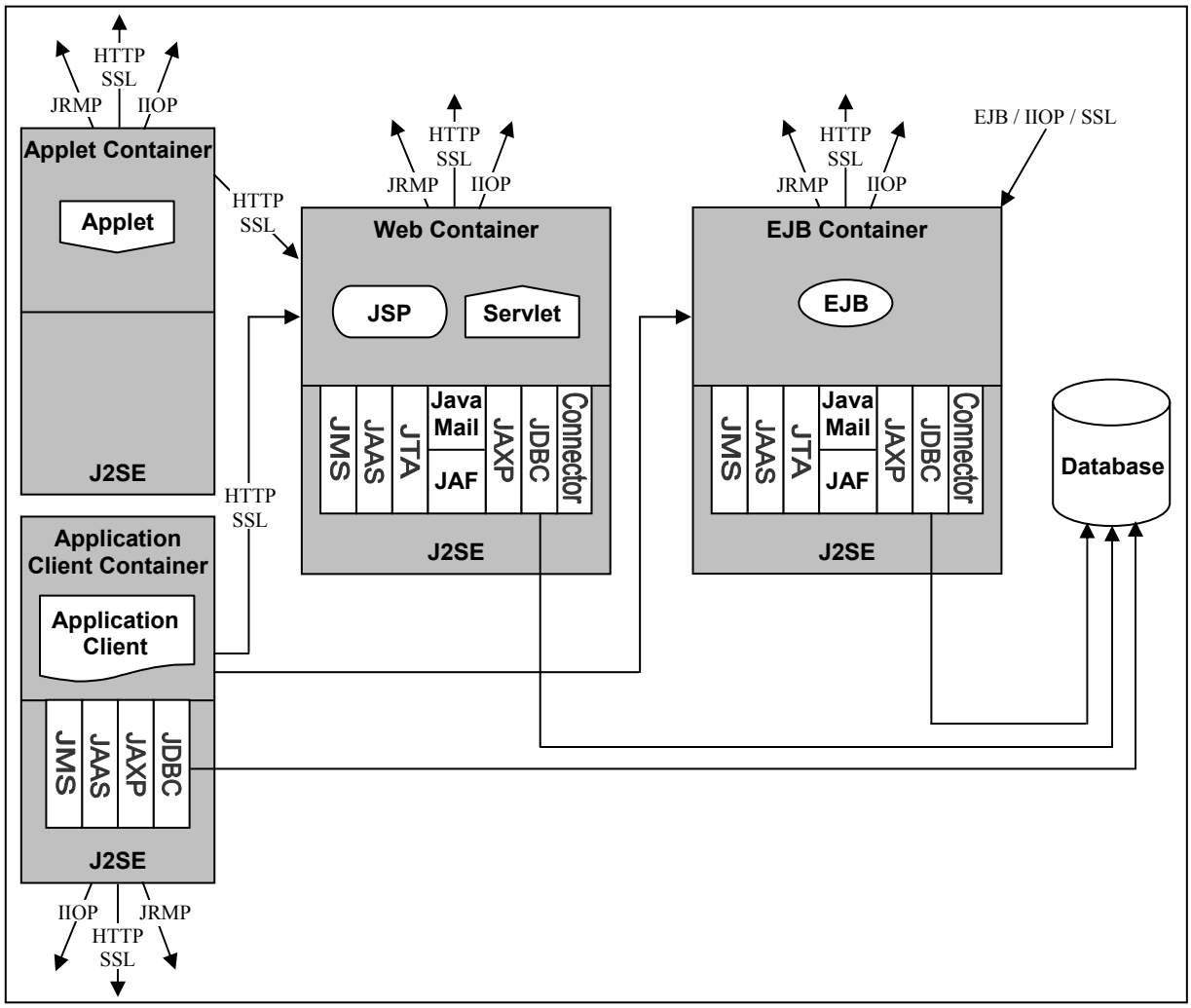


Abb. 6.7: Darstellung der Architektur und Interoperabilität von J2EE (Version 1.3)

Die J2EE-Laufzeitumgebung *(J2EE runtime enviroment)* besteht aus den folgenden Teilen:

- **Applikationskomponenten**
  Das APM, welches Regeln zur effektiven Anwendung von J2EE beinhaltet, definiert vier Typen von Applikationskomponenten, die ein J2EE-Produkt unterstützen soll:
  - **Application Clients**
    Application Clients sind typische GUI-Javaprogramme, die auf Desktop Computer ausgeführt werden. Sie bieten dem Anwender native Applikationen und haben Zugriff auf alle Ressourcen des J2EE-Middle-Tier.
  - **Applets**
    Applets sind Java-Programme, die in Webseiten integriert sind und auf Client-Rechnern innerhalb eines Webbrowsers ausgeführt werden. Grundsätzlich haben Applets eingeschränkte Rechte, wie z. B. nicht lesen und nicht schreiben.
  - **Java Servlets und Java Server Page**
    Java Servlets sind Programme zur Erweiterung der Funktionalität eines Servers, wie z. B. dessen Anbindung an eine Datenbank. So können mit *Java Servlets* leistungsfähige Web-Dienste für die meisten Webserver abgewickelt werden. Die *Java Servlet API* ist in den Pakete *javax.servlet* und *javax.servlet.http* definiert. Während das Paket *javax.servlet* Klassen zur Implementierung von protokollunabhängigen *Servlets* enthält,

umfasst das Packet *javax.servlet.http* Klassen zur Implementierung von Servlets, welche das HTTP Protokoll benutzen. *Java Server Page* (JSP) ermöglicht als Java-Technologie die Aufbereitung von dynamischen Inhalten. So werden *Java Server Pages* als HTML Seiten betrachtet, die zusätzlich eingebettete Java Befehle umfassen. Wenn ein JSP aufgerufen wird, erstellt der Server automatisch ein *Servlet* aus dem JSP, so dass der HTML Code aus dem JSP und die Java Befehle zusammen ausgeführt werden.

- **Enterprise JavaBeans**
  Enterprise JavaBeans (EJB) wird als ein Framework betrachtet, in dem die JavaBeans sowohl portabel, als auch verteilt eingesetzt werden können. Zur Kommunikation mit EJB kann sowohl RMI als auch CORBA/IIOP verwendet werden. Mit *JavaBeans* ist ein plattformunabhängiges Komponentenmodell zur Entwicklung von Softwarekomponenten in Java definiert. Diese Komponenten werden in IDEs *(Integrated Development Environments)* verwendet. In *JavaBeans*, welche in *Session Beans* und *Entity Beans* sich gliedern, werden *Events* und *Properties* beschrieben. Während bei *Session Beans* die Instanz immer nur einem Benutzer zugeordnet ist und üblicherweise zur Ausführung von Verarbeitungen eingesetzt wird, repräsentieren *Entity Beans* persistente Daten.

- **Container**
  Das APM gestattet die Konstruktion von Containern, mit denen leistungsfähige, skalierbare Ablaufumgebungen für EJB, Servlets und JSP realisiert werden können. Üblicherweise werden diese vom Applikationsserver- oder Webserveranbieter bereits fertig angeboten. Container sind hilfreiche Konzepte insbesondere für die Anwendungskomponenten. So übernehmen Container dessen Transaktionsmanagement, überwachen ihren Lebenszyklus und steuern den Zugriff auf Ressourcenmanager. Durch Applikationsschnittstellen (APIs), die im Container bereitgestellt werden, kann auf die J2EE-Dienstleistungen zugegriffen werden. Grundsätzlich wird eine Applikationskomponente in einem Container installiert und innerhalb von *Java Runtime Enviroment* (JRE) unter der Verwaltung des Containers ausgeführt.

- **Resource Manager Driver**
  Die *Resource Manager Driver* dienen zur Herstellung von entfernten Verbindungen zum Ressourcenmanager. Die JDBC-Treiber werden als typisches Muster von Ressourcenmanagertreibern betrachtet.

- **Datenbank**
  Die J2EE-Plattform beinhaltet Datenbanken zur persistenten Speicherung der Daten. Auf diese Datenbank wird von Web-Komponenten, EJB und *Application Clients* durch die JDBC-API zugegriffen.

Die J2EE-Standarddienste *(J2EE standard services)* umfassen folgende Applilkationsschnittstellen (APIs):

- **HTTP**
  HTTP (Hypertext Transfer Protocol) ist ein Protokoll zur Übertragung der Informationen zwischen WWW-Servern und WWW-Clients über das Internet. Während die API der HTTP-Clientseite im Java-Packet java.net definiert sind, ist die API der HTTP-Serverseite in der Servlet- und JSP-Interfaces spezifiziert.

- **HTTPS**
  HTTPS (Hypertext Transfer Protocol Secure) ist ein SSL-basiertes Protokoll, welches eine ähnliche Spezifikation wie HTTP besitzt. SSL (Secure Socket Layer) und das HTTPS gestatten eine abhör- und fälschungssichere Verbindung zwischen Web-Browser und Web-Server.

- **Java Transaction API (JTA)**
  JTA spezifiziert Schnittstellen zwischen Transaktionsmanager und den beteiligten Teilen eines verteilten Transaktionssystems, wie Ressourcenmanager, Applikationsserver und Transaktionsapplikationen. JTA, welches in den Paketen *javax.transaction* und *javax.transaction.xa* definiert ist, besteht aus zwei Teilen. Der erste Teil ist das *application-level demarcation interface* und wird bei den Container- und Applikationskomponenten zur Festlegung der Transaktionsbeschränkungen eingesetzt. Der zweite Teil ist ein Interface zwischen dem Transaktionsmanager und dem Ressourcenmanager, welches in der *J2EE SPI* -Ebene angewandt wird.

- **JavaIDL**
  Mit JavaIDL können J2EE-Applikationskomponenten entfernte CORBA-Objekte aufrufen werden, wobei hier das Protokoll IIOP *(Internet Inter-ORB Protocol)* verwendet wird. CORBA-Objekte können damit aufgerufen werden, unabhängig davon, in welcher Programmiersprache sie geschrieben sind und auf welcher Plattform sie sich befinden.

- **RMI-IIOP**
  Das RMI-IIOP Subsystem ist eine Ansammlung von APIs. Mit RMI-IIOP können Java-Anwendungen mit anderen Anwendungen über CORBA und ohne den Einsatz von CORBA-IDL kommunizieren. CORBA-Objekte können auch mit dem *OMG-Internet Inter-Orb Protocol* auf Java-Anwendungen zugreifen. RMI *(Remote Method Invocation)* ist ein API für *Java-to-Java* Kommunikation und benutzt JRMP *(Java Remote Method Protocol)* als Protokoll. Bei dem Einsatz von RMI-IIOP werden die *Distributed Garbage Collection* (DGC) und die *RMISocketFactory* nicht verwendet. Die *RMISocketFactory* dient zum Aufbau von eigenen Socket-Verbindungen zwischen Client und Server.

- **Java Database Connectivity**
  Java Database Connectivity (JDBC) ist eine API zur Ausführung von SQL-Anweisungen auf relationalen Datenbanken durch Java. JDBC ist sowohl von der verwendeten Datenbank als auch von der Art der Verbindung zur Datenbank unabhängig.

- **Java Messaging Service (JMS)**
  Java Messaging Service (JMS) wird als API zur Integration von *Message-Oriented Middleware* (MOM) eingesetzt. JMS, welches im Paket *javax.jms* realisiert ist, unterstützt sowohl die *Point-to-Point* Kommunikation, als auch die *Publish/Subscribe* Kommunikation.

- **Java Naming and Directory Interface**
  Java Naming and Directory Interface (JNDI) ist eine API zum Zugriff von Java-Anwendungen auf unterschiedlichen Namens- und Verzeichnisdiensten, wie *Remote Method Invocation* (RMI), *Domain Naming Service* (DNS), das *Lightweight Directory Access Protocol* (LDAP) und das Dateisystem. Die JNDI-API, welches im Paket *javax.naming* definiert ist, enthält die zwei Teile *application-level-interface* zum Einsatz bei Applikationskomponenten zum Zugriff auf Namens- und Verzeichnisdienste, sowie ein *Service Provider Interface* (SPI) zum Anbinden der Provider der Namens- und Verzeichnisdienste.

- **JavaMail**
  Da viele Internetanwendungen die Fähigkeit zum Senden von *Emails* erfordern, beinhaltet die J2EE-Plattform die *JavaMail-API* zur Definition der Schnittstelle zum Einsatz von *Emails* zur Benachrichtigung von Anwendern über das Internet. Die JavaMail-API enthält die zwei Teile *application-level-interface* zum Einsatz bei Applikationskomponenten zum Senden von *Emails*, sowie ein SPI zum Einsatz in der J2EE-SPI-Ebene.

- **JavaBeans Activation Framwork**
  JavaBeans Activation Framwork (JAF) wird von *JavaMail-API* benötigt und ist im Paket *javax.activation* definiert.

- **Java API for XML Parsing**
  Java API for XML Parsing (JAXP) wird zur Unterstützung von SAX *(Simple API for XML)* und DOM *(Document Object Model)* zur *XML-Parsing* eingesetzt. Unter *XML-Parsing* wird der Prozess zur Umwandlung der XML-Dateien in SAX oder DOM bezeichnet. SAX ist eine API zur Verarbeitung einer XML-Applikation mit einer Objektorientierten Programmiersprache. DOM ist ein Objektmodell zur Beschreibung der Elemente einer bestimmten XML-Anwendung als Objekte, die dann mit einer Objektorientierten Programmiersprache verarbeitet werden. XML *(Extensible Markup Language)* ist eine vereinfachte Form von SGML *(Standard Generalized Markup Language)*. XML und SGML werden zur Definition von HTML *(Hypertext Markup Language)* eingesetzt, wobei die mit XML definierten HTML-Dateien als XHTML bezeichnet werden.

- **J2EE Connector Architecture**
  Die *Connector Architecture* ist eine J2EE-SPI und erlaubt es, Ressourcenadapter, welche den Zugriff auf EIS *(Enterprise Informationssystems)* unterstützen, in J2EE-Produkte hinzuzufügen. Als EIS werden hier z. B. SAP R/3, CICS und IMS von IBM betrachtet. Die *Connector Architecture* ist erst ab der Version 1.3 der J2EE-Plattform als Bestandteil präsent.

- **Java Authentication and Authorization Service**
  Java Authentication and Authorization Service (JAAS) bietet Dienste zur Zugriffsberechtigung und Durchsetzung der Zugriffskontrolle von Anwendern. Dazu wird eine Java-basierte Version von PAM *(Pluggable Authentication Module)* genutzt und erweitert die Architektur der Zugriffskontrolle der Java 2 Pattform zu einem kompatiblen Verfahren zur Unterstützung von anwenderbasierter Zugriffsberechtigung.

6.4.1.2 Remote Method Invocation

Mit *Remote Method Invocation* (RMI) können JAVA-Objekte miteinander kommunizieren, auch wenn sie sich auf unterschiedlichen, auch entfernten, Plattformen befinden.

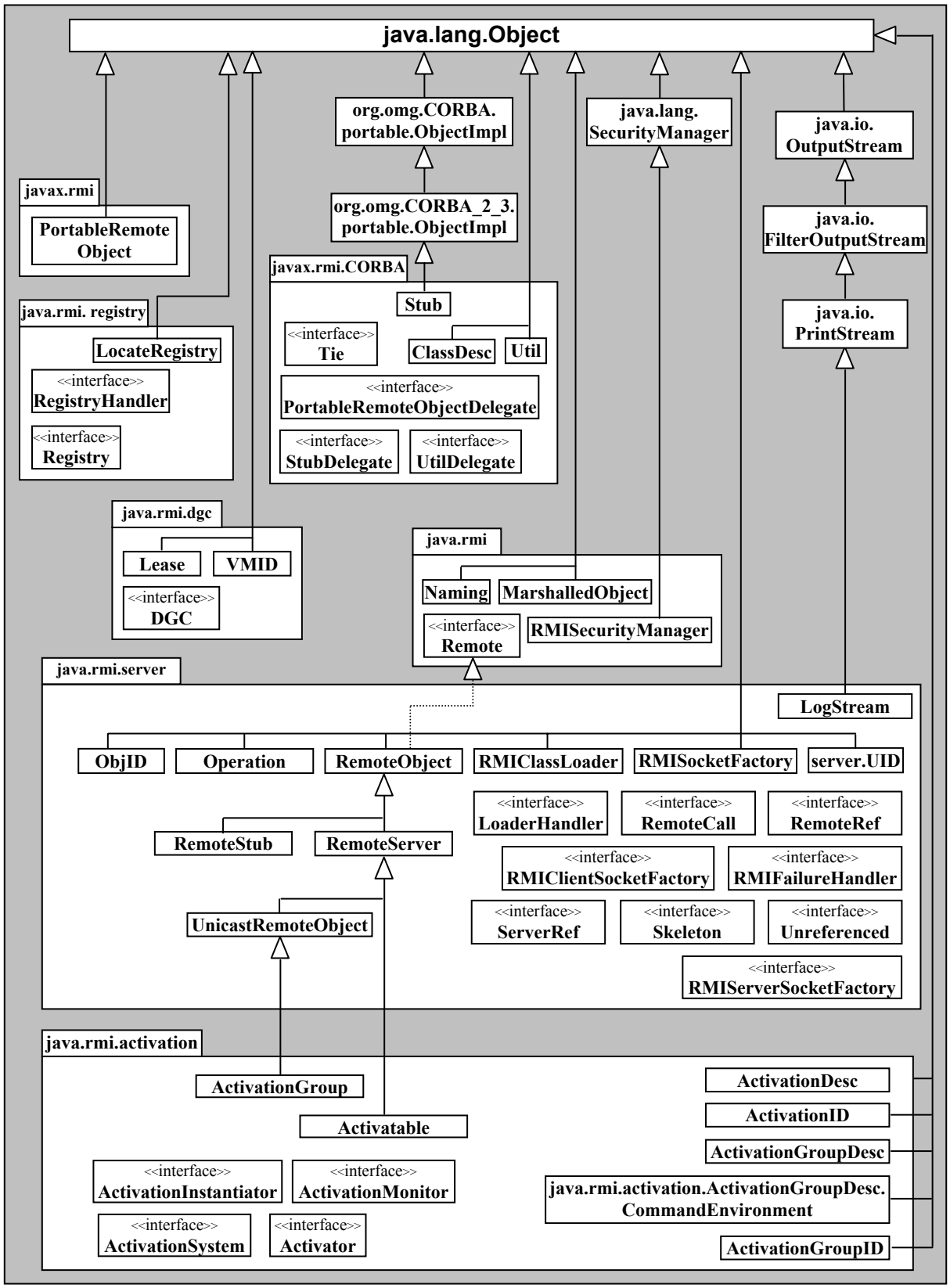


Abb.6.8: Darstellung der Klassenhierarchie der RMI-Pakete in Java 2

Da sich der Einsatz von RMI auf *Java-to-Java* Kommunikation einschränkt, ist der Einsatz von RMI nicht so komplex wie bei CORBA. RMI wird in mehreren Paketen in J2SE realisiert. Die Struktur dieser RMI-Pakete sowie deren Klassenhierarchie sind in Abbildung 6.8 dargestellt.

Die RMI Architektur ist in drei Ebenen aufgeteilt (Abb. 6.9):
- *Stub-Skeleton Layer*
- *Remote Reference Layer*
- *Transport Layer*

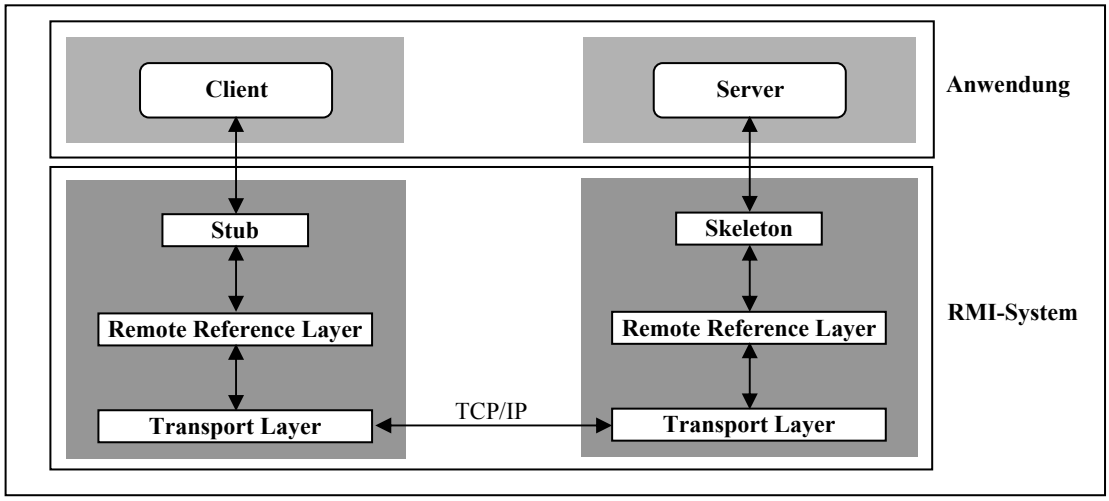


Abb. 6.9. Darstellung der RMI-Struktur

RMI realisiert die Transportschicht auf der Basis von *Java-Sockets* und TCP und benutzt dafür das *Java Remote Method Protocol* (JRMP). Java bietet allerdings auch andere Konzepte zur Kommunikation zwischen Clients und Servern, wie z. B. über *Sockets* und die Datenbankschnittstelle JDBC. Zur Realisierung von Objektorientierten verteilten Systemen sind diese jedoch nicht geeignet.

Der Einsatz von RMI erlaubt es, auf den vollen Funktionsumfang von Java zurückzugreifen. Damit können ganze Objekte zusammen mit deren Verhalten über das Netzwerk verschickt werden. Ferner wird die *Garbage Collection* zur *Distributed Garbage Collection* erweitert, und damit können auch entfernte Referenzen aus dem Speicher beseitigt werden, falls diese nicht mehr gebraucht werden. RMI verwendet *JAVA-Interfaces* zur Schnittstellendefinition und dadurch wird der Einsatz von IDL, im Gegenteil zu CORBA, nicht notwendig. Der Einsatz von *RMISocketFactory* ermöglicht es dem *RMI-Transport-Layer* über IP andere Transportprotokolle als TCP anzuwenden.

Stub ist ein lokales Stellvertreterobjekt auf der Clientseite zum Aufruf einer Methode eines Serverobjekts im Auftrag eines Clientobjektes. Aus den Parametern des Methodenaufrufs und der aufgerufenen Methode, sowie der Referenz des Remote-Objektes wird durch die Stub-Methode ein Datenblock gebildet und mittels des *Remote Reference Layer* zum Server gesendet. Auf der Serverseite empfängt der ***Skeleton*** als entferntes Stellvertreterobjekt des eigentlichen Remote-Objektes über den dortigen *Remote Reference Layer* den Datenblock und ruft die gewünschte Methode des Serverobjektes auf. Rückgabewerte werden auf dem analogen Weg zum Clientobjekt zurückgesendet. Eine Beschreibung der RMI-Funktionsweise ist in Abbildung 6.10 dargestellt.

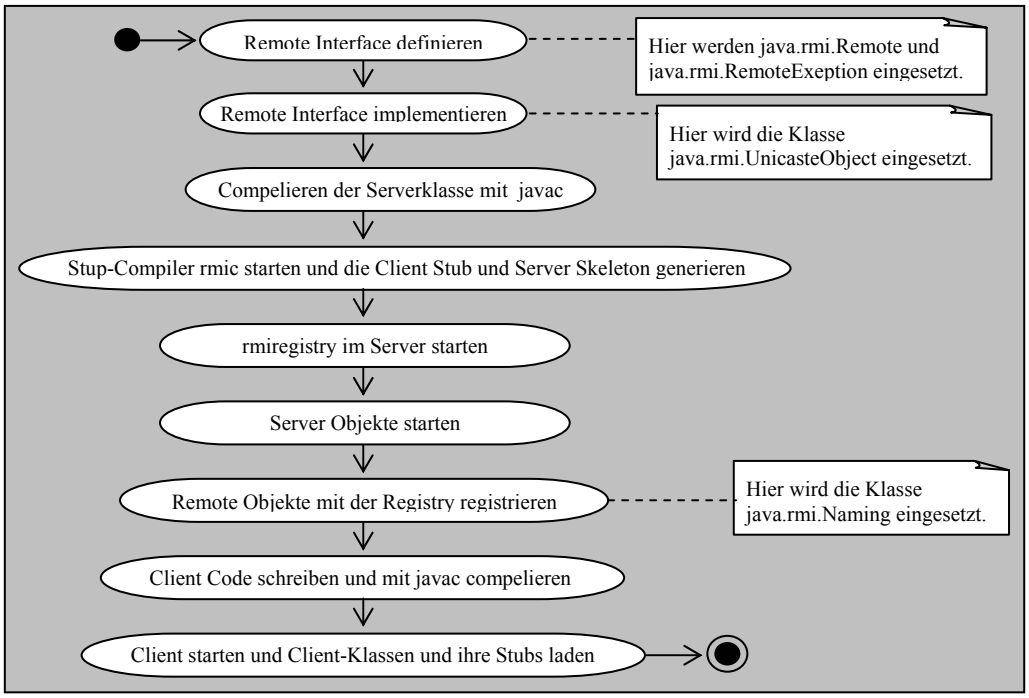


Abb. 6.9: Aktivitätsdiagramm der Funktionsweise von RMI

Die Lokalisierung von entfernten Objekten spielt beim entfernten Methodenaufruf eine bedeutende Rolle. Ein Clientobjekt, welches eine entfernte Methode aufruft, soll zuerst eine Referenz auf das entfernte Objekt erhalten. Diese Referenz wird man normalerweise als Rückgabewert eines Methodenaufrufs erhalten. Eine Schlüsselrolle bei der Lokalisierung von entfernten Objekten in einem RMI System spielt das *rmiregistry*, welches die Rolle eines Nameservers ausübt, und über den Objekte von einem entfernten *Host* geliefert werden können. Basierend auf einem eindeutigen URL *(Uniform Ressource Locator)* bietet die Klasse *java.rmi.Naming* Methoden zur Lokalisierung von entfernten Objekten.

6.4.2 Common Object Request Broker Architecture

Common Object Request Broker Architecture (CORBA) wurde von OMG als geeignete Architektur für die Verteilung und Kooperation Objektorientierter Softwarebausteine in vernetzten, heterogenen Systemen entwickelt. CORBA ist sprach- und plattformunabhängig. Sie stellt die Infrastruktur zur Programmierung von verteilten Anwendungen. Mit CORBA können Anwendungen jedoch nicht programmiert werden. Dazu benötigt man eine Programmiersprache, wie z. B. Java und C++, die von CORBA unterstützt wird. Dass Anwendungen in unterschiedlichen Programmiersprachen geschrieben werden, und diese mit CORBA kommunizieren können, wird als bedeutendste Überlegenheit von CORBA betrachtet. Dazu wird die sprachunabhängige Notation IDL *(Interface Definition Languag)* eingesetzt. Anhand von IDL werden die Objekte und die Aufrufschnittstellen ihrer Methoden definiert, mit denen die Bereitstellung der Dienste des ORB erfolgt. Das IIOP, welches auf dem TCP-IP basiert, wird zur Kommunikation in der CORBA-Umgebung eingesetzt.

Die CORBA-Spezifikation 3.0 beinhaltet Technologien zum Einsatz in den Bereichen Internet, Komponenten und Quality of Service (QoS), wobei diese Bereiche eine Schlüsselrolle bei der Softwareentwicklung spielen. Im Internetbereich, wo Java als Programmiersprache erfolgreich eingesetzt wird, hat sich die Client-Server-Kommunikation auf Firewall-Basis etabliert. Um dieser Herausforderung nachzugehen, bietet CORBA 3.0 eine RMI-IIOP orientierte Abbildung von Java nach IDL sowie eine Firewire Spezifikation. Mit dem Einsatz des RMI-IIOP als Protokoll sind Infrastrukturen von CORBA und RMI zusammengekoppelt. Damit können Objekte anstatt von Referenzen übertragen werden. Neue CORBA-Techniken, wie der interoperable Verzeichnisdienst, das GIOP *(General Inter ORB Protocol)* als Bestandteil der Firewall-Spezifikation und die *Reverse-Mapping* von Java nach IDL, womit Javaprogrammierer CORBA-fähige Clients und Server unter Anwendung der RMI-Programmiertechnik entwickeln können, machen die Nutzung von CORBA für Internet-Entwickler attraktiv.

Das *CORBA Components Model* (CCM) stellt eine serverbasierte Komponententechnologie dar, welche als Alternative zu EJB und COM+ betrachtet werden kann. Die CCM-Spezifikation orientiert sich bezüglich einiger Konzepte, wie z. B. der Zugriff von Clients auf Komponenten, an EJB. In CCM geschieht die Beschreibung der Komponenten in der *Component IDL*, welche als eine Erweiterung der IDL identifiziert wird. CCM unterstützt ein verteiltes Ereignismodell, welches EJB z. Z. nicht kann. CCM unterstützt, ähnlich wie COM, mehrere Eingangs- und Ausgangsschnittstellen je Komponente. Das Konzept des Zugriffs von Clients auf CORBA-Komponenten lehnt sich an EJB an. So werden Komponenten von einem Home administriert. Eine CORBA Komponente ist also immer einem Home zugeordnet. Damit Komponenten eines zugeordneten Typs kreiert werden können, enthält die Schnittstelle eines Home die geeigneten Methoden. Container werden hier in persistente und transierte klassifiziert, wobei zwischen der Komponente und dem Container ein Rahmenwerk mit unterschiedlichen Programmierschnittstellen existiert.

Die Anforderung an die Plattform wird in der QoS-Spezifikation erfasst. Während im Bereich der Konsumgeräte die Minimum CORBA als stark abgespeckte CORBA-Plattform eingesetzt wird, eignet sich Realtime CORBA zum Einsatz auf dem Gebiet der Automatisierungs- und Leitsysteme. Bei Anwendungen, welche hohe Anforderungen bezüglich ihrer Fehlertoleranz erfordern, wie z. B. im Bereich der industriellen Systeme, ist die Fault-Tolerant CORBA geeignet. Durch die CORBA-Messaging-Spezifikation wird im Rahmen der Quality of Service eine asynchrone Kommunikation unterstützt.

6.4.2.1 Die CORBA Architektur

Die gesamte Architektur von CORBA wird bei der OMG als OMA (*Object Management Architecture*) bezeichnet. OMA beinhaltet die Hauptkomponenten *Objekt Request Broker (ORB)*, *Common Objekt Services*, *Common Facilities*, *Applikation Objects* und *Domain Services*, wobei der ORB das Kernstück dieser Architektur bildet [Redl96].

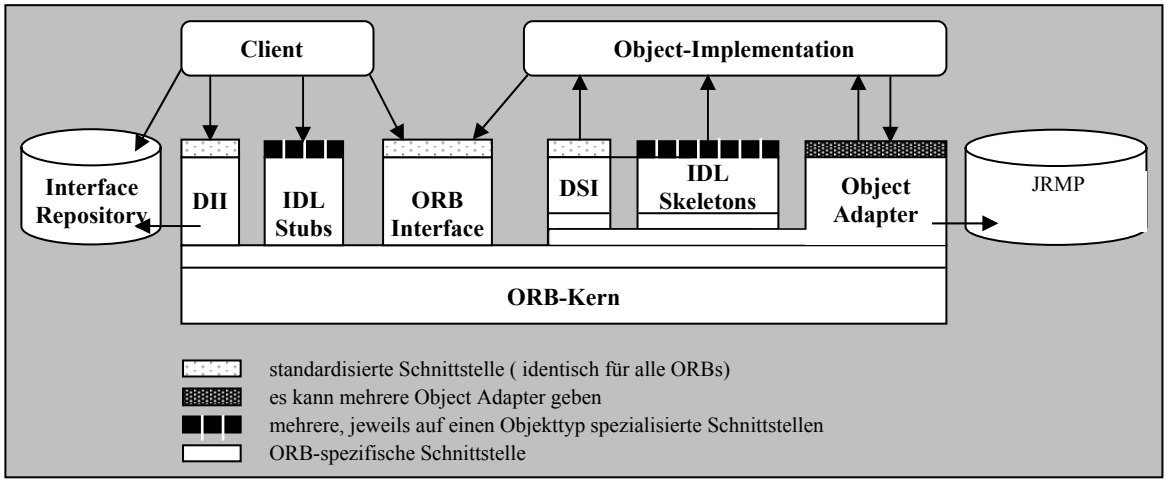


Abb. 6.11: Struktur des ORB

Der ***ORB*** gestattet es den Objekten miteinander zu kommunizieren, unabhängig von ihrem Implementierungsort, und bildet das Kernstück von OMA zur Realisierung der Interoperabilität zwischen den heterogenen Softwarekomponenten in verteilten Systemen. Er übermittelt die Operationsaufrufe von Clients an Objekte im entfernten Server und gibt die Ergebnisse zurück. Die Struktur von ORB ist in Abbildung 6.11 dargestellt.

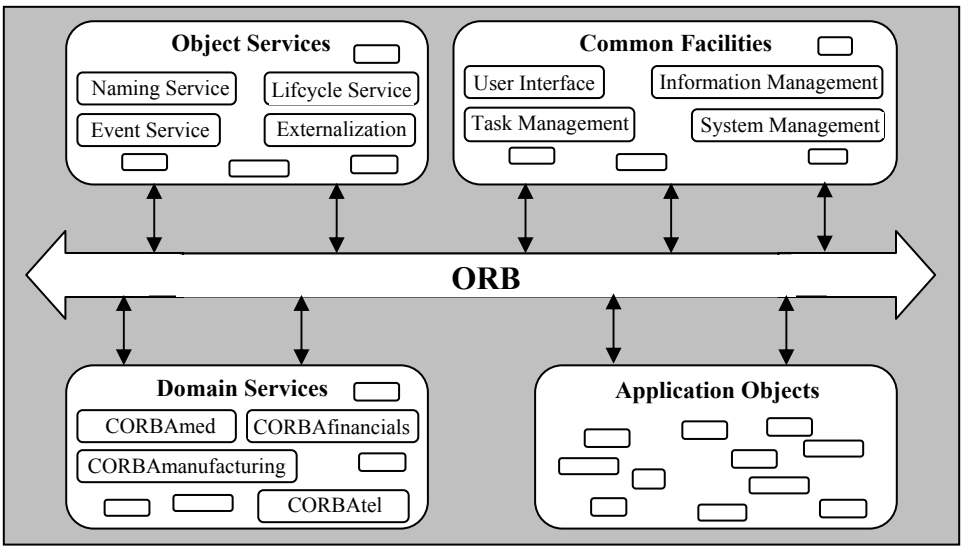


Abb. 6.10: Die OMA-Architektur

Die ***Object Services*** liefern die fundamentalen Operationen sowohl für die logische Modellierung als auch für die physikalische Speicherung von Objekten. Sie sind eine Sammlung von

Diensten mit IDL-definierten Schnittstellen auf Systemebene. Im Unterschied zu den *Objekt Services* leisten die ***Common Facilities*** ihre Dienste als Endbenutzeranwendungen. Sie sind Sammlungen von IDL-definierten Komponenten, die Dienste für den direkten Zugriff auf Applikationsobjekte bereitstellen. Die *Common Facilities* werden in horizontale und vertikale *Facilities* eingeteilt.

Bei ***Domain Services*** handelt es sich überwiegend um eine systematische Darstellung von bereits existierenden Softwarelösungen für bestimmte Branchen in IDL, wie z. B. *CORBAfinance* für das Finanzwesen, *CORBAmed* für das Gesundheitswesen und *CORBAtel* für die Telekommunikation. Um eine effektive Anwendung der verteilten Objektverwaltungssysteme im Bauwesen zu gewährleisten, soll auch das Bauwesen in die *Domain Services* anhand von z. B. *CORBAbc (CORBA building construction)* eingebunden werden.
Applikation Objects sind Komponenten, die auf Endbenutzeranwendungen spezialisiert sind. Diese anwendungsspezifisch entwickelten Schnittstellen unterliegen keiner Standardisierung. In Abbildung 6.10 ist das gesamte OMA Architektur dargestellt.

6.4.2.2 Funktionsweise von CORBA

Die Handlungsweise der CORBA-Architektur ist mit einem klassischen Client/Server-Modell vergleichbar. Er stellt Dienste zur Initialisierung und Auffindung von Zielobjekten und zur Übermittlung vom Methodenaufrufen *(requests)* und deren Resultaten zwischen Clients und Zielobjekten zur Verfügung.

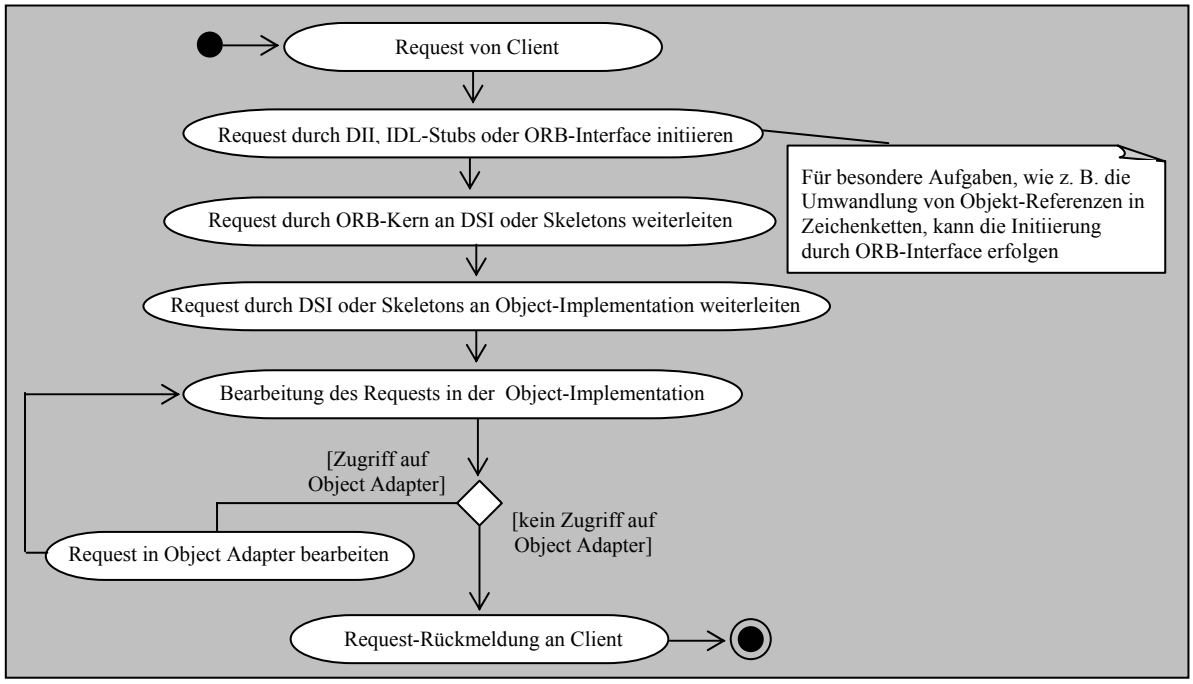


Abb. 6.12: Aktivitätsdiagramm der Kommunikation von verteilten Objekten mit CORBA

Objekte werden durch Clients-Methodenaufrufe auf der Serverseite ausgeführt, wobei auch mehrere Clients durch Austausch einer Objektreferenz auf dasselbe Objekt zugreifen dürfen. Das IIOP, welches auf dem TCP-IP basiert, wird zur Kommunikation in CORBA-Umgebung eingesetzt. In Abbildung 6.12 ist ein Aktivitätsdiagramm zur Beschreibung der Kommunikation von verteilten Objekten mit CORBA gezeigt.

6.4.3 Vergleich von RMI und CORBA als Kommunikationskonzepte

Bei der klassischen Client-Server Architektur werden Anwendungen auf der Client-Seite durchgeführt. Dadurch werden auf der Client-Seite leistungsfähige Hard- und Softwarekomponenten benötigt, um hinreichende Ergebnisse zu erzielen. Eine Unterbringung der Ausführung von Teilen der Applikationslogik auf dem Server kann dieses Problem zwar lösen, führt aber zu Performanceproblemen. Da die Kommunikation zwischen Client und Server über bestimmte Protokolle abgewickelt wird, kann der Einsatz von heterogenen Umgebungen zu Komplikationen führen, weil möglicherweise andere Applikationen ähnliche Protokolle benutzen.
Eine Betrachtung der verschiedenen Architekturen von verteilten Systemen im Hinblick auf die Trennung von Applikations- und Darstellungslogik macht die Vorteile von CORBA deutlicher. So können bei einer CORBA-Umgebung Teile der Applikationslogik auf dem Server ausgeführt werden, während die Kommunikation nach dem standardisierten Protokoll IIOP erfolgt.
Bei der Entwicklung der Java-Technologien wird der Stand der Entwicklung in CORBA zunehmend berücksichtigt. Das *Java Development Kit* (JDK) beinhaltet ab der Version 1.2 integrierte CORBA Unterstützung. Dies ermöglicht die Implementierung von CORBA-Applikationen, ohne Einsatz von Produkten anderer Hersteller. Diese Unterstützung deckt aber nicht das gesamte Spektrum der CORBA-Dienste ab.

RMI ist eine Java-Technologie für reine Java-Anwendungen. Zur Kommunikation mit einem entfernten Objekt wird ein Interface definiert. Dieses Interface implementiert das Serverobjekt und nutzt das Clientobjekt. Der RMI-Compiler (*rmic)* erzeugt aus dem übersetzten Serverobjekt den *Client-Stub* und das *Server-Skeleton*.
Der *Stub* fungiert auf Clientseite als Stellvertreter (Proxy) für das Serverobjekt. Wenn ein Client eine Methode eines entfernten Objektes benötigt, ruft er die zugehörige Methode des *Stub* auf. Beim Server arbeitet der *Skeleton* als Stellvertreter eines Clients.
Damit eine Verbindung zwischen einem Client und einem Serverobjekt zustande kommt, muss sich der Client unter einem eindeutigen Namen registrieren. Dazu wird auf dem Server das Programm *RMI-Registry* eingesetzt.
Dann kann der Client über den registrierten Namen des Serverobjektes eine lokale Instanz des definierten Interfaces erstellen. CORBA funktioniert ähnlich wie RMI, ist jedoch leistungsfähiger und komplexer. Bei CORBA sind *Stub* und *Skeleton* nicht direkt über *Sockets* miteinander verbunden. Ihre Kommunikation wird vielmehr über den ORB realisiert.

Der ORB muss auf allen Plattformen existieren, die verteilte CORBA-Anwendungen realisieren sollen. In CORBA, wie es auch der Fall bei RMI ist, sind Clients und Server per Nameservice miteinander verbunden. Der Nameservice ist in Java IDL integriert und wird mit dem Programm *tnameserv* ausgeführt.

6.5 Darstellung der geeigneten Architektur des verteilten Systems

Mit dem zunehmenden Einsatz und Entwicklung des Internets als globales Kommunikationsmedium, gewinnen verteilte Systeme an Bedeutung. Ein verteiltes Hardwaresystem sowie Koordinationssoftware und verteilte Anwendungen, welche auf dem Hardwaresystem ausgeführt werden sollen, werden als Bestandteile eines verteilten Systems betrachtet.

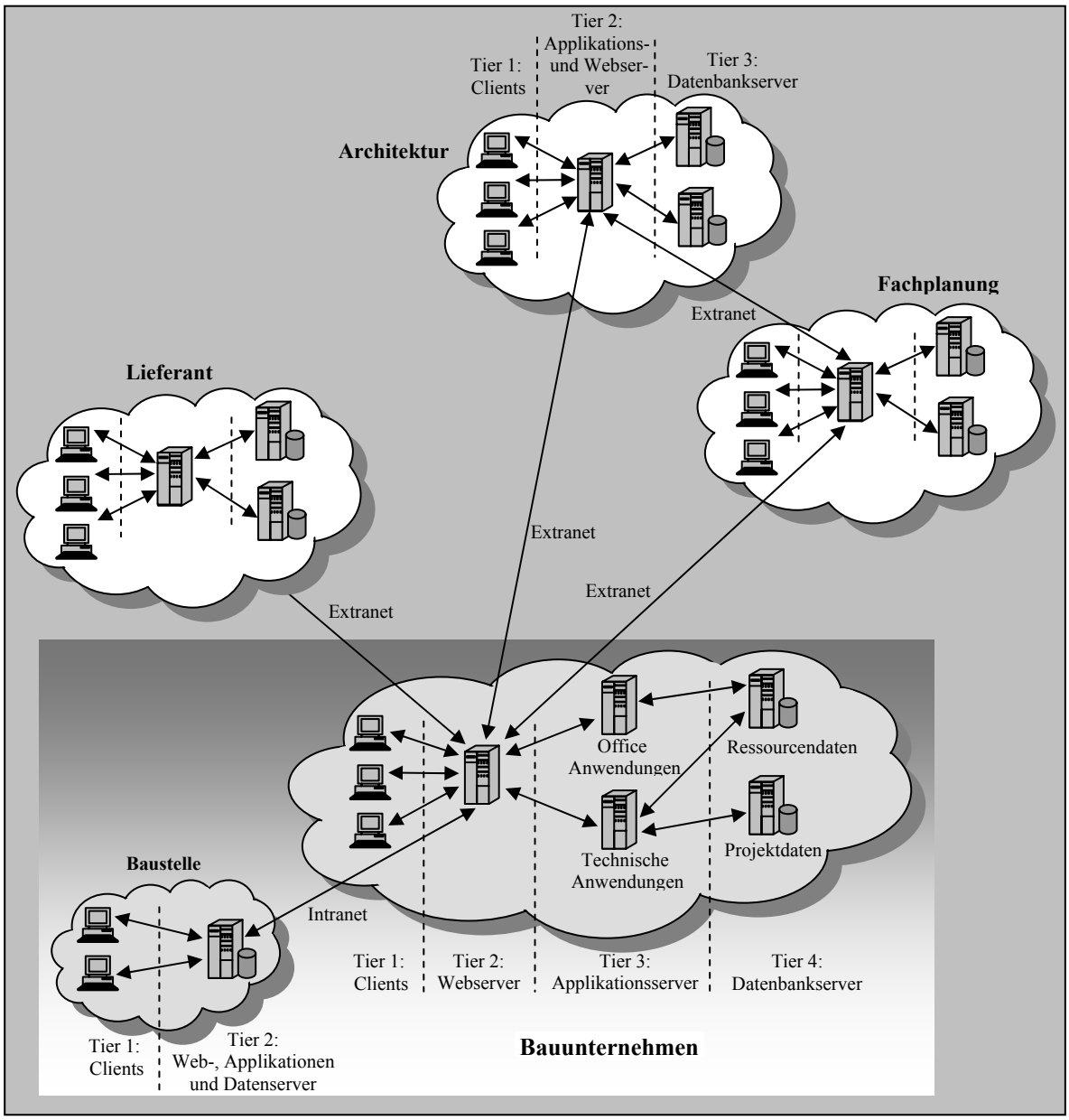


Abb. 6.13: Darstellung der verteilten Architektur im Konstruktiven Ingenieurbau

Die Entwicklungstendenzen im Bereich des Konstruktiven Ingenieurbaus deuten auf verstärkte internetbasierte Ansätze. Diese Tatsache soll auch bei Definition der geeigneten Architektur des Systems berücksichtigt werden. An einem verteilten System werden hauptsächlich folgende Anforderungen gestellt:

- Benutzer-Präsentationsschnittstelle mit Software zum Zugriff auf Server.
- Schnittstelle zum Zugriff auf den Server.
- Ablauflogik der Serverapplikation.
- Persistenz der Daten in Form von Datenbanken.
- Integration der vorhandenen Systeme im Bauunternehmen.
- Integration der Baustelle als ein Teil des Bauunternehmens.
- Schnittstelle zum Zugriff auf das System des Architekten.
- Schnittstelle zum Zugriff auf das System des Fachingenieurs.
- Schnittstelle zum Zugriff auf das System der Lieferanten.

Diese Anforderungen können im Bereich des Bauunternehmens optimal von einer verteilten 4-Tier-Architektur erfüllt werden. In den anderen Bereichen reicht eine verteilte 3-Tier-Architektur. Auf der Baustelle kann eine 2-Tier-Architektur eingesetzt werden, wobei der Applikations- und Datenbankserver als Teil von Tier 3 des Bauunternehmens betrachtet werden soll (Abb. 6.13).

Bei einer 4-Tier-Architektur handelt es sich um eine TCP-basierte Client-Server-Anwendung, wobei auf der Serverseite eine ausgedehnte Aufteilung erfolgt. Bei dieser Aufteilung wird die Serverschnittstelle von den Applikationen getrennt. Die wichtigsten Eigenschaften der 4-Tier-Architektur bestehen darin, dass die Clients nur mit dem Web-Server kommunizieren können. Der Web-Server dient damit ausschließlich zur Übersetzung der Clients-Anfragen im Rahmen von geeigneten Kommunikationsprotokollen wie z. B. HTTP.

Eine 4-Tier-Architektur bringt viele Vorteile auf den Gebieten der Datensicherheit und Skalierbarkeit mit sich.
Die wichtigsten Aspekte der **Datensicherheit** sind die Sicherung der Daten vor unerlaubtem Zugriff und irrtümlicher Vernichtung. In einer 4-Tier-Architektur werden diese Aspekte durch wohldefinierte Schnittstellen effektiver realisiert, womit Clients nur auf den Webserver als Middle-Tier zugreifen können. In der Middle-Tier können Servlets oder Corba-Objekte eingesetzt werden, um den Zugriff der Clients auf Datenbanken über Applikationsserver zu steuern.

Die Aspekte der **Skalierbarkeit** lassen sich durch eine Betrachtung der TCP-Verbindung zwischen Clients und Datenbankserver deutlich zeigen. Während jeder Client eine TCP-Verbindung braucht, falls er einen direkten Zugriff auf Datenbanken hat, wird bei einer Verbindung der Clients mit den Datenbanken durch Middle-Tier nur eine TCP-Verbindung notwendig, nämlich die zwischen Middle-Tier und den Datenbanken. Damit können viele Ressourcen auf dem Datenbankserver gespart werden.

Bei dem Einsatz einer 4-Tier-Architektur kann das Konzept ***Thin Clients*** realisiert werden. *Thin Clients* sind Clients, die nur ein Webbrowser zur Darstellung von Webseiten und zur Ausführung von Applets beinhalten. Dieses Konzept, welches ein wichtiger Schritt zum Ersetzen der Personal Computing durch Network Computing ist, verzichtet auf den Einsatz von Anwendungslogik auf der Clientseite. Damit können Clients wirtschaftlicher ausgestattet werden. Weitere Vorteile liegen im Bereich der Einsparungen bei der Installation und Konfiguration der Soft- und Hardware auf der Clientseite, sowie bei dem Schulungsaufwand auf neue Benutzerschnittstellen.

Auf dem **Webserver** können sowohl *Servlets* als auch JSP eingesetzt werden. *Servlets* sind normale Java-Klassen, welche auf der Serverseite eingesetzt werden, um dynamische Webinhalte anzuzeigen. Sie dienen auch zur Erweiterung der Funktionalität des Webservers durch die Steuerung des Einsatzes von *Requestes*. Zur Ausführung von *Servlets* auf dem Webserver ist eine *Servlet-Engine* notwendig.

JSP baut als Technologie auf *Servlets*, wobei die Funktionsweise dieser beiden Java-Technologien ähnlich ist. Bei der Auswahl des Einsatzes dieser beiden Java-Technologien in den verteilten Systemen im Bereich des Konstruktiven Ingenieurbaus ist zu berücksichtigen, dass *Servlets* bei komplexen Aufgaben, wo der Java-Anteil wichtiger als der HTML-Anteil ist, die bessere Wahl ist. Die Präsentationslogik im Webserver kann mit JSP realisiert werden, wobei *Servlets* parallel dazu eingesetzt werden können, um *Requestes* zu steuern.

Grundsätzlich werden zur Bearbeitung der Aufgabenstellungen im Bauunternehmen Anwendungen auf **Applikationsservern** angeboten. Bei dem Einsatz von J2EE-Plattform-basierten Applikationsservern werden EJB innerhalb von geeigneten Containern laufen. Hier werden auch geeignete APIs wie die *J2EE Connector Architecture* zur Integration von vorhandenen Softwaresystemen im Bereich des Konstruktiven Ingenieurbaus angeboten.

Auf **Datenbankservern** werden sämtliche Daten des Bauunternehmens wie z. B die Ressourcen, die bei den Bauprojekten eingesetzt werden können, sowie die notwendigen Daten zur Beschreibung der Bauobjekte bereitgestellt.

6.6 Modellierung der Verteilung und Kommunikation des Systems

Bei dem Entwurf der Verteilung und Kommunikation des Systems werden u. a. solche Aspekte wie die Organisation und Abhängigkeiten von Komponenten sowie deren Verteilungs- und Kommunikationsstruktur behandelt. Hier werden die Softwarekomponenten, ihre Beziehungen sowohl untereinander als auch zu einzelnen Knoten einer Hardwaretopologie beschrieben. Zur Modellierung dieser Aspekte bietet die UML die Komponentendiagramme *(component diagrams)* und Verteilungsdiagramme *(deployment diagrams)* als Konzepte an. Diese Diagramme erlangen ihre Bedeutung analog mit dem wachsenden Einsatz der verteilten Anwendungen. Zunächst ist es notwendig, eine geeignete Architektur des verteilten Systems festzulegen und dann die Software-Komponenten innerhalb dieses Systems zu verteilen.

6.6.1 Komponentendiagramme

Eine Komponente *(component)* im Sinne der UML ist ein Softwaremodul mit eigener Identität und wohldefinierten Schnittstellen. So handelt es sich bei einer Komponente um ein physisches, vertauschbares Element eines Systems, welches u. a. zur Anpassung und Realisierung von Schnittstellen dient. Grundsätzlich gliedern sich Komponenten in Quellecode-Komponenten, Binärcode-Komponenten und ausführbare Komponenten. Quellecode-Komponenten sind Klassenspezifikationen und Klassenimplementierungen, die nur in der Übersetzungszeit relevant sind, sie werden mit dem Stereotyp *<<file>>* beschildert. Binärcode-Komponenten werden zur Bindung der Programme eingesetzt. Als ausführbare Komponenten werden ausführbarer Code, Bibliotheken, Datenbanktabellen und dynamisch generierte Webseiten betrachtet. Mit den Komponenten als Konzept kann die komponentenorientierte Softwareentwicklung realisiert werden. Eine Komponente kann also durch eine andere Komponente ersetzt werden, wenn sie dieselbe Schnittstelle präsentiert. In UML kann eine Komponente nur als Container für Softwarebausteine betrachten werden. Das Komponenten-Konzept von UML lässt sich in Java durch *JavaBeans* auf der Clientseite und *Enterprise JavaBeans* (EJB) auf der Serverseite implementieren.

Die Begriffe Komponenten sowie komponentenbasierte Softwareentwicklung sind eigentlich noch nicht einheitlich definiert. Die Spezifikationen der Komponentenmodelle wie *JavaBeans* und *Enterprise JavaBeans* von Java, COM von Microsoft sowie das *CORBA Components Model (CCM)* von OMG weisen zwar im allgemeinen Ähnlichkeiten auf, aber im Detail unterscheiden sie sich voneinander. Grundsätzlich kann eine Komponente als ein eingekapselter, ablauffähiger Softwarebaustein mit bestimmter Funktionalität zur Durchführung von bestimmten Anwendungsaufgaben betrachtet werden.

In Hinsicht auf Client-Server-Architektur unterscheidet man zwischen clientseitigen Komponentenkonzepten wie JavaBeans von Java und ActiveX-Controls von Microsoft und serverseitigen Komponentenkonzepten wie EJB CCM. Mit den Komponentenmodellen JavaBeans und Enterprise JavaBeans von Java können plattformunabhängige Komponenten entwickelt werden. In Java entsteht grundsätzlich eine Komponente, welche mehr Selbstständigkeit als ein Objekt hat, durch das Zusammenwirken von mehreren Objekten. Bezüglich der Sprachgrundform besteht in Java keine Differenz zwischen Komponenten und Objekten.

Komponentendiagramme präsentieren die Organisation und Abhängigkeiten von Komponenten als Softwaremodule. So werden Komponentendiagramme eingesetzt, um Komponenten des Anwendungssystems mit ihren Interfaces und ihren Kompilierungs- und Laufzeitabhängigkeiten zu beschreiben. Grundsätzlich wird ein Komponentendiagramm als Graph dargestellt. Die Knoten dieses Graphs sind die Komponenten und die Kanten sind die Beziehungen zwischen diesen Knoten.

Eine Gegenüberstellung der Objektorientierten Programmierung (OOP) und der komponentenbasierten Softwareentwicklung führt zum Ergebnis, dass diese nicht auf einem theoretischen Modell aufbaut, wie es bei der OOP der Fall ist. Die komponentenbasierte Softwareentwicklung ist eher das Ergebnis von pragmatischen Überlegungen, nach denen man übersichtliche und gesonderte Softwarebausteine entwickelt, die danach durch ihre wohldefinierte Schnittstellen, zu einer Anwendung integriert werden können. Bezüglich ihrer Einordnung als Technologie baut die komponentenbasierte Softwareentwicklung auf die Objektorientierung auf. Während die komponentenbasierte Softwareentwicklung auf Aspekte der Softwareorganisation beschränkt, deckt die Objektorientierung das gesamte Spektrum der Softwareentwicklung, von der Analyse und Entwurf bis zur Implementierung. Die komponentenbasierte Softwareentwicklung ermöglicht eine effiziente, kostengünstige und rasante Softwareentwicklung. Da Komponenten einen erheblichen Entwicklungsaufwand benötigen, werden sie hauptsächlich entwickelt, um sie zu einem späteren Zeitpunkt mehrfach einzusetzen.

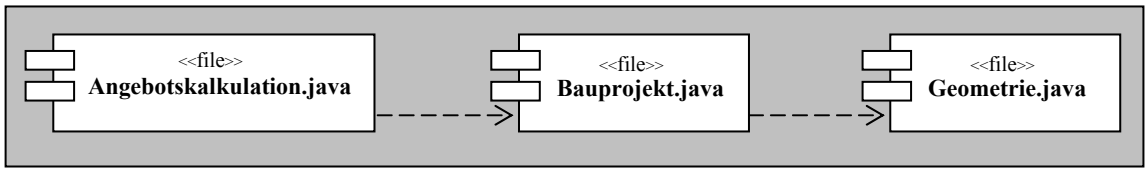


Abb. 6.14: Komponentendiagramm der Quellcode-Komponenten des Systems der Angebotskalkulation

Zur Verbindung von Komponenten kann der Austausch von Ereignissen, sowie der direkte Aufruf von Methoden eingesetzt werden. Auch der Infobus und das Container-Konzept als Java Technologien sowie Adapterklassen und Skriptsprachen können als Verbindung zwischen den Komponenten angewendet werden. JNDI als Namens- und Verzeichnisdienst in Java-Umgebung kann auch zur Verbindung zwischen Komponenten eingesetzt werden, wobei Namens- und Verzeichnisdienste hauptsächlich zur Ermittlung der Kommunikationspartner verwendet werden. Frameworks können Infrastruktur für Komponenten anbieten. Damit wird innerhalb dieser Frameworks die gemeinsame Semantik für die Komponenten festgelegt.

In Abbildung 6.14 ist das Komponentendiagramm des Systems der Angebotskalkulation dargestellt, wobei hier nur die Quellcodekomponenten mit ihren Übersetzungsabhängigkeiten gezeigt werden.

6.6.2 Verteilungsdiagramme

Ein Verteilungsdiagramm beschreibt die Abhängigkeiten zwischen den Soft- und Hardwarekomponenten eines Systems. Generell besteht dieses Diagramm aus Knoten, die Hardwareverarbeitungseinheiten darstellen, sowie Kanten, die Kommunikationspfade repräsentieren. Bei einem Knoten eines Verteilungsdiagramms handelt es sich um ein zur Laufzeit konkretes physisches Teil zur Beschreibung einer Rechnerressource.

Ein Knoten hat zumindest einen Speicher und häufig auch Rechenkapazität. Zwischen diesen Knoten, die in UML als Quader dargestellt werden, bestehen Verbindungen. Bei diesen Verbindungen, die als Linien gezeigt werden, handelt es sich um physikalische Kommunikationspfade, die Informationen über das verwendete Protokoll repräsentieren können.

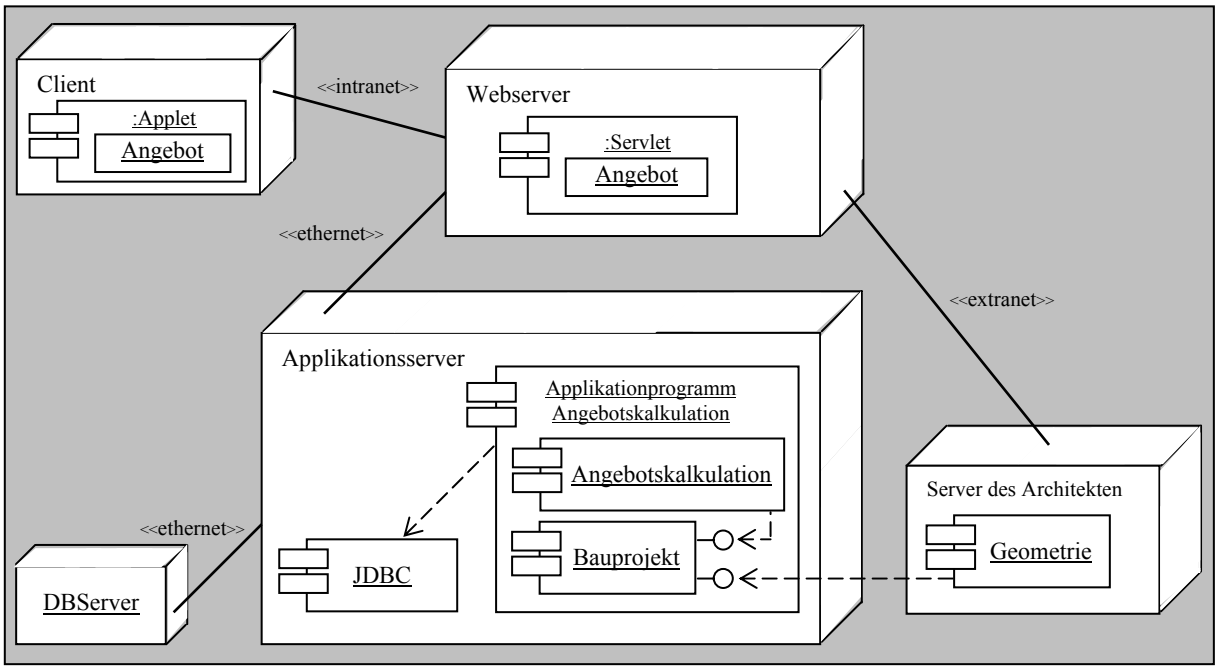


Abb. 6.15: Verteilungsdiagramms des Systems der Angebotskalkulation

Verteilungsdiagramme werden hauptsächlich erstellt, um anzuzeigen, welche Software auf welcher Hardware ausgeführt werden kann. Dieses Ziel wird nur dann erreicht, wenn die Komponenten- und Verteilungsdiagramme in einem Diagramm dargestellt werden. In Abbildung 6.15 ist das Verteilungsdiagramm des Systems der Angebotskalkulation im Konstruktiven Ingenieurbau dargestellt. In dieser Abbildung wird gezeigt, dass im Rahmen der Verteilung, welche auf eine Multi-Tier-Architektur basiert, die Geometrie als Komponente im Server des Architekten ausgeführt wird.

7 Implementierung des Systems der Kosten in Bauprojekten

In der Implementierungsphase wird das Entwurfsmodell des Systems in die gewählte Programmiersprache Java übersetzt. Die Objekte des Systems werden damit als Java-Objekte implementiert. Das Verfahren zur Implementierung des Systems der Kosten in Bauprojekten gliedert sich hauptsächlich in vier Schritte. Während der erste Schritt die Implementierung des statischen Strukturmodells umfasst, spezifiziert der zweite Schritt die Auswahl und Definition des Persistenzkonzeptes. Der dritte und vierte Schritt befasst sich mit der Anfertigung der *User-Interfaces* bzw. der Codierung der Dynamik des Systems. Falls ein Objektorientiertes Datenbanksystem als Plattform zur Realisierung der Persistenz des Systems eingesetzt wird, sollen die ersten beiden Schritte zusammenfließen.

Bei der Implementierung des Systems der Kosten in Bauprojekten in einer Umgebung eines Objektorientierten Datenbanksystems ist darauf zu achten, dass diese Implementierung mit einer engen Verbindung mit der verwendeten Programmiersprache Java (als ausgewählte Implementierungssprache in der vorliegenden Arbeit) erfolgen soll. In diesem Zusammenhang wird zuerst eine Schemadeklaration definiert. Zur Schemadeklaration kann die ODL *(Object Definition Language)* oder Java-ODL verwendet werden [Balz99]. Ein Präprozessor wird dann verwendet, um diese Schemadeklaration in das Datenbankschema und in ein Java-Deklarationsteil zu übersetzten. Dann wird das Anwendungsprogramm in Java erstellt, wobei hier Klassenbibliotheken zur Manipulation der Datenbank eingesetzt werden. Zur Erstellung einer lauffähigen Anwendung werden der Java-Deklarationsteil und das Anwendungsquellprogramm übersetzt und mit dem Datenbanksystem eingebunden.

7.1 Umsetzung der UML-Konzepte in Java

Nachfolgend wird die Umsetzung einiger UML-Konzepte in Java erläutert. Bei diesen Konzepten handelt es sich um solche, die in der vorliegenden Arbeit wiederholt auftreten.

- **Generalisierung**
  Das Konzept der Generalisierung in UML kann in Java durch die Vererbung und anhand des Schlüsselwortes *extends* umgesetzt werden. Ist eine Klasse ohne *extends*-Klausel deklariert, bedeutet dies, dass sie direkt aus der Java-Klasse `Object` abgeleitet wird. Diese Klasse, welche als Superklasse aller anderen Klassen in Java betrachtet wird, deklariert Methoden, die für alle anderen Java-Klassen gültig sind.
- **Assoziationen**
  Java bietet die Möglichkeit zur Implementierung von Assoziationen zwischen Klassen, durch die Objektreferenz. Diese Implementierung erfolgt durch die Vereinbarung von neuen Referenzattributen in den Klassen, in denen die Assoziationen enden. Der Typ eines Referenzattributes ist die Zielklasse und sein Name ist entweder der Name der Rolle oder der Name der Assoziation.
  Aggregation und **Komposition** sind in UML gerichtete Assoziationen. Sie können anhand von Containerklassen wie Vector implementiert werden.
- **Schnittstellen (*Interfaces*)**
  Das *Interface*-Konzept in UML kann in Java anhand der Schlüsselwörter *Interface* und *implements* umgesetzt werden. Eine Klasse in Java kann mehrere *Interfaces* implementieren.
- **Pakete**
  Das UML-Konzept Paket *(Package)* kann durch das Java-Konzept *Package* umgesetzt werden.

7.2 Implementierung des Strukturmodells

Im Strukturmodell sind die einzelnen Klassen und ihre Beziehungen zueinander spezifiziert worden. In der Implementierungsphase dient das verfeinerte Klassendiagramm des Strukturmodells der Angebotskalkulation der Entwurfsphase als Grundlage zur Erstellung des Codes in Java. Dieser Schritt der Implementierung kann auch durch UML-taugliche CASE-Werkzeuge wie z. B. Rational Rose realisiert werden. Nachfolgend ist der Java-Code dargestellt:

Hier ist der Code für die Klasse Bauprojekt:

```
Import java.util.*;

public class Bauprojekt
{

  // Attribute
  public int Nummer;
  public String Bezeichnung;
  public Date Baubeginn;
  public Date Bauende;
  public double Kosten;

  // Referenzattribute
  Public Vector Bauprojektkomponenten;
  Public Angebotsleistungsverzeichnis Angebotsleistungsverzeichnis;
  Public Bauherr Bauherr;
  Public Angebot Angebot;

  // Methoden
  public Bauprjekt () {
  Bauprojektkomponenten = new Vector ();
  }
  public double Kosten () {
  double Summe = 0;
    for (int i = 0; i < Bauprojektkomponenten.size (); i++ )
        {
        Bauprojektkomponente K = (Bauprojektkomponente)
Bauprojektkomponenten.elementAt (i);
        Summe = Summe + K.Kosten ();
         }
         return Summe;
      }
     public void zustandAktualisieren ()
     public void zustandAnzeigen ()
     public void fertigBringen ()
     }
```

Hier ist der Code für die Klasse Bauprojektkomponente:

```
public abstract class Bauprojektkomponente
{

  // Attribute
  public int Nummer;
  public String Bezeichnung;
  public double Kosten;

  // Referenzattribute
  public Bauprojekt Bauprojekt;
  public Bauprodukt Bauprodukt;
  public Ressource Ressource ;

 //  Methoden
  public void hinzufügen ()
  public void entfernen ()
  public void Kosten ()
}
```

Hier ist der Code für die Klasse Baugrundstück:

```
Import java.util.*;

public class Baugrundstück extends Bauprojektkomponente
{

  // Attribute
  public String Adresse;

  // Referenzattribute
  public Bauprojekt Bauprojekt;
  public Vector Bauprodukte;
  public Geometrie Geometrie;

  // Methoden
  public Baugrundstück ()
  {
  Bauprodukte = new Vector ();
  }
  public Baugrundstück Kosten (double k)
  {
  Kosten = k;
  }
  public double Kosten ()
  }
  return Kosten;
  }
}
```

Hier ist der Code für die Klasse Bauprodukt:

```
Import java.util.*;

public class Bauprodukt extends Bauprojektkomponente
{

  // Attribute
  public String Zustand;

 // Referenzattribute
  public Vector Bauprojektkomponenten;
  public Baugrundstück Baugrundstück;
  public Vector Ressourcen;
  public Geometrie Geometrie;
  Public Leistungskostenposition Leistungskostenposition;

  // Methoden
  public Bauprodukt ()
  {
  Bauprojektkomponenten = new Vector ();
  Ressourcen = new Vector ();
  }
  public double Kosten ()
   {
  double Summe = 0;
     for (int i = 0; i < Bauprojektkomponenten.size (); i++ )
          {
          Bauprojektkomponente K = (Bauprojektkomponente)
          Bauprojektkomponenten.elementAt (i);
          Summe = Summe + K.Kosten ();
           }
          return Summe;
    }
  public hinzufügen (Bauprojektkomponente);
  {
  Bauprojektkomponenten.addElement (K);
  }
  public entfernen (Bauprojektkomponente);
  {
  Bauprojektkomponenten.removeElement (K);
  }
}
```

Hier ist der Code für die Klasse Geometrie:

```
public class Geometrie
{

  // Attribute
  public int kordinaten;

  // Referenzattribute
  public Bauprodukt Bauprodukt;
  public Baugrundstück Baugrundstück;
```

```
  // Methoden
  public zeichnen ()
  public anzeigen ()
  public verschieben ()
}
```

Hier ist der Code für die Klasse Ressource:

```
Import java.util.*;

public class ressource extends Bauprojektkomponente
{

  // Attribute
  public String Vorrat;

  // Referenzattribute
  public Vector Bauprojektkomponenten;
  public Bauprodukt Bauprodukt;
  public Leistungskostenliste Leistungskostenliste;
  public Leistungskostenposition Leistungskostenposition;

  // Methoden
  public Ressource ()
  {
  Bauprojektkomponenten = new Vector ();
  }
  public double Kosten ()
  {
  double Summe = 0;
    for (int i = 0; i < Bauprojektkomponenten.size (); i++ )
        {
        Bauprojektkomponente K = (Bauprojektkomponente)
        Bauprojektkomponenten.elementAt (i);
        Summe = Summe + K.Kosten ();
         }
         return Summe;
    }
  public hinzufügen (Bauprojektkomponente);
  {
  Bauprojektkomponenten.addElement (K);
  }
  public entfernen (Bauprojektkomponente);
  {
  Bauprojektkomponenten.removeElement (K);
  }
  public void vorratPrüfen ()
}
```

Hier ist der Code für die Klasse Bauherr:

```
Import java.util.*;

public class Bauherr
{

  // Attribute
  public String Name;
  public String Adresse;

  // Referenzattribute
  Public Angebot Angebot;
  Public Vector Bauprojekte;

  // Methoden
  public Bauherr ()
  {
   Bauprojekte = new Vector ();
   }
   public void anzeigen ()
   public void informieren ()
}
```

Hier ist der Code für die Klasse Angebotsleistungsverzeichnis:

```
Import java.util.*;

public class Angebotsleistungsverzeichnis
{

  // Attribute
  public int Nummer;
  public String Bezeichnung;
  public Date Datum;

  // Referenzattribute
  public Bauprojekt Bauprojekt;
  public Vector Positionen;
  public Leistungskostenliste Leistungskostenliste;

  // Methoden
  public Angebotsleistungsverzeichnis ()
  {
  Positionen = new Vector ();
  }
  public void hinzufügen ()
  public void entfernen ()
  public void anzeigen ()
}
```

Hier ist der Code für die Klasse Angebotsleistungsverzeichnisposition:

```
public class Angebotsleistungsverzeichnisposition
{

  // Attribute
   public String Positionsnummer;
   public String Titel;
   public String Kurzbezeichnung;
   public String Langbezeichnung;
   public double Menge;
   public String Einheit;

  // Referenzattribute
   public Angebotsleistungsverzeichnis Angebotsleistungsverzeichnis;
   public Leistungskostenposition Leistungskostenposition;
}
```

Hier ist der Code für die Klasse Leistungskostenliste:

```
Import java.util.*;

public class Leistungskostenliste
{

  // Attribute
   public int Nummer;
   public String Bezeichnung;
   public double Mengenkostensumme,
   public String Währung;
   public Date Datum;

  // Referenzattribute
   public Angebotsleistungsverzeichnis Angebotsleistungsverzeichnis;
   public Angebot Angebot;
   public Vector Positionen;


 // Methoden
   public Leistungskostenliste ()
   {
   Positionen = new Vector ();
   }
   public void hinzufügen ()
   public void entfernen ()
   public void anzeigen ()
}
```

Hier ist der Code für die Klasse Leistungskostenposition:

```
public class Leistungskostenposition
{

  // Attribute
  public String Lfd.Nr.desLV;
  public String KurzBezeichnung;
  public String LangBezeichnung;
  public double Menge;
  public double StundenansätzeJeEinheit;
  public double StoffansätzeJeEinheit;
  public double FremdleistungsansätzeJeEinheit;
  public double StundenansätzeJePosition;
  public double StoffansätzeJePosition;
  public double FremdleistungsansätzeJePosition;
  public String Währung;
  public double StundenkostenJeEinheit;
  public double StoffkostenJeEinheit;
  public double FremdleistungskostenJeEinheit;
  public double Einheitskosten;
  public double MengenkostenJePosition;
  public double StundenkostenMitZuschlagJeEinheit;
  public double StoffkostenMitZuschlagJeEinheit;
  public double FremdleistungskostenMitZuschlagJeEinheit;
  public double AngebotspreisJeEinheit;
  public double AngebotspreisJePosition;

  // Referenzattribute
  public Angebotsleistungsverzeichnisposition Angebotsleistungsverzeichnisposition;
  public Leistungskostenliste Leistungskostenliste;
  public Angebotsposition Angebotsposition;
  public Vector Ressourcen;
  public Vector Bauprodukte;
}
```

Hier ist der Code für die Klasse Angebot:

```
Import java.util.*;

public class Angebot
{

  // Attribute
  public int Nummer;
  public String Bezeichnung;
  public double Angebotssumme;
  public String Währung;
  public Date Datum;

  // Referenzattribute
  public Leistungskostenliste Leistungskostenliste;
  public Bauherr Bauherr;
  public Bauprojekt Bauprojekt;
```

```
   Public Vector Positionen;

   // Methoden
   public Angebot ()
   {
   Positionen = new Vector ();
   }
   public void hinzufügen ()
   public void entfernen ()
   public void anzeigen ()
}
```

Hier ist der Code für die Klasse Angebotsposition:

```
public class Angebotsposition
{

   // Attribute
   public String Lfd.Nr.desLV;
   public String KurzBezeichnung;
   public double Menge;
   public String Währung;
   public double AngebotspreisJeEinheit;
   public double AngebotspreisJePosition;

   // Referenzattribute
   public Angebot Angebot;
   public Leistungskostenposition Leistungskostenposition;
}
```

7.3 Definition und Realisierung der Persistenzmechanismen

Bei der Definition und Realisierung der Persistenzmechanismen soll zuerst festgelegt werden, welche Objekte des Systems der Kosten in Bauprojekten im Konstruktiven Ingenieurbau persistent gehalten werden. Es ist von großer Bedeutung, solche Daten über Kosten von Leistungen persistent zu halten, um sie später bei der Erstellung von Angebotskalkulationen als vergleichbare Beispiele zu verwenden und damit wettbewerbsfähige Angebote zu erstellen.

Bei der Definition der Art der Persistenzmechanismen geht es in erster Stelle darum, welches Datenmodell und Datenbankmanagementsystem zur Realisierung der Persistenz geeignet ist. Grundsätzlich ist bei der Realisierung der Persistenz von Objekten auf die Gewährleistung der Objektidentität und die korrekte Abbildung von komplexen Beziehungen zwischen den Objekten des Systems wie z. B. die Aggregation und die Vererbung zu achten. Daher bilden Objektorientierte Datenbanken eine erste Wahl zur Realisierung der Persistenz der Objekte des Systems der Kosten in Bauprojekten im Konstruktiven Ingenieurbau. In Abbildung 7.1 ist das Strukturmodell des Systems im Hinblick auf die Realisierung der Persistenz mit Datenbanken dargestellt. Diese Abbildung zeigt deutlich, dass bei dem Einsatz von Objektorientierten Datenbanken als Plattform zur Realisierung der Persistenz keine Strukturverluste hingenommen werden müssen.

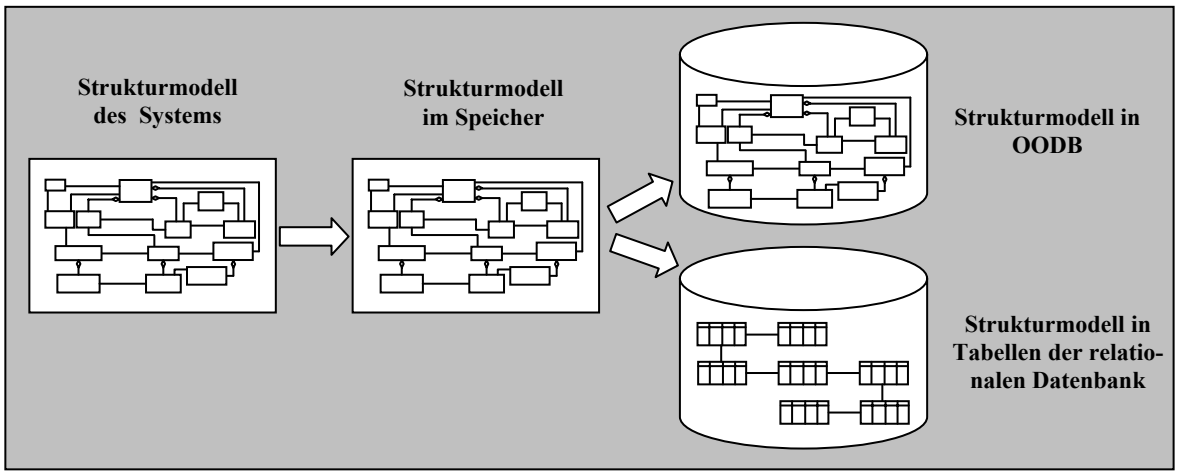


Abb. 7.1: Transformation des Strukturmodells des Systems bei der Realisierung der Persistenz

In Objektorientierten Datenbanksystemen wie z. B. ObjectStore, können Objektstrukturen ohne Strukturverluste persistent gehalten werden. Mit ObjectStore, welches als Objektorientiertes Datenbanksystem über eine Client-Server-Architektur verfügt, wird die Datenhaltung durch eine objektabhängige Persistenz realisiert. Während die klassenabhängige Persistenz alle Objekte unterhalb einer persistenten Wurzelklasse persistent macht, können bei dem Einsatz der objektabhängigen Persistenz persistente und transiente Objekte in einer Klasse vorhanden sein. Mit ObjectStore kann die Persistenz der Objekte unabhängig vom Datentyp realisiert werden. Dem ObjectStore liegt kein eigenes Datenmodell zugrunde und wird deshalb als eine Erweiterung des Konzeptes der Objektorientierten Programmiersprachen betrachtet [Voss99] [Heue97].
ODMG als Standard legt nicht fest, wie die Realisierung der Persistenz einer Klasse erfolgen soll. ObjectStore realisiert die Persistenz auf der Basis eines Postprozessors, welcher vorher

kompilierte Klassen persistent machen kann. Die Java-Anbindung von ObjectStore basiert auf dem Postprozessor *osjcfp*, der den Bytecode einer Java-Klasse so verändert, dass diese Klasse persistenzfähig wird [Saak00].

Bei dem Einsatz der Objektorientierten Technologie, wie z. B. Java-Umgebung, existieren Objekte üblicherweise nur solange, wie das Programm ausgeführt wird. Als simpelste Möglichkeit zur Realisierung der Persistenz in einer Java-Umgebung gestattet Serialisierung das Speichern und Wiederherstellen von komplexen Objekten bzw. Objekt-Hierarchien. Dieses Konzept wird in Java durch *Streams* realisiert. Damit können Java-Objekte in Dateien gespeichert oder über das Netz versendet werden. So kann die Serialisierung in Java als Konzept sowohl zur Realisierung der Persistenz als auch zur Datenübertragung im Netzwerk betrachtet werden. Applets können auch serialisiert werden, wobei sie vor der Serialisierung gestoppt werden müssen. Diese können später in den Browser geladen und neu gestartet werden. Persistenz soll aber nicht mit dem Begriff Serialisierung gleichgestellt werden. Schließlich handelt es sich bei dem Begriff Persistenz um das dauerhafte Speichern von Daten durch Datenbanken auf externe Datenträger.

In [Takl00] ist ein Konzept eines generischen Persistenz-Frameworks dargestellt, welches im Zusammenspiel mit einem leistungsfähigen Code-Generator eine allgemeine, verteilte Datenbankanbindung realisiert. Als eine experimentelle persistente Programmierumgebung für Java ist PJama in [Boge99] dargestellt. Sie realisiert die Prinzipien der orthogonalen Persistenz, wobei jeder Datentyp ohne Transformation in ein anderes Datenformat persistent gemacht und im Hauptspeicher wieder geladen werden kann.
Komponentenmodelle besitzen Mechanismen, mit denen die Persistenz der Komponenten realisiert wird. So können persistente Komponenten in *JavaBeans* anhand der Serialisierung erzeugt werden. Zur Realisierung der Persistenz von Anwendungsobjekten in EJB-basierten Applikationsservern wird das Konzept Entity-Beans, als Bestandteil der Spezifikation von EJB eingesetzt.

Objektrelationale Datenbanksysteme können auch zur Realisierung der Persistenz der Objekte des Systems der Kosten in Bauprojekten eingesetzt werden. Oracle 8, dessen Basis ein objektrelationales Datenmodell bildet, kann als Plattform zu diesem Zweck verwendet werden. Hier sollen aber zuerst die Java-Objekte auf Oracle 8-Objekte abgebildet werden. Die von Oracle angebotenen JDBC-Treiber können eingesetzt werden, um von der objektrelationalen Unterstützung des Datenmodells Gebrauch zu machen und damit Java-Objekte ohne Performanzverluste in geschachtelten Tabellen in Oracle8 darzustellen.

Bei der Festlegung der Art der eingesetzten Persistenzmechanismen sollen aber an erster Stelle die bereits vorhandenen Datenbanksysteme, sowie die Einbeziehung von Legacy-Systemen als entscheidende Faktoren berücksichtigt werden. In der Praxis, insbesondere im kommerziellen Bereich, werden eigentlich immer noch relationale Datenbanken favorisiert. Da das System der Kosten in Bauprojekten ähnliche Strukturen wie Systeme des kommerziellen Bereichs aufweist, wird anhand eines Anwendungsbeispiels in Kapitel 8 die Persistenz durch relationale Datenbanken realisiert. Bei dem Einsatz eines relationalen Datenbanksystems als Plattform zu Realisierung der Persistenz wird das Strukturmodell für eine objekt-relationale Abbildung verwendet. Mit dieser Abbildung können einzelne Klassen des Strukturmodells in Tabellen der Datenbank umgewandelt werden.

8 Anwendungsbeispiele

8.1 Anwendungsbeispiel 1: Realisierung der Datenhaltung des Objektmodells

In diesem Anwendungsbeispiel wird gezeigt, wie die Ergebnisse der Objektorientierten Modellierung bei der Realisierung der Datenhaltung durch relationale Datenbanken verwendet werden können. Dabei wird die Abbildung der Objekte in Tabellen der relationalen Datenbank dargestellt. In Rahmen dieses Anwendungsbeispiels wird die Modellierung der Angebotskalkulation für ein Parkhaus als Bauprojekt dargestellt. Dieses Anwendungsbeispiel bezieht sich auf ein Beispiel in [Pran95] und wird in der vorliegenden Arbeit entsprechend dem entwickelten Strukturmodell der Entwurfsphase modifiziert.

8.1.1 Darstellung des Bauprojektes

Die Skizzen des Bauprojektes Parkhaus und das Angebotsleistungsverzeichnis sind in [Pran95] detailliert dargestellt. Das Angebotsleistungsverzeichnis ist folgendermaßen gegliedert:

Leistungsbeschreibung mit Leistungsverzeichnis

Titel I Erdarbeiten

Pos. 1.1
Erdaushub für die Baugrube, Bodenklasse 4, von OK.-Fundamente bis OK.-Gelände, bis zu einer Maximaltiefe von ca. 8,00 m. Aushub lösen und laden.
15.000 m³

Pos. 1.2
Aushub für Einzel- und Streifenfundamente, Tiefe bis zu ca. 1,00 m unter Baugrubensohle, sonst wie Pos. 1.1. Der Aushub ist hier besonders maßhaltig herzustellen, weil keine Fundamentschalung vergütet wird.
320 m³

Pos. 1.3
Abfuhr des gesamten Bodenmaterials auf eine ca. 3 km entfernt liegende Kippe, einschließlich Auffüllgebühren.
15.320 m³

Pos. 1.4
Liefern von verdichtungsfähigem Material und Lagenweiser Einbau. Maximalschichtdicke 30 cm, einschließlich verdichten auf 1,06fache Proctordichte.
680 m³

Pos. 1.5
Trägerbohlwand als Baugrubenverbau herstellen, einschließlich der erforderlichen Anker, H = 6,0 bis 8,0 m.
1.680 m²

Titelsumme I Erdarbeiten

Titel II Beton- und Stahlbetonarbeiten

Pos. 2.1
Sauberkeitsschicht in B5, d = 5 cm, nach dem Abrütteln des Untergrundes einbauen und verdichten.
2.000 m²

Pos. 2.2
Beton für Einzel- und Streifenfundamente in B 25.
320 m³

Pos. 2.3
Eventualposition
Schalung für Fundamente, in rauer Qualität.
512 m²

Pos. 2.4
Bodenplatte aus B 25, d = 20 cm, eben und waagerecht einbauen, einschließlich erforderlicher Randschalung.
1.440 m²

Pos. 2.5
Beton für Stützen, B 25, in allen Stockwerken ab OK.-Fundamente, Querschnitt 40/40 cm.
85 m³

Pos. 2.6
Beton für Treppenläufe und Podeste, B 25, in allen Geschossen.
44 m³

Pos. 2.7
Beton für Wände in B 25, d = 20 bis 25 cm in allen Stockwerken.
848 m³

Pos. 2.8
Beton für Unter-, Überzüge, Randbalken und Attika in B 25.
310 m³

Pos. 2.9
Beton für Decken in B 25, Deckenstärke ca. 15 cm.
1.680 m³

Pos. 2.10
Beton für Rampenplatten in B 25, d = 15 cm, obere Betonfläche geneigt
126 m³

Pos. 2.11
Schalung für Stützen 40/40 cm in glattem Sichtbeton.
850 m²

Pos. 2.12
Schalung für Treppenläufe und Podeste in glattem Sichtbeton, die Seitenschalung wird nicht extra vergütet.
320 m³

Pos. 2.13
Schalung für die Wände der Pos. 2.7, in glattem Sichtbeton.

10.380 m²

Pos. 2.14
Schalung für die Unterzüge etc. der Pos. 2.8 in glatter SB-Qualität.
2.480 m²

Pos. 2.15
Schalung für die Decken der Pos. 2.9, Oberfläche geschlossen.
11.200 m²

Pos. 2.16
Schalung für Rampenplatten der Pos. 2.10, Oberfläche geschlossen.
960 m²

Pos. 2.17
Alternativ-Position zu den Pos. 2.5 und 2.11
Herstellen und Versetzen von FT-Stützen, Querschnitt 40/40, Form leicht konische, in B 25, glatte SB-Qualität, die Bewehrung wird nach Pos. 2.18 abgerechnet.
594 m

Pos. 2.18
Betonstahl BSt 420/500 nach DIN 488 für die Bewehrung von Stahlbetonbauteilen in Ortbeton und Stahlbetonfertigteilen, ∅ 6 bis 28 mm, nach den Angaben der Bewehrungspläne liefern, schneiden, biegen und verlegen.
50 t

Pos. 2.19
Betonstahlmatten BSt 500/550 RK nach DIN 488, als Lagermatten, nach den Bewehrungsplänen liefern, schneiden, biegen und verlegen.
65 t

Pos. 2.20
Liefern und Einbauen von Halfenschienen, Profil 50/30, verzinkt, mit Styroporfüllung, einschließlich Entfernen der Füllung.
1 m

Titelsumme II Beton- und Stahlbetonarbeiten

Titel III Mauerarbeiten

Pos. 3.1
Kalksandsteinmauerwerk für die Längs- und Stirnwände herstellen, KSV 1,8 und KSL 1,6, d = 24 cm.
645 m³

Titelsumme III Mauerarbeiten

Zusammenstellung der Titelsummen

Titel I Erdarbeiten
Titel II Beton- und Stahlbetonarbeiten
Titel III Mauerarbeiten
Nettoangebotssumme
Umsatzsteuer 15 % (Stand 1994)
Bruttoangebotssumme

8.1.2 Beschreibung des Kalkulationsverfahrens

Als Kalkulationsverfahren wird in diesem Anwendungsbeispiel die Kalkulation über die Angebotsendsumme (Umlagekalkulation) festgelegt. Gemäß diesem Kalkulationsverfahren erfolgt die Ermittlung der Gemeinkosten der Baustelle für jedes Bauprojekt separat. Allgemeine Geschäftskosten und Zuschläge für Gewinn und Wagnis können dann mit einem vordefinierten Zuschlagssatz den Herstellkosten hinzugefügt werden. Nach diesem Kalkulationsverfahren wird die Angebotskalkulation folgendermaßen durchgeführt:

- **Stufe 1:**
  Diese Stufe beinhaltet die Ermittlung der Kosten für Mittellohn, Stoffe und Geräte, sowie die Definition der Nachunternehmerleistungen. Außerdem werden die Zuschlagssätze für Zusatzkosten auf Löhne und Gehälter, Lohnnebenkosten, Kleingeräte und Werkzeuge, sowie die allgemeine Geschäftskosten ermittelt. Diese Ermittlung erfolgt nicht für jedes Bauprojekt, sondern einmalig am Jahresende und auf der Basis der Ergebnisse der Betriebsbuchhaltung. Ausnahmsweise können nur Bauprojekte mit Besonderheiten bei der Ermittlung der Zuschlagssätze berücksichtigt werden.

- **Stufe 2:**
  In dieser Stufe werden die Herstellkosten ermittelt. So werden die Einzelkosten der Teilleistungen, die baustellenspezifischen Gemeinkosten ermittelt. Danach werden die Einzelkosten der Teilleistungen und die Gemeinkosten summiert.

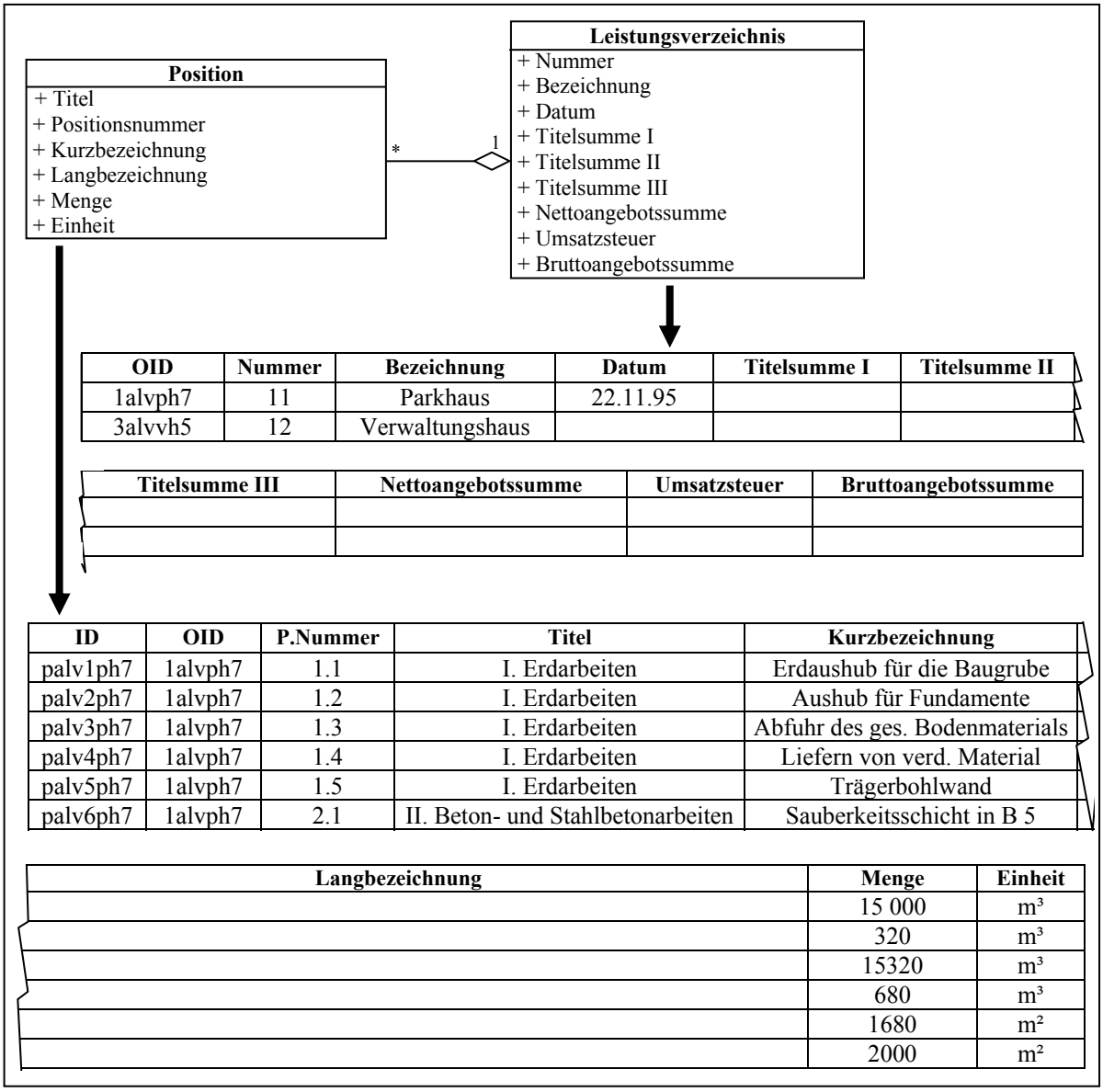
- **Stufe 3:**
  Hier wird die Nettoangebotssumme ermittelt.

- **Stufe 4:**
  In dieser Stufe werden die Zuschlagssätze auf Stoffkosten, Gerätekosten, Nachunternehmerleistungen festgelegt. Hier wird auch der Zuschlagssatz auf den Mittellohn ermittelt.

- **Stufe 5:**
  Diese Stufe befasst sich mit der Ermittlung der Einheitspreise der Teilleistungen und der Angebotspreise.

8.1.3 Modellierung des Leistungsverzeichnisses

Bei der Modellierung des Leistungsverzeichnisses werden die Klassen Leistungsverzeichnis und Position identifiziert. Damit wird das Leistungsverzeichnis als Aggregation von mehreren Positionen betrachtet. Demgemäss wird die Klasse Angebotsleistungsverzeichnis als Container-Klasse spezifiziert, welche die Positionen des Leistungsverzeichnisses verwaltet.



OID	Nummer	Bezeichnung	Datum	Titelsumme I	Titelsumme II
1alvph7	11	Parkhaus	22.11.95		
3alvvh5	12	Verwaltungshaus			

Titelsumme III	Nettoangebotssumme	Umsatzsteuer	Bruttoangebotssumme

ID	OID	P.Nummer	Titel	Kurzbezeichnung
palv1ph7	1alvph7	1.1	I. Erdarbeiten	Erdaushub für die Baugrube
palv2ph7	1alvph7	1.2	I. Erdarbeiten	Aushub für Fundamente
palv3ph7	1alvph7	1.3	I. Erdarbeiten	Abfuhr des ges. Bodenmaterials
palv4ph7	1alvph7	1.4	I. Erdarbeiten	Liefern von verd. Material
palv5ph7	1alvph7	1.5	I. Erdarbeiten	Trägerbohlwand
palv6ph7	1alvph7	2.1	II. Beton- und Stahlbetonarbeiten	Sauberkeitsschicht in B 5

Langbezeichnung	Menge	Einheit
	15 000	m³
	320	m³
	15320	m³
	680	m³
	1680	m²
	2000	m²

Abb. 8.1: Abbildung des Klassendiagramms des Angebotsleistungsverzeichnis in relationale Tabellen

In Abbildung 8.1 sind das Klassendiagramm des Leistungsverzeichnisses sowie die Abbildung der Klassen des Klassendiagramms in Tabellen der relationalen Datenbank gezeigt. Bei der objekt-relationalen Abbildung werden Klassendiagramme in Tabellen einer relationalen Datenbank nachgeformt. Bei dieser Abbildung werden auch solche Aspekte wie die Realisierung der Objektidentität in relationalen Datenbank behandelt. Diese Abbildung der Objekte in veranschaulichte Tabellen der relationalen Datenbank kann auch als Nachweis einer korrekten Modellierung betrachtet werden.

8.1.4 Modellierung der Mittellohnberechnung

Zur Mittellohnberechnung wird in der Praxis eine Liste erstellt. Diese Liste beinhaltet die Bezeichnung der Arbeiter, welche am Bauprojekt eingesetzt werden, deren Anzahl und Eigenschaften sowie Angaben über Menge der Arbeit die sie leisten, sowie die Kosten, die dadurch entstehen. Die Modellierung der Tabellen zur Mittellohnberechnung aus [Pran95] erfolgt auf der Basis des Ressourcenmodells. In der Regel wird der Mittellohn für Aufsicht und Belegschaft berechnet. Im Ressourcenmodell der Analysephase werden Aufsicht und Belegschaft als Spezialisierungsklassen der Klasse Arbeiter modelliert. Die Angaben aus den Tabellen zur Mittellohnberechnung aus [Pran95] werden als Attribute dieser Klassen spezifiziert. Die ermittelten Mittellöhne werden als Attribute der Klassen Belegschaftsliste und Aufsichtsliste betrachtet.

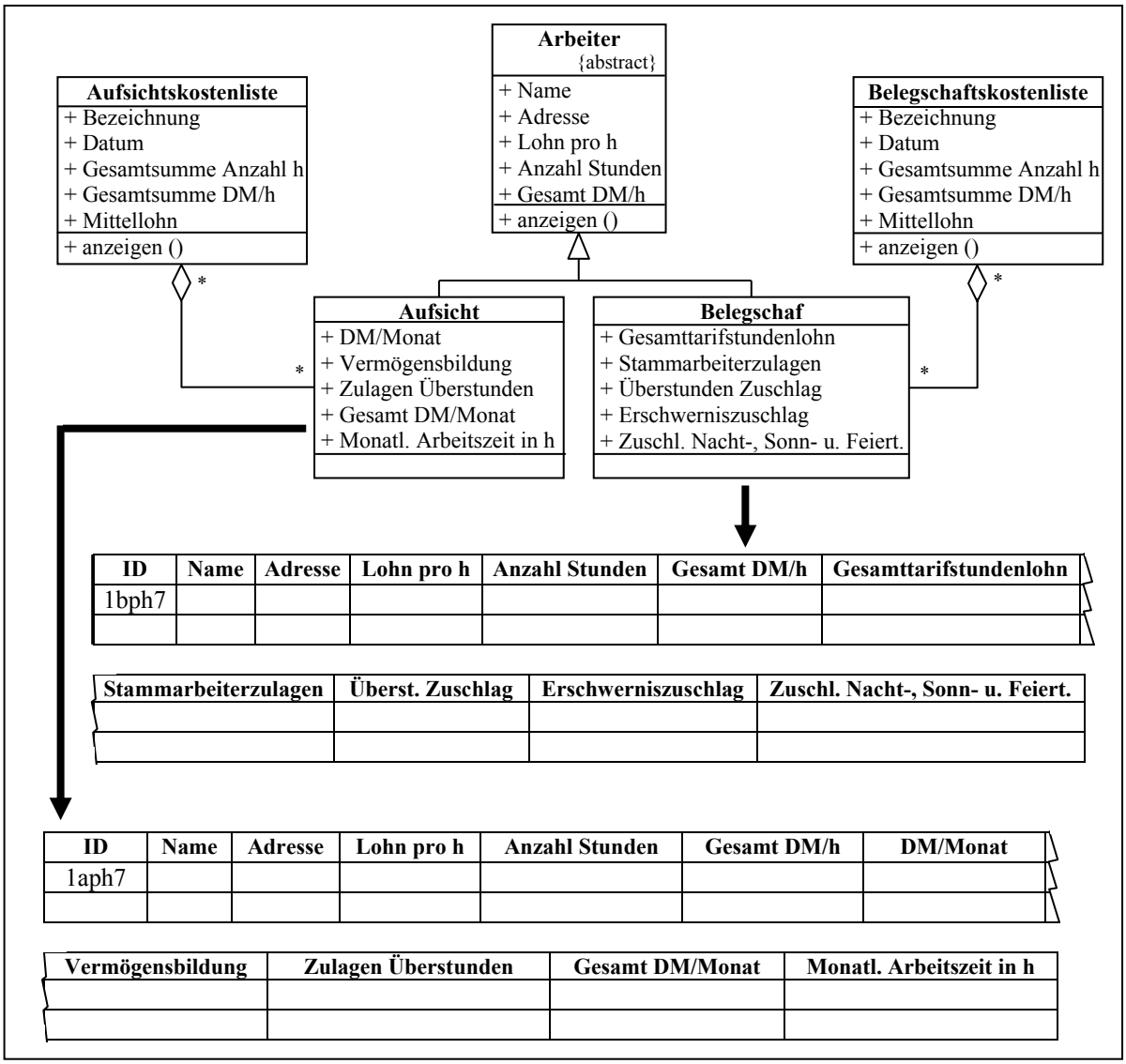


Abb. 8.2: Das Klassendiagramms zur Mittellohnberechnung und die Abbildung der Vererbung in Tabellen

In Abbildung 8.2 wird das Klassendiagramm zur Mittellohnberechnung dargestellt. In dieser Abbildung wird die Vererbungsbeziehung zwischen Arbeiter, Aufsicht und Belegschaft in Tabellen der relationalen Banken abgebildet.

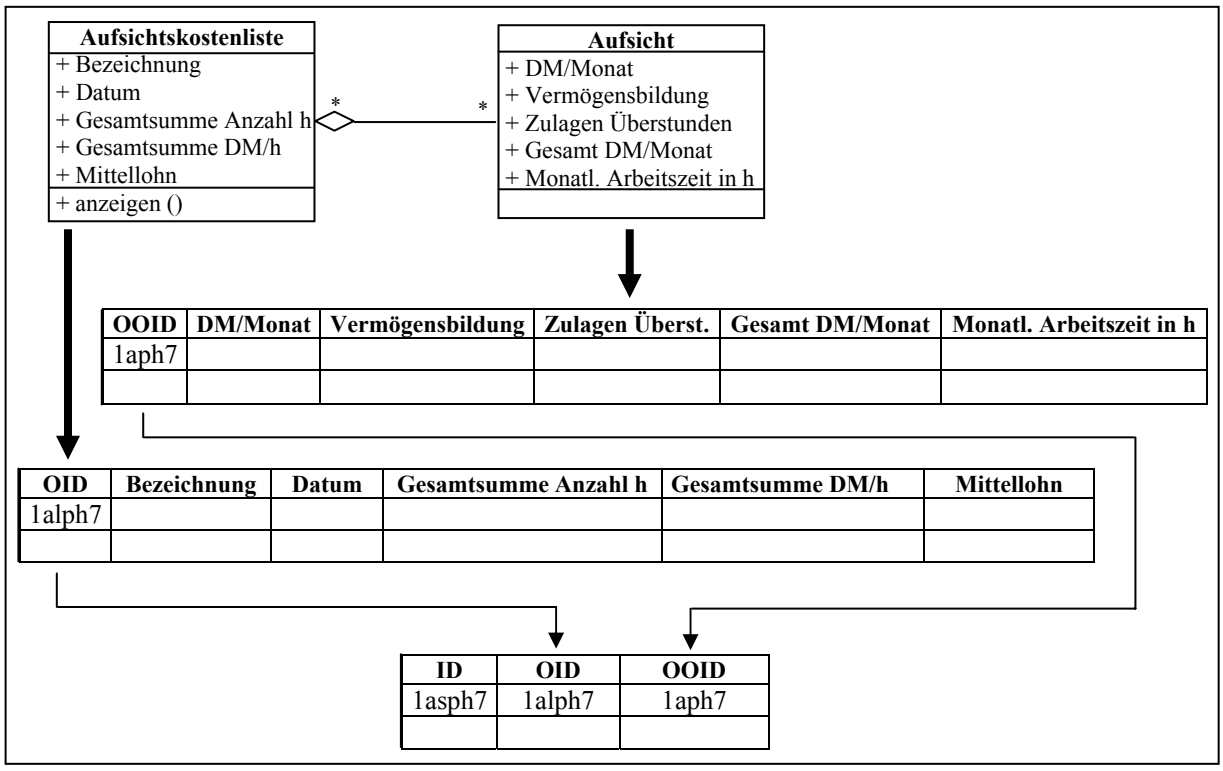


Abb. 8.3: Abbildung der Aggregation zwischen Aufsichtskostenliste und Aufsicht in Tabellen

Im weiteren Verlauf wird dieses Klassendiagramm zerlegt und die Klassen in Tabellen der relationalen Datenbank abgebildet (Abb. 8.3 und Abb. 8.4).

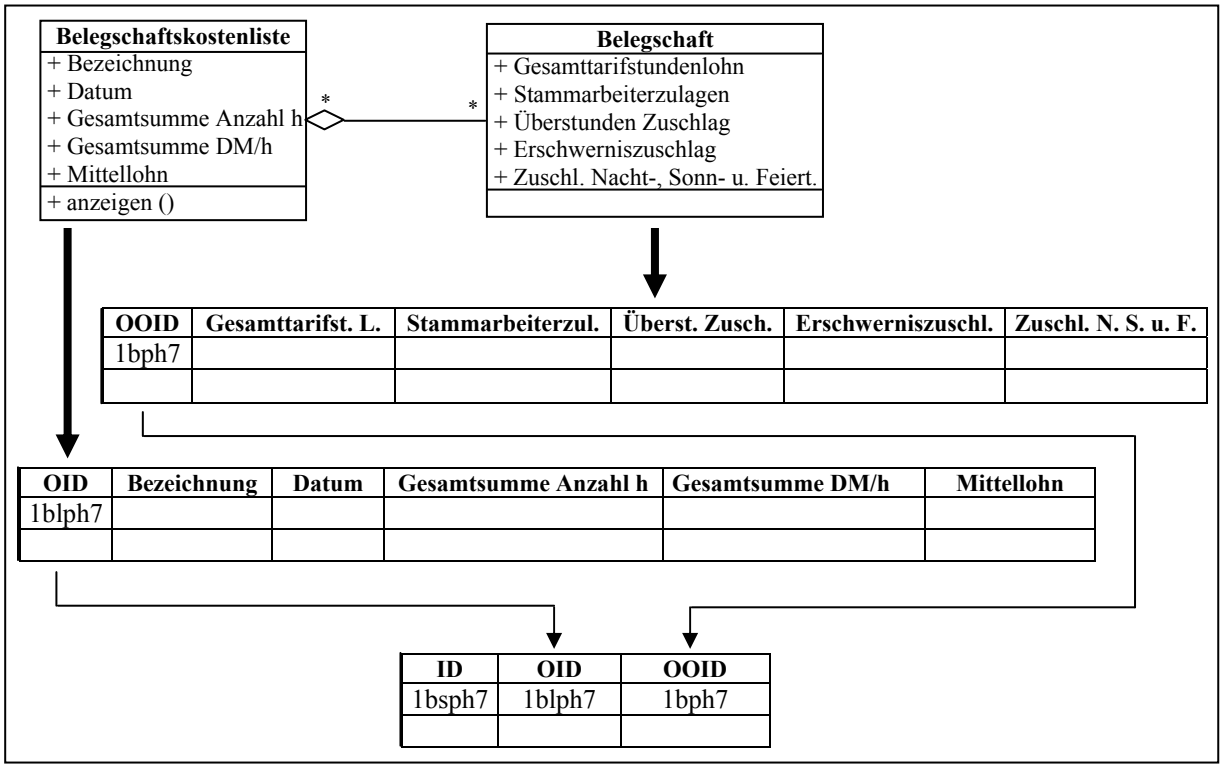


Abb. 8.4: Abbildung der Aggregation zwischen Belegschaftskostenliste und Belegschaf in Tabellen

8.1.5 Modellierung der Ermittlung der Gerätekosten

In der Praxis werden die ermittelten Gerätekosten in einer Liste aufgenommen. Diese Liste beinhaltet die Bezeichnung der Geräte, welche am Bauprojekt eingesetzt werden, deren Anzahl und Eigenschaften. Diese Liste beinhaltet auch Angaben über die Menge der Arbeit, die diese Geräte leisten und die Kosten die dadurch entstehen. In Abbildung 8.5 ist das Klassendiagramm der Liste zur Ermittlung der Gerätekosten dargestellt, wobei hier die Klasse Gerätekostenliste als Aggregation der Klasse Gerät spezifiziert wird. Die Abbildung dieser Aggregation in Tabellen der relationalen Datenbank kann auf ähnlicher Weise wie in den Abbildungen 8.3 und 8.4 erfolgen.

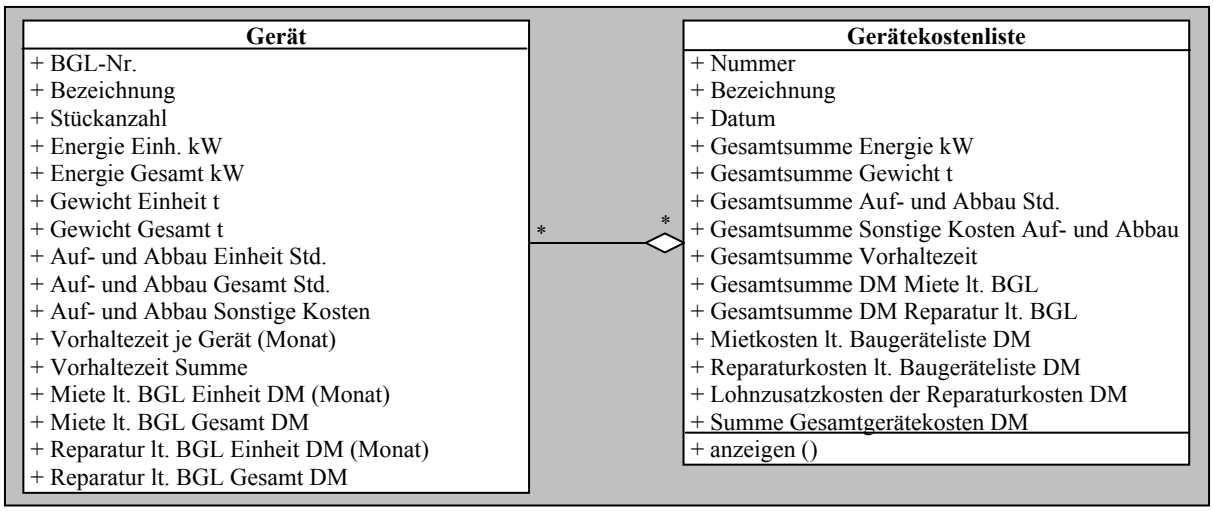


Abb. 8.5: Klassendiagramm der Liste zur Ermittlung der Gerätekosten

8.1.6 Modellierung der Ermittlung der Gemeinkosten der Baustelle

Bei der Ermittlung der Gemeinkosten der Baustelle wird in der Praxis eine Liste erstellt. Aus dieser Liste können die Klassen Gemeinkostenliste der Baustelle und Position spezifiziert werden (Abb.8.6). Grundsätzlich beziehen sich die Gemeinkosten der Baustelle auf verursachte Kosten von Geräten, Arbeitern sowie andere Arten von Ressourcen wie Holz, welche auf der Baustelle eingesetzt werden. Diese Arten von Ressourcen sind eigentlich im Ressourcenmodell als Klassen dargestellt. Dadurch können die Angaben in dieser Liste von den entsprechenden Objekten des Ressourcenmodells übertragen werden. Die Darstellung der Aggregation in Tabellen der relationalen Datenbank kann auf ähnlicher Weise wie z. B. in der Abbildung 8.3 erfolgen.

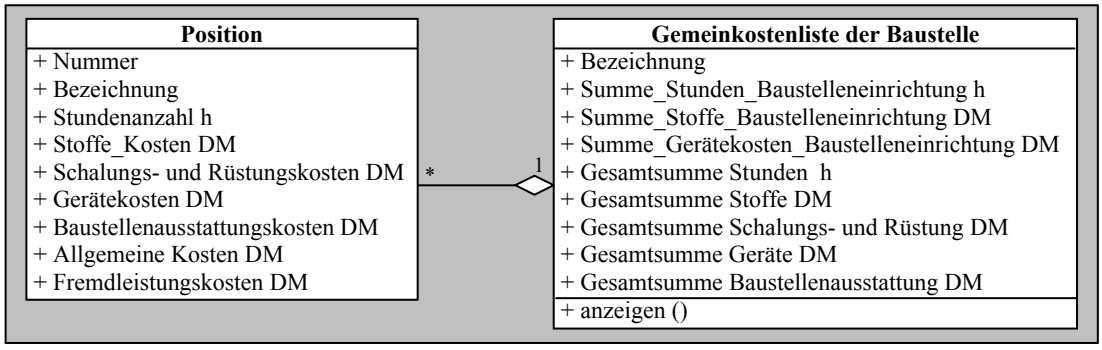


Abb. 8.6: Klassendiagramm der Liste zur Ermittlung der Gemeinkosten der Baustelle

8.1.7 Erstellung und Modellierung der Leistungskostenliste

Zur Erstellung der Leistungskostenliste werden die Tabellen zur Ermittlung der Einzelkosten der Teilleistungen, zur Zusammenstellung der Einzelkosten der Teilleistungen, zur Ermittlung der Herstellkosten, zur Ermittlung der Einheitspreise und der Positionspreise, welche in [Pran95] verwendet werden, zu einer Leistungskostenliste zusammengefasst. In diesen Tabellen sind die Ansätze je Einheit und Einzelkosten je Einheit identisch.

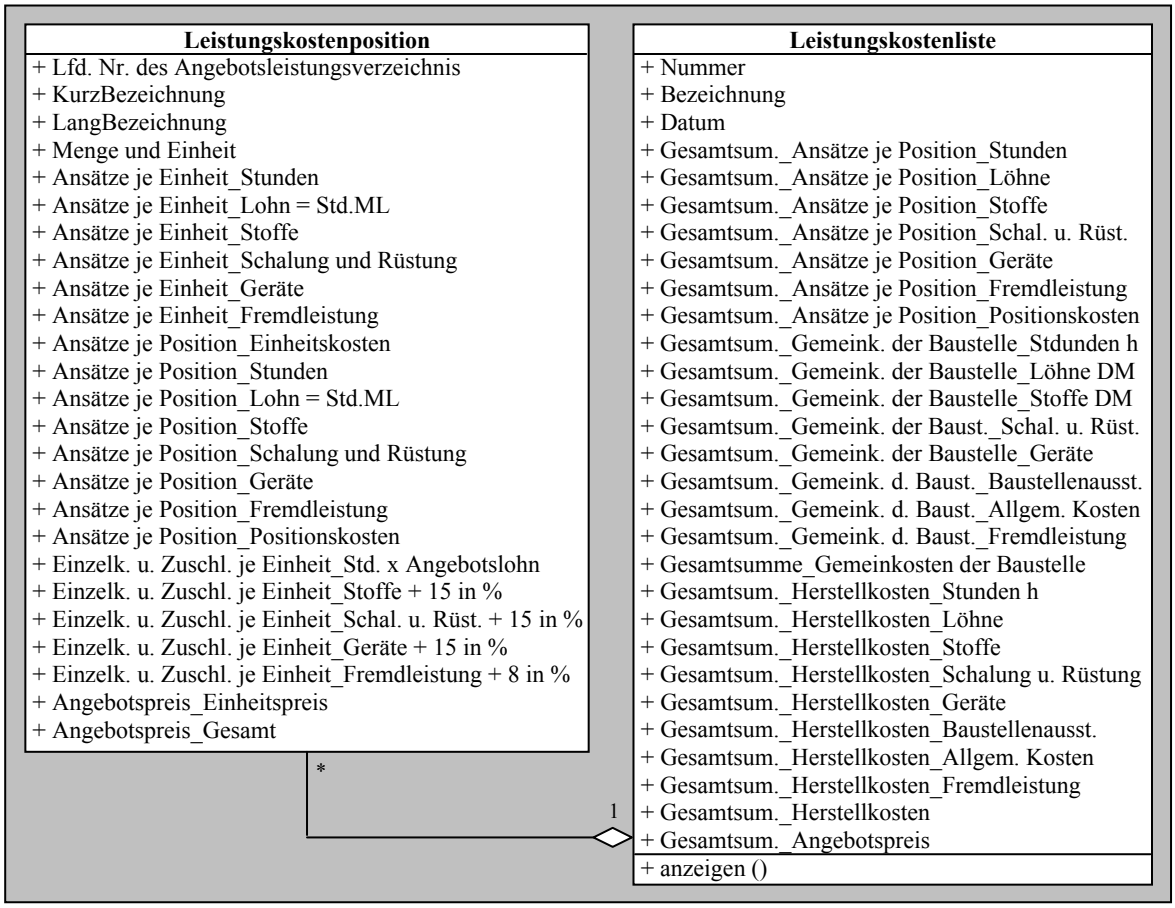


Abb. 8.7: Klassendiagramm der Leistungskostenliste

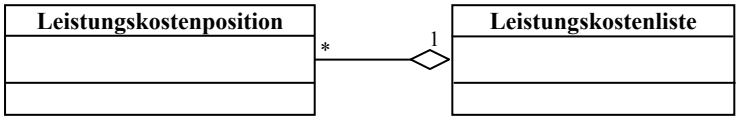
Bei der Modellierung dieser Leistungskostenliste werden die Klassen Leistungskostenliste und Leistungskostenposition spezifiziert. In Abbildung 8.7 sind diese Klassen innerhalb eines Klassendiagramms gezeigt. In Abbildung 8.8 ist die Aggregation zwischen Leistungskostenliste und Leistungskostenposition in Tabellen der relationalen Datenbank abgebildet.

OID	Nummer	Bezeichnung	Datum	Atr. 4	Atr. 5	Atr. 6
1ph7	19	Parkhaus	22.11.95	39629,82	2102361,95	900347,00
3vh5	20	Verwaltungshaus				

Atr. 7	Atr. 8	Atr. 9	Atr. 10	Atr. 11	Atr. 12	Atr. 13	Atr. 14
173677,50	39179,20	131095,00	4346660,65	7772,00	412304,60	161708,00	0,00

Atr. 15	Atr. 16	Atr. 17	Atr. 18	Atr. 19	Atr.20	Atr. 21	Atr. 22
403691,00	105373,00	357750,00	48320,00	1489146,60	47401,82	2514666,55	1062055,00

Atr. 23	Atr. 24	Atr. 25	Atr. 26	Atr. 27	Atr. 28	Gesamtsum._Angebotspreis
173677,50	442870,20	105373,00	357750,00	1179415,00	5835807,00	6.707.712,55



ID	OID	Lfd. Nr.	Kurzbezeichnung	Langbezeichnung	Menge u. Einheit	Atr. 4	Atr. 5
p1ph7	1PH7	Title I	I. Erdarbeiten				
P2ph7	1PH7	1.1	Baugrubenaushub		15000,00 m	0,05	2,65
P3ph7	1PH7	1.2	Fundamentaushub		320,00 m	0,15	7,96
P4ph7	1PH7	1.3	Abfuhr		15320,00 m		0,00
P5ph7	1PH7	1.4	Hinterfüllen		680,00 m	0,26	13,79
P6ph7	1PH7	1.5	Trägerbohlwand		1680,00 m		0,00

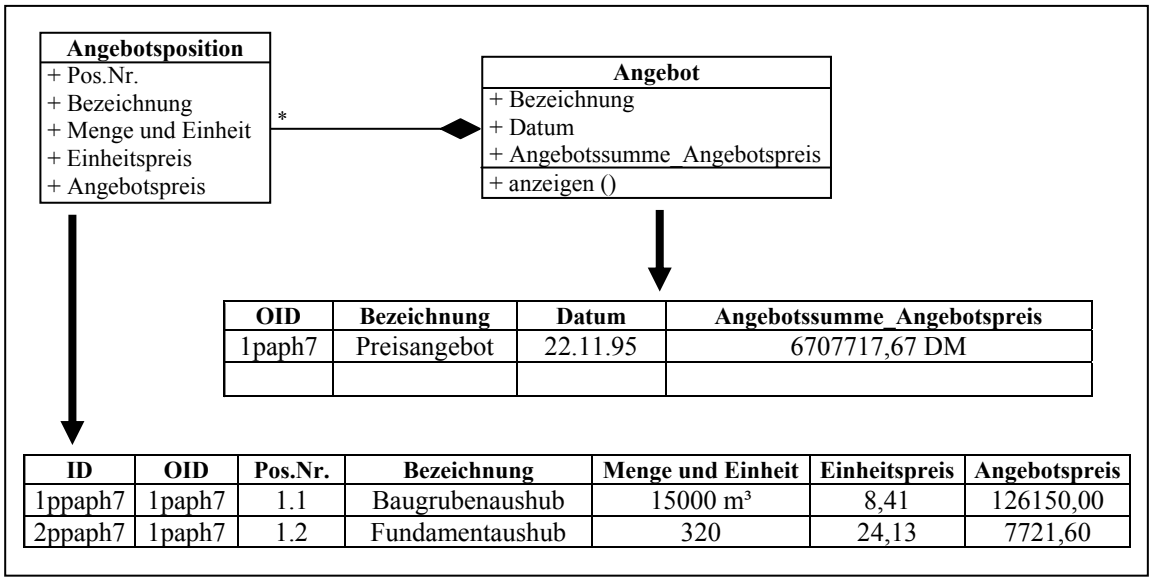
Atr. 6	Atr. 7	Atr. 8	Atr. 9	Atr.10	Atr. 11	Atr. 12	Atr. 13	Atr. 14	Atr. 15
					974,80	51713,14	17732,00	0,00	39179,20
0,42		2,30		5,37	750,00	39787,50	6300,00	0,00	34500,00
1,98		5,23		15,17	48,00	2546,40	633,60	0,00	1673,60
			18,00	18,00	0,00	0,00	0,00	0,00	0,00
15,88		4,42		34,09	176,80	9379,24	10798,40	0,00	3005,60
			400,00	400,00	0,00	0,00	0,00	0,00	0,00

Atr. 16	Atr. 17	Atr. 18	Atr. 19	Atr. 20	Atr.21	Atr. 22	Atr. 23	Angebotspreis_Gesamt
947760,00	1056384,34							1210883,77
0,00	80587,50	5,28	0,48	0,00	2,65	0,00	8,41	126090,00
0,00	4853,60	15,83	2,28	0,00	6,01	0,00	24,13	7720,16
275760,00	275760,00	0,00	0,00	0,00	0,00	19,80	19,80	303336,00
0,00	2318324,00	27,45	18,26	0,00	5,08	0,00	50,79	34537,61
672000,00	672000,00	0,00	0,00	0,00	0,00	440,00	440,00	739200,00

Abb. 8.8: Abbildung der Aggregation in der Leistungskostenliste in Tabellen der relationalen Datenbank

8.1.8 Modellierung der Erstellung des Preisangebotes

Bei der Erstellung des Preisangebotes wird in der Praxis eine Preisangebotsliste erstellt. Diese Liste kann als Komposition von Positionen des Preisangebotes modelliert werden. Demgemäss werden die Klassen Angebot und Angebotsposition spezifiziert.



OID	Bezeichnung	Datum	Angebotssumme_Angebotspreis
1paph7	Preisangebot	22.11.95	6707717,67 DM

ID	OID	Pos.Nr.	Bezeichnung	Menge und Einheit	Einheitspreis	Angebotspreis
1ppaph7	1paph7	1.1	Baugrubenaushub	15000 m³	8,41	126150,00
2ppaph7	1paph7	1.2	Fundamentaushub	320	24,13	7721,60

Abb. 8.9: Klassendiagramm des Preisangebotes und Abbildung der Komposition in Tabellen

In Abbildung 8.9 sind das Klassendiagramm des Preisangebotes sowie die Abbildung der Klassen in Tabellen der relationalen Datenbank dargestellt.

8.2 Anwendungsbeispiel 2: Datenaustausch durch RMI-Einsatz

Dieses Anwendungsbeispiel soll das Entwicklungsverfahren von RMI-Anwendungen sowie die RMI-Funktionsweise verdeutlichen. Das Beispiel besteht aus den Klassen Bauherr und Bauprojekt (Abb. 8.10). Es wird zuerst angenommen, dass sich diese Klassen in einem lokalen System befinden, und diese danach in einem verteilten System eingesetzt werden können.

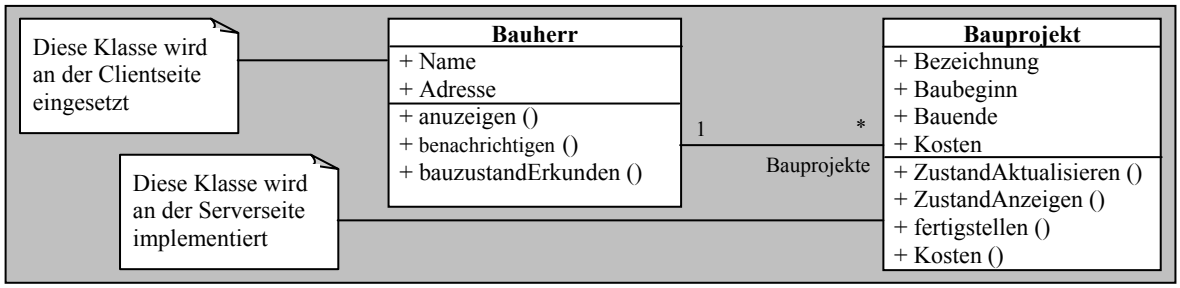


Abb. 8.10: Darstellung der Klassen Bauherr und Bauprojekt in einem Klassendiagramm

Der Code für die Klassen Bauprojekt und Bauherr ist folgend dargestellt, wobei hier nur die Deklarationen dargestellt werden, die für RMI-Einsatz von Bedeutung sind.

```
Public class Bauprojekt {

  public void zustandAnzeigen () {
       System.out.println ("Zustand angezeigt");
  }
}
```

```
public class Bauherr {

  public void bauzustandErkunden (Bauprojekt bauprojekt)
  {
   bauprojekt.zustandAnzeigen ();
   }
}
```

In diesem Anwendungsbeispiel ruft die Klasse Bauherr die Methode zustandAnzeigen () von der Klasse Bauprojekt mit dem Ziel auf, den Zustand des Bauprojekts anzusehen. Die Klasse Bauprojekt reagiert auf diese Methode mit der Ausgabe, dass der Zustand des Bauprojekts angezeigt wurde. Bei dem Einsatz der beiden Klassen in einem verteilten System soll die Klasse Bauherr auf einem lokalen Computer als Client und die Klasse Bauprojekt auf einem entfernten Computer als Server gestartet werden.

Das Verfahren zur Entwicklung dieser verteilten Anwendung mit RMI umfasst sowohl die Server- als auch die Clientseite. Auf der Serverseite werden ein Interface und zwei Klassen definiert. Die Klasse BauprojektImpl implementiert das Interface RemoteBauprojekt. Die Klasse BauprojektServer umfasst eine *main*-Methode und dient zur Verwaltung der Klasse BauprojektImpl. Auf der Clientseite wird die Klasse BauherrClient definiert. In Abbildung 8.11 sind diese Klassen anhand eines Klassendiagramms dargestellt.

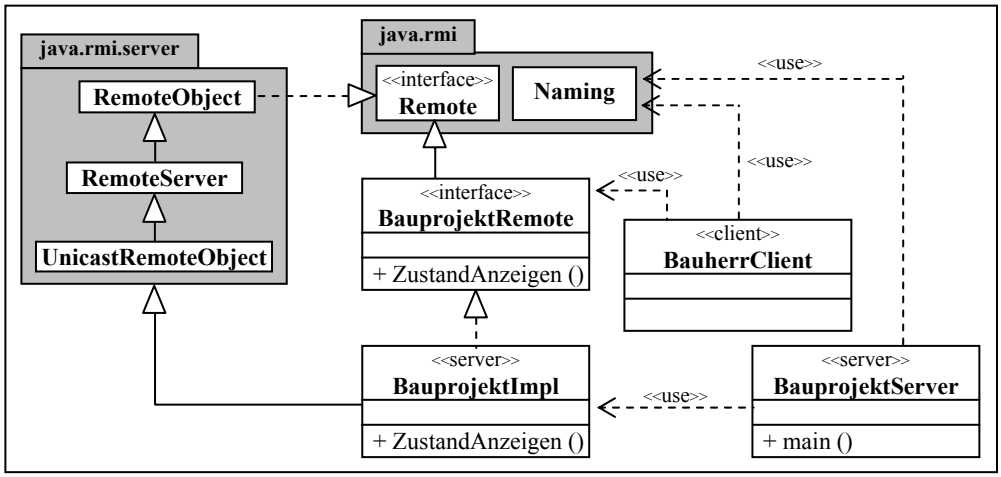


Abb .8.11: Klassendiagramm des RMI-Anwendungsbeispiels

Der Ablauf des Aufrufs der Methode zustandAnzeigen () ist anhand eines Sequenzdiagramms in Abbildung 8.12 dargestellt. Der BauherrClient erhält von der Methode lookup () eine Referenz auf ein Stub-Objekt. Die Methode zustandAnzeigen () wird vom Client aufgerufen und wird als eine entfernte Nachricht an die Serverseite gesendet. Das Skeleton-Objekt als entferntes Stellvertreterobjekt des Serverobjektes BauprojektImpl empfängt die entfernte Nachricht und ruft die Methode zustandAnzeigen () des Serverobjektes BauprojektImpl auf. Rückgabewerte werden auf dem analogen Weg zum Clientobjekt zurückgesendet, wobei hier mit *Exeptions* gerechnet werden kann.

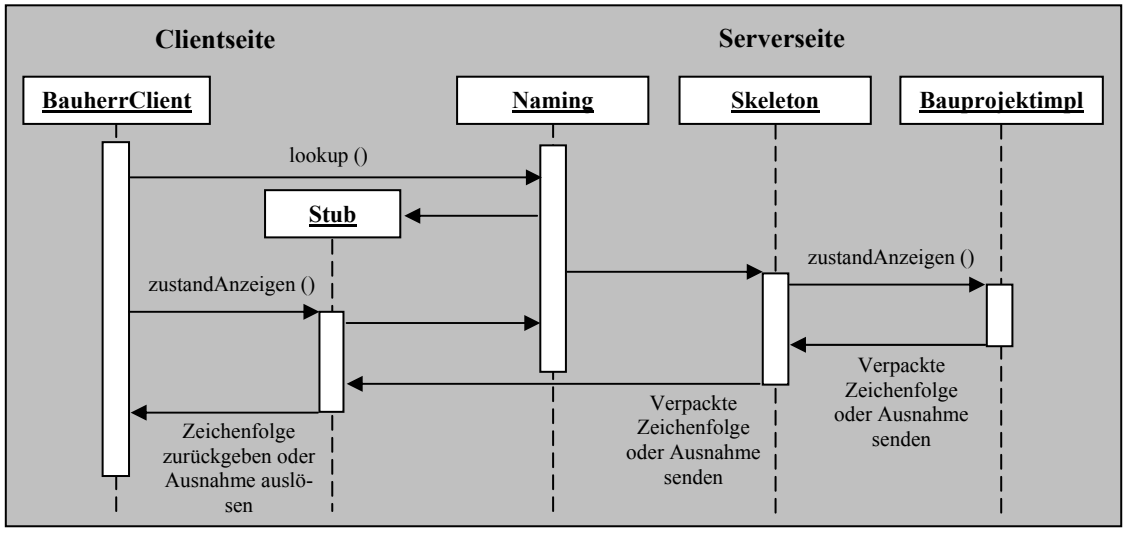


Abb. 8.12: Das Sequenzdiagramm zum Aufruf der Remote-Methode zustandAnzeigen

8.2.1 Verfahrensweise auf Serverseite

Auf der Serverseite werden ein Interface und zwei Klassen definiert. Die Klasse BauprojektImpl implementiert das Interface RemoteBauprojekt. Die Klasse BauprojektServer umfasst eine *main*-Methode und dient zur Verwaltung der Klasse BauprojektImpl.

- **Definition des Interface RemoteBauprojekt**

Mit der Definition der entfernten Klasse Bauprojekt als Java-Interface wird der Server seine Dienste zum entfernten Zugriff auf diese Klasse ankündigen. Dieses Java-Interface, welches RemoteBauprojekt genannt wird, wird folgendermaßen deklariert:

```
Package Bauprojekt ;

import java.rmi.* ;

// Das neu definierte Interface wird RemoteBauprojekt genannt
public interface RemoteBauprojekt extends Remote {
  public void ZustandAnzeigen ()
  throws java.rmi.RemoteException;
}
```

Dieses Interface erweitert das Interface *Remote* und beinhaltet die Methode zustandAnzeigen (), auf die entfernt zugegriffen werden soll. Bei der Definition des Interfaces soll berücksichtigt werden, dass bei einem entfernten Aufruf von Methoden noch Fehlerquellen auftreten können. Deswegen müssen stets alle diese Methoden Deklarationen beinhalten, die eine Ausnahme des Typs RemoteException auslösen.

Bei dem Einsatz der verteilten Anwendungen spielt die Ausnahmebehandlung eine wichtigere Rolle als es bei lokalen Anwendungen der Fall ist. Ursache dafür ist, dass hier mehr Fehlersituationen entstehen können, weil Methodenaufrufe entfernter Objekte sowieso fehleranfällig sind. Der Ausfall einzelner Server bzw. Störung der Netzverbindung und eine geringe Performance auf dem Netz werden hier als potentielle Fehlerquellen betrachtet.
Bei einem Fehlerauftritt wird eine *Exeption* ausgelöst. *Exeptions* sind spezielle Klassen, die im lokalen Fall von der Superklasse *Exception,* und im Remote Fall von der Superklasse *RemoteException* abgeleitet werden. Die Fehlerbehandlung kann in den beiden Fällen parallel vorgenommen werden, in dem die Exeptions abgefangen und von speziellen Programmteilen (*Exeption-Handler*) verarbeitet werden. Exceptions werden in Klassendefinitionen mit dem Schlüsselwort *throws* und durch die Angabe der möglichen Exeptionsobjekte deklariert.

- **Definition der Implementierungsklasse**

Bei der Definition der Implementierungsklasse wird eine Klasse deklariert, deren Ziel die Implementierung der entfernten Interfaces ist. Diese Klasse kann folgende Aufgaben erledigen:

- Spezifizierung der entfernten Interfaces, die von ihr implementiert werden sollen.
- Deklaration eines Konstruktors dieser Klasse.
- Implementierung der in den Interfaces definierten Methoden, die den entfernten Methodenaufruf gestatten.
- Einrichtung eines Sicherheitsmanagers.

- Erstellung von Instanzen dieser Klasse.
- Registrierung der Klasse, um sie dadurch bekannt zu machen und den entfernten Zugriff auf ihre Methoden zu ermöglichen.

Diese Implementierungsklasse wird BauprojektImpl genannt und erbt von der Klasse *UnicastRemoteObject* eine Subklasse von *RemoteServer*, welche eine TCP-basierte Punkt-zu-Punkt-Verbindung gestattet. Die Implementierungsklasse BauprojektImpl wird folgendermaßen deklariert:

Package Bauprojekt;

BauprojektImpl.java

```
import java.util.Vector;
import java.rmi.*;
import java.rmi.server.*;

// BauprojektImpl – Implementierung des Interfaces RemoteBauprojekt
// Diese Klasse speichert ihre Instanzen in einem Vector
public class BauprojektImpl extends UnicastRemoteObject
  implements RemoteBauprojekt {

   // Vector zur Aufnahme der Instanzen der Klasse
  Vector bauprojekte = new Vector ();

   // Konstruktor zur Erzeugung der Instanzen der Klasse
   public BauprojektImpl () throws RemoteException {

     // Hier wird der Konstruktor der Superklasse UnicastRemoteObject aufgerufen
     super ();
     }

   // Hier wird die Methode von Interface RemoteBauprojekt implementiert
  public void ZustandAnzeigen () {

     System.out.println ("Zustand angezeigt");
  }
}
```

• Definition der Verwaltungsklasse

Zur Verwaltung der Objekte der Klasse BauprojektImpl wird die Klasse BauprojektServer bereitgestellt, welche eine main-Methode enthält. Die Funktionalität dieser Klasse besteht nur darin, ein Bauprojekt für jeden Client bereitzustellen.
Eine Instanz der Klasse BauprojektImpl wird vom Server mit einem Namen verbunden. Somit werden Instanzen veröffentlicht, die mit einem Namen verbunden sind. Eine Instanz der Klasse BauprojektImpl wird veröffentlicht, indem der Server sie mit einem Namen verbindet. Um diese Verbindung aufzubauen wird die Methode *rebind ()* der Klasse *java.rmi.Naming* eingesetzt. Die Klasse BauprojektServer wird folgendermaßen deklariert:

Package Bauprojekt ;

BauprojektServer.java

```
// Diese Klasse implementiert den Server und registriert die Klasse Bauprojekt

import java.rmi.* ;

public class BauprojektServer {
   public static void main (String args[]) {

     // Erstellung und Installation des SecurityManager
      System.setSecurityManager (new RMISecurityManager ());
      try {

       // Erstellung einer Instanz von BauprojektImpl für die Registrierung
        System.out.println ("Bauprojektserver.main: Erstelle BauprojektImpl");
        BauprojektImpl bauprojektImpl = new BauprojektImpl ();

       // Bindung des Objektes an den Regestry
        System.out.println ("BauprojektServer.main: Verbinde mit Namen: bauprojektManager");
        Naming.rebind (“bauprojektManager“ , bauprojektImpl);
        System.out.println("bauprojektManager Server bereit.");
        }
        catch (Exeption e) {
            System.out.println ("BauprojektServer.main: Eine Exeption ist aufgetreten: “
            + e.getMessage ());
            e.printStackTrace ();
            }
   }
}
```

8.2.2 Verfahrensweise auf Clientseite

Die Aufgabe auf der Clientseite besteht nur darin, mit der Methode lookup (), in der Registry ein Bauprojekt-Objekt zu finden. Diese Aufgabe übernimmt hier die Klasse BauherrClient, welche folgendermaßen deklariert wird:

Package Bauherr;

BauherrClient.java

```
// Diese Klasse implementiert den Client
public class BauherrClient {

   public void BauzustandErkundigen (RemoteBauprojekt bauprojekt) {
     try {
     bauprojekt.ZustandAnzeigen ();
    }
    catch (RemoteExeption e) {
      System.out.println (e);
    }
}

    public static void main (String args[ ]) {
      Bauherr bauherr = new Bauherr ();

      // Erstellung und Installation des SecurityManager
       System.setSecurityManager (new RMISecurityManager ());
       Try {
          RemoteBauprojekt remoteBauprojekt = (RemoteBauprojekt) Naming.lookup (url);
          bauherr.BauzustandErkundigen (remoteBauprojekt);
       }
          catch (Exeption e)
          {
             System.out.println (e);
           }
   }
}
```

Wenn RMI von einem Applet aus benutzt wird, müssen einige Besonderheiten, insbesondere was die Sicherheit betrifft, beachtet werden. Weil Applets innerhalb eines Browser laufen, besitzen sie ihren eigenen Sicherheitsmanager. Demzufolge ist es hier nicht notwendig, den *RMISecurityManager* einzusetzen.

8.2.3 Verfahrensweise zur Kompilierung und Ausführung

Bei der Kompilierung ist der Einsatz von Classpath von Bedeutung, insbesondere zur späteren Wiederfindung der Klassen. Deshalb wird die Option "–d" bei der Kompilierung verwendet.

```
$ java –d . *.java
```

Mit dem Einsatz vom RMI-Compiler (*rmic)*, als Bestand des JDK, wird dann der Stub und das Skeleton der Klasse Bauprojekt erstellt. So werden mit dem Befehl:

```
$ rmic –d . Bauprojekt.BauprojektImpl
```

die Klassen BauprojektImpl_Stub.class und BauprojektImpl_Skel.class erzeugt. Ab Java 2 Plattform wird allerdings Skelet nicht mehr benötigt.

Zur Ausführung soll die *rmiregistry* aufgerufen werden. Dies geschieht allerdings nach der Kompilierung und vor dem Start des RMI-Server.

```
$ rmiregistry &
```

Der Server wird dann gestartet:

```
$java –Djava.security.policy=c:/java.policy
Bauprojekt.BauprojektServer &
```

Der Client wird danach gestartet:

```
$java –Djava.security.policy=c:/java.policy
Bauprojekt.BauherrClient
```

8.3 Anwendungsbeispiel 3: Verwaltung von Objekten mit Vector

In diesem Anwendungsbeispiel sind die Klassen Bauherr und Angebot einbezogen worden. In Abbildung 8.13 wird das Klassendiagramm dieses Beispiels dargestellt.

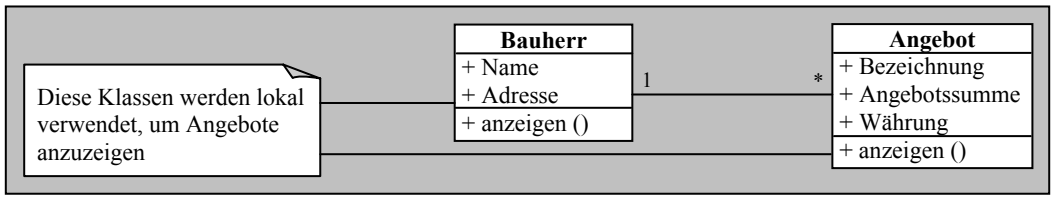


Abb. 8.13: Klassendiagramm des Anwendungsbeispiels

Nachfolgend werden die Deklarationen der Klassen Bauherr und Angebot dargestellt.

```
public class Bauherr {

  public String Name;
  public String Adresse;

  public Bauherr (String n, String a)
  {
     Name=n;
     Adresse=a;
   }

  public void anzeigen ()
  {
     System.out.println ("Name: "+Name+", Adresse: " +Adresse);
   }
}


public class Angebot
{
  public String Bezeichnung;
  public double Angebotssumme;
  public String Währung;

  public Angebot (String B, double S, String W)
  {
  Bezeichnung = B;
  Angebotssumme = S;
  Währung = W;
  }

  public void anzeigen ()
  {
  System.out.println ("Bezeichnung: " +Bezeichnung+", Angebotssumme:  "+
  Angebotssumme+Währung);
  }
}
```

Diese Klassen werden durch einen *Vector* verwaltet. Die Klasse *Vector* des Paketes *java.util* wird zur Darstellung einer linearen Liste verwendet. Grundsätzlich kann diese Liste, deren Länge zur Laufzeit inkonstant ist, Elemente beliebigen Typs umfassen. In Java wird ein *Vector* als *Array* von Elementen des Typs *Object* realisiert.
Prinzipiell kann ein *Vector* als ein komfortables Konzept zur Speicherung von Objekten betrachtet werden. Auf diese Objekte, die an beliebiger Stelle einer Liste eingefügt werden können, kann dann sowohl beliebig als auch sequentiell zugegriffen werden. Im Hinblick auf seine Flexibilität bezüglich der Art und Menge der zu speichernden Elemente wird dieses Konzept bei der Implementierung im Rahmen der vorliegenden Arbeit mehrfach eingesetzt.

```
import java.util.Vector;

public class Verwaltung
{
  public static void main (String [ ] argv)
  {
      Vector v = new Vector (5,3);

     //Bauherrdaten an v einfügen
      v.addElement (new Bauherr (“Neumann AG“, “Stadtring 39 - 03046 Cottbus“));
      v.addElement (new Bauherr (“DataSecurity GmbH“, “Jungstrasse 41 – 12052 Berlin“));
      v.addElement (new Bauherr (“Bush GmbH“, “Ringstrasse 22 - 72052 Essen“));
      v.addElement (new Bauherr (“SWare AG“, “Kantstrasse 39 - 03046 Cottbus“));
      v.addElement (new Bauherr (“DataGate GmbH“, “Entenstrasse 61 - 12052 Berlin“));
      v.addElement (new Bauherr (“HWare GmbH“, “Wolfstrasse 32 - 72052 Essen“));
      v.addElement (new Bauherr (“ITelecom AG“, “Am Turm 20 - 72052 Essen“));

      //Angebotsdaten an v einfügen
      v.insertElementAt (new Angebot (Parkhaus, 1357000 , DM ), 1);
      v.insertElementAt (new Angebot (Lagergebäude, 2343500, DM), 3);
      v.insertElementAt (new Angebot (Laborgebäude , 11363000, DM), 5);

      System.out.println (“Angaben von Bauherrn und Angebote \n “);
      for (int i = 0 ; i < v.size () ; i++)
      {
          Object o = v.elementAt (i);
          System.out.print (i+”: “);
          If (o insancof Bauherr)
            ((Bauherr) o).anzeigen ();
          else
            ((Angebot) o).anzeigen ();
      }
  }
}
```

Ein Vector verfügt über die Beigaben:

- eine Kapazität zur Angabe der Anzahl der Elemente, die aufgenommen werden können, sowie
- einen Ladefaktor zur Festlegung der Anzahl der Elemente, um die der interne Puffer erweitert wird, wenn beim Einfügen eines neuen Elements nicht mehr genügend Platz verfügbar ist.

In der folgenden Deklaration wird ein Vector mit einer anfänglichen Kapazität von 5 Elementen und einen Ladefaktor von 3 deklariert:

```
Vector v = new Vector (5, 3);
```

Die Anzahl der festgelegten anfänglichen Kapazität und des Ladefaktors haben Einfluss auf die Performance des Programms. Dies ergibt sich aus der Tatsache, dass je kleiner die Anzahl der anfänglichen Kapazität und des Ladefaktors sind, umso häufiger ist beim andauernden Einfügen von Elementen ein zeitaufwendiges Umkopieren notwendig. Bei der Instanzierung eines argumentlosen Vectors werden sein Puffer auf eine anfängliche Kapazität von 10 Objekten und sein Ladefaktor auf 0 gesetzt, wodurch die Kapazität bei jeder Erweiterung verdoppelt wird.

Grundsätzlich können neue Elemente wunschgemäß an das Ende des Vectors oder an einer anderen freien Stelle eingefügt werden. Um neue Elemente am Ende einzufügen, wird die Methode *addElement* eingesetzt. So wird in der folgenden Deklaration das *Object o* an das Ende der Liste von Elementen hinzugefügt:

```
public void addElement (Object o)
```

Um neue Elemente an einer beliebigen Stelle innerhalb der Liste einzufügen, wird die Methode *insertElementAt* verwendet. So wird in der folgenden Deklaration das *Object o* an die Position *index* in der Liste hinzugefügt:

```
public void insertElementAt (Object o, int index)
```

In der Klasse Verwaltung wird der *instanceof*-Operator verwendet, um die Zugehörigkeit von Objekten zu Klassen aufzuzeigen.

9 Schlussbetrachtungen

9.1 Wertung und Vergleich

Die Erfahrungen aus der Objektorientierten Modellierung in der vorliegenden Arbeit sowie die Entwicklung der letzten Jahre auf dem Gebiet der Objektorientierung zeigen, wie schwierig ein einheitliches Vorgehensmodell zu finden ist. Die unterschiedlichen und sich ständig ändernden Vorgehensweisen und Konzepte der Objektorientierung führten dazu, dass nur die Zuordnung dieser Vorgehensweisen und Konzepte eine Wissenschaft für sich geworden ist.
Die UML bringt auch keine Patentlösung. Sie beinhaltet wohlüberlegt kein Vorgehensmodell, um den Bereich der möglichen Anwendungen nicht zu verringern [OMG99]. Es ist mehr oder weniger von dem zu modellierenden System abhängig, wann und welche UML-Konzepte eingesetzt werden sollen. Auch bei der Modellierung des Systems in der vorliegenden Arbeit werden solche Konzepte eingesetzt, deren Einsatz zweckmäßig ist.

Die in der vorliegenden Arbeit dargestellte Vorgehensweise zur Modellierung des Systems der Kosten in Hochbauprojekten führt zur Erhöhung der Übersichtlichkeit und der Effizienz in den verschiedenen Phasen der Softwareentwicklung. Der Einsatz von Konzepten der UML hat den Effekt, dass diese Konzepte auf ihre Tauglichkeit zur Modellierung der baubetrieblichen Prozesse näher untersucht werden können. Auch die Strukturen der baubetrieblichen Prozesse können untersucht werden, um heraus zu finden, ob eine Vereinfachung dieser Strukturen zu einer besseren Unterstützung des Einsatzes neuer Technologien der Softwareentwicklung führt.

Aus der vorliegenden Arbeit ist zu folgern, dass die baubetrieblichen Prozesse zur Kalkulation der Kosten im Hochbau in Hinsicht einer effektiven EDV-Unterstützung zu komplex sind. Eine Vereinfachung dieser Prozesse wäre sinnvoll und führt zur Steigerung der Effizienz bei der Softwareentwicklung in diesem Bereich.

Die im Rahmen der vorliegenden Arbeit vorgestellten Modelle machen deutlich, dass die Objektorientierung eine zweckmäßige Technologie bei der Modellierung von komplexen Strukturen im Bereich der baubetrieblichen Aufgaben darstellt. Diese Technologie bietet auch die geeigneten Konzepte zur Implementierung der Ergebnisse der Objektorientierten Modellierung.

Zur Modellierung der baubetrieblichen Prozesse ist die UML gut geeignet. Die dadurch erstellten Modelle sind gut nachvollziehbar. Als Ausgangspunkt zur Modellierung bauwirtschaftlicher Prozesse eignet sich insbesondere die Modellierung anhand von Anwendungsfalldiagrammen. Aktivitätsdiagramme sowie Sequenzdiagramme sind für die Modellierung der Dynamik der bauwirtschaftlichen Prozesse von besonderer Bedeutung.

Im Hinblick auf die Modellierung der bauwirtschaftlichen Prozesse können bei der UML keine Defizite festgestellt werden. Es sind jedoch allgemeine Defizite bei der UML im Hinblick auf mangelnde Berücksichtigung einiger Konzepte festzustellen, die bei Objektorientierten Sprachen neu eingeführt wurden wie z. B. Properties, Services und Javas Package-Sichtbarkeit.

9.2 Zusammenfassung und Beurteilung

Ziel dieser Arbeit war die Objektorientierte Modellierung der Kosten in Bauprojekten des Hochbaus. Wesentliche Gesichtspunkte dieser Modellierung sind die Entwicklung des Strukturmodells der Analysephase und deren Verfeinerung in der Entwurfsphase. Dieses Strukturmodell gestattet die Abbildung der Aufbaustruktur der Bauprojekte sowie die Prozesse zur Kalkulation der Kosten in diesen Bauprojekten in Modellen, welche für die Softwareentwicklung in diesem Bereich von großer Bedeutung sind.
Die Konzepte der Objektorientierten Technologie, welche die Basis für die Objektorientierte Softwareentwicklung bilden, sind im Rahmen der vorliegenden Arbeit umfassend eingesetzt worden.

Bei der Modellierung auf Objektorientierter Basis, wie sie innerhalb der vorliegenden Arbeit eingesetzt wurde, liegen die Vorteile in der Möglichkeit, Programmteile modular zu erstellen und zu testen, sowie bereits entwickelte Programmteile wieder verwenden zu können und letztlich in der Tatsache, dass sie den Stand der Technik zeigt.

In der Auseinandersetzung mit verschiedenen Literaturquellen und der Analyse der Strukturen der baubetrieblichen Prozesse zur Kalkulation der Kosten in Bauprojekten wurden diese Strukturen hinsichtlich einer EDV-Unterstützung untersucht. Ausgehend von der Beschreibung dieser Strukturen wurde ein Objektmodell konzipiert.
Mit der vorliegenden Arbeit stellt der Autor den Anspruch, die baubetrieblichen Prozesse zur Kalkulation der Kosten im Hochbau Objektorientiert zu modellieren. Dabei soll eine Brücke zwischen den Fachgebieten Baubetrieb und Informatik geschlagen und eine Diskussionsplattform für weitere Forschungstätigkeiten im Bereich Informatik im Baubetrieb geschaffen werden.

Für die Praxis bringen die Modellierungsansätze in der vorliegenden Arbeit ein Objektmodell, das auf der Basis neuer Technologien entwickelt wurde. Es ist auch von Bedeutung für die Praxis, die Herangehensweise bei der Modellierung der bauwirtschaftlichen Prozesse innerhalb der vorliegenden Arbeit zu erwähnen. Insbesondere ist der vorteilhafte Einsatz der Aktivitätsdiagramme, mit denen auch Objekte identifiziert werden können, von großer Bedeutung. Im Übrigen eignen sich diese Diagramme auch zur Beschreibung der baubetrieblichen Aufgaben nicht nur zum Zweck der Softwareentwicklung, sondern auch zur Abbildung und Verdeutlichung dieser Prozesse. In der Praxis kann dieses Objektmodell erweitert werden. Die einzelnen Objektklassen können auch als Teilobjektmodelle entwickelt und verfeinert werden. Vorhandene Teilobjektmodelle können auch in diesem Objektmodell integriert werden.
Die Objektorientierte Modellierung der baubetrieblichen Prozesse gewinnt an Bedeutung mit der Zunahme der Softwareentwicklung auf Objektorientierter Basis. Die in der vorliegenden Arbeit vorgestellten Modellierungsergebnisse dienen als Grundlage zur Softwareentwicklung auf dem Gebiet der Kalkulation der Kosten in Bauprojekten des Hochbaus.
Diese Ergebnisse dienen auch zu einer besseren Integration der beteiligten Bestandteile an das Gesamtsystem der baubetrieblichen Prozesse. Die Bedeutung solcher Integration begrenzt sich nicht nur auf die Verbesserung der Effizienz und Effektivität der Durchführung des Bauprozesses durch die Entwicklung von geeigneten Softwaresystemen sondern auch auf ein zeitgerechtes und kostengünstiges Bauen durch die Realisierung der bauplanungsbegleitenden Ermittlung der Kosten in Projekten des Hochbaus.

In der vorliegenden Arbeit werden die Klassendiagramme als Ergebnisse der Objektorientierten Modellierung innerhalb eines Anwendungsbeispiels auf Tabellen einer relationalen Datenbank abgebildet. Dadurch wird gezeigt, dass bei der Softwareentwicklung Objektorientiert

modelliert werden kann und die Ergebnisse dieser Modellierung in eine andere Technologie implementiert werden können. Schließlich ist die Objektorientierte Modellierung effektiver und wirkungsvoller als die relationale Modellierung.

Das in der vorliegenden Arbeit entwickelte Strukturmodell kann als konzeptionelle Basis für die Entwicklung von Modellierungsumgebungen im Bereich Baubetrieb verwendet werden. Dieses Modell kann als Grundstein für die Standardisierung der Objekte in diesem Bereich betrachtet werden. Eine konsequente Definition und Modellierung dieser Objekte leistet einen Beitrag zur Steigerung der Effizienz und der Qualität im Bauprozess und wird als Fundament für die Automatisierung in diesem Bereich betrachtet.

Innerhalb der vorliegenden Arbeit ist der Bedarf nach mehr Investitionen in den frühen Softwareentwicklungsphasen, wie z. B. der Anforderungsanalyse, wahrgenommen worden. So wird eine detaillierte Beschreibung der Anforderungen des zu modellierenden Systems durchgeführt. Die Vorgehensweise bei der Objektorientierten Modellierung innerhalb der vorliegenden Arbeit eignet sich damit als Muster zur Softwareentwicklung im Bereich der Kalkulation der Kosten in Bauprojekten des Hochbaus.

Von wissenschaftlicher Bedeutung sind aus der vorliegenden Arbeit gewonnene Erkenntnisse im Bezug auf die Möglichkeit zur Vermeidung der Nachteile der gängigen Vorgehensweisen zur Softwareentwicklung im Bereich der baubetrieblichen Prozesse.
Bei der Vorgehensweise zur Softwareentwicklung innerhalb der vorliegenden Arbeit werden die Nachteile, die das Wasserfallmodell als Vorgehensmodell zur Softwareentwicklung mit sich bringt, überwunden. Diese Nachteile sind wesentlich durch die Entwicklung der Software für das ganze System verursacht [Balz00]. In diesem Zusammenhang wird die Annahme vorwiegend als Irrtum betrachtet, dass die Anforderungen bei der Entwicklung eines Softwaresystems nach diesem Vorgehensmodell gleichbleibend sind. Ein weiterer Irrtum ist die Annahme, dass ein System komplett und fehlerfrei modelliert werden kann, ehe mit der Implementierung angefangen wird. Daher ist es optimal, Software iterativ zu entwickeln. Demzufolge sollen die Phasen der Entwicklung nicht für das ganze System, sondern nur für ein Subsystem ausgeführt werden und dies soll im weiteren Verlauf der Entwicklung verfeinert, vervollständigt und auf das ganze System ausgedehnt werden. In Anbetracht dieser Feststellungen und nach der bisherigen Beschreibung des Systems der Baukalkulation innerhalb der vorliegenden Arbeit stellt sich heraus, dass es sinnvoller ist, dieses System nicht als Ganzes zu modellieren sondern in Subsysteme zu gliedern und zu modellieren. Diese Subsysteme, die unterschiedlichen Zwecken dienen, können auch als Systeme modelliert werden, wobei die Modellierung dieser Subsysteme als Erweiterung der Modellierung des Subsystems der Angebotskalkulation betrachtet werden kann.

Die bei der Entwicklung des Entwurfs innerhalb der vorliegenden Arbeit eingesetzten Entwurfsmuster zeigen, dass der Einsatz von solchen fortgeschrittenen Konzepten der Softwareentwicklung auch bei der Modellierung der baubetrieblichen Prozesse sinnvoll ist.
In der Entwurfsphase der vorliegenden Arbeit wurden auch einige Aspekte der Verteilung dargestellt. Hierzu wurden einige Konzepte der Verteilung insbesondere in Hinsicht auf Java als Plattform beschrieben.

9.3 Ausblick

Der netzwerkbasierte EDV-Einsatz im Bauwesen dringt immer mehr in den Vordergrund. Insbesondere im Bereich Baubetrieb werden künftig Daten nicht mehr in Aktenordnern ausgetauscht, sondern nur noch per Datennetz zwischen den Arbeitsplätzen verkehren. Aus diesem Grund sollen diese Daten effizient modelliert werden.

Damit die Bauwirtschaft den Anforderungen aus einem ständig wachsenden EDV-Einsatz und einem globalen Wettbewerb nachkommen kann, müssen insbesondere die baubetrieblichen Organisationsstrukturen und Prozessabläufe entsprechend umgestaltet werden. Die Bedeutung der Modellierung dieser Organisationsstrukturen und Prozessabläufe als Basis einer effizienten Softwareentwicklung wird deshalb weiter wachsen.

Die im Rahmen dieser Arbeit modellierten Strukturen der Baukalkulation scheinen für den modernen EDV-Einsatz zu komplex zu sein. Damit kann die Vereinfachung dieser Strukturen als weiterer Schwerpunkt für die Forschung im Bereich Baubetrieb und Bauwirtschaft betrachtet werden. Außerdem sollen im Fachgebiet Baubetrieb und Bauwirtschaft auch andere Strukturen und Verfahrensweisen im Hinblick auf den zunehmenden EDV-Einsatz vereinfacht werden, mit dem Ziel, den Aufwand bei der Softwareentwicklung auf diesem Gebiet zu reduzieren. In diesem Zusammenhang und im Hinblick auf Möglichkeiten der Vernetzung, kann die Baustelle als Teil der Unternehmensstruktur betrachtet werden. Dies bedeutet, dass viele der auf dem Gebiet der Wirtschaftsinformatik entwickelten Strukturen der stationären Industrie von Seiten der Bauindustrie übernommen werden können.

Die Erweiterung des innerhalb der vorliegenden Arbeit entwickelten Produktmodells durch Komponenten, die den Zeitaspekt betreffen, kann als Schwerpunkt des Forschungsvorhabens in diesem Bereich betrachtet werden. In diesem Zusammenhang kommt insbesondere eine umfassende Modellierung der Prozessabläufe auf der Baustelle in Betracht.

Im Hinblick auf den zunehmenden Einsatz der verteilten internetbasierten Anwendungen in der Bauwirtschaft können die vorhandenen verteilten Softwarearchitekturen bezüglich ihrer Tauglichkeit für den Einsatz im Bereich der baubetrieblichen Prozesse untersucht werden. Der Einsatz von verteilten Anwendungen kann auf der Basis eines zweischichtigen Client-Server-Modells realisiert werden. Jedoch bei dem Einsatz von Applikationen, die Datenbanken verwenden sollen, wie es im Hochbau der Fall ist, ist der Einsatz von Multi–Tier Architekturen unumgänglich.
Um eine effektive Anwendung der verteilten Objektverwaltungssysteme im Bauwesen zu gewährleisten, sollte auch das Bauwesen in die *Domain Services* anhand von z. B. CORBA bc *(CORBA building construction)* eingebunden werden.

Literaturverzeichnis

[Abel91] Abelson, H.; Sussman, G. J.; Sussman, J.: Struktur und Interpretation von Computerprogrammen. Springer-Verlag Berlin Heidelberg, 1991.

[Abel95] Abeln, O. (Hrsg.): CAD-Referenzmodell: Zur arbeitsgerechten Gestaltung zukünftiger computergestützter Konstruktionsarbeit. B. G. Teubner Stuttgart, 1995.

[Acht97] Achter, W.: Objektorientierte Software-Entwicklung: von der Strukturierung bis zur Migration. 2. Auflage, München, Computerwoche-Verlag, 1997.

[Andr87] Andrews, T; Harris, C: Combining Language and Database Advances in an Object-Oriented Development Environment. OOPSLA' 87 Conference Proceedings (October 4-8, Orlando, Florida, SIGPLAN Notices), Vol. 22, No. 12, 1987, S. 430-440.

[Andr92] Andres. A; Uhl, J.: Objektorientierte Software-Entwicklung: Eine Herausforderung für die Projektführung. in: Informatik Spektrum, (1992)15, S. 255-263.

[Atki89] Atkinson, M. et al.: The Object-Oriented Database System Manifesto. In DOOD89 The First Int. Conf. on Deductive and Object-Oriented Databases Proceedings (Dec. 4-6, 1989, Kyoto),W. Kim, ed. Elsevier Science Publishers B.V., Amsterdam, 1989, S. 40-57.

[Balz99] Balzert, H.: Lehrbuch der Objektorientierung: Analyse und Entwurf. Heidelberg, Berlin: Spektrum akad. Verl., 1999.

[Balz01] Balzert, H.: Lehrbuch der Software-Technik. Bd. 1. Software-Entwicklung, 2. Auflage, Heidelberg, Berlin: Spektrum akad. Verl., 2001.

[Bark92] Barker, R.: CASE*METHOD - Entity-Relationship Modelling. Addison-Wesley, Bonn München Paris, 1992.

[Beet97] Beetz, K.: O.P.E.N. Objectoriented Productdata Engineering Network. In: 9. Forum Bauinformatik, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 140, 1997

[Berk96] Berkhahn, V. und Sellerhoff, F.: Anwendung der Geometrischen Modellierung auf Trassierungen im dreidimensionalen Geländemodell. In: Forum Bauinformatik, Cottbus '96, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 135, 1996.

[Bert91] Bertino, E., Martino, L.: Object-Oriented Database Management Systems: Concepts and Issues. IEEE Computer 24 (4) 1991, S. 33-47.

[Blei90] Bleimann, U.: Betriebsinformatik: Informationsverarbeitungssysteme in Unternehmen und Verwaltung. Carl Hanser Verlag München Wien, 1990.

[Bolk87] Bolkart, W.: Programmiersprachen der vierten und fünften Generation. McGraw-Hill Book Company, Hamburg, 1987.

[Booch99] Booch, g., Rumbaugh, J., Jacobson, I.: Das UML-Benutzerhandbuch. Addison-Wesley_Longman, 1999.

[Broc95] Brockhaus: LexiROM. Microsoft Corporation und Bibliographisches Institut & F. A. Brockhaus AG, 1995.

[Casp90] Casper, E.: Ingenieurgerechte Wissensrepräsentation am Beispiel eines Objektorientierten Expertensystems für Stahlbau-Nachweise. In: J. Gauchel (Hrsg.), KI-Forschung im Baubereich, Ernst & Sohn Verlag für Architektur und techn. Wiss., Berlin,1990.

[Chen76] Chen, P.: The Entity-Relationship Modell - Toward a Unified View of Data. In: ACM Transactions on Database Systems, Vol. 1, March 1976, S. 9-36.

[Chen91] Chen, P.: Der Entity-Relationship-Ansatz zum logischen Systementwurf: Datenbank- und Programmentwurf. BI-Wiss.-Verlag, Mannheim Wien Zürich, 1991.

[Chen92] Chen, P.: Entwicklungsrichtungen des Entity Relationship Ansatzes. In: Wirtschaftsinformatik 34 (1992) 4, S. 453-454.

[Copl92] Coplien, j. O.: Advanced C++ Programming Styles and Idioms. Addison-Wesley, Reading, Massachusetts, 1992.

[Dall87] Dallman, H.; Elster, K. H.: Einführung in die höhere Mathematik. Gustav Fischer Verlag Jena, 1987.

[Diaz97] Diaz. J.: Objektorientierte Modellierung geotechnischer Systeme. Dissertation, TU Darmstadt, Berichtsheft Nr. 2/98, Institut für Numerische Methoden und Informatik im Bauwesen.

[Ditt90] Dittrich, K.: Objektorientierte Datenbanken. In: P. Mertens (Hrsg.), Lexikon der Wirtschaftsinformatik, Springer-Verlag, Berlin Heidelberg, 1990.

[Ege92] Ege, R. K.: Programming in an Object-Oriented Environment. Academic Press, Inc. San Diego, California 92101, 1992.

[Eign91] Eigner, M. et al: Engineering Database: strategische Komponente in CIM-Konzepten. München Wien, Hanser, 1991.

[End90] End, W. et al: Softwareentwicklung: Leitfaden für Planung, Realisierung und Einführung von DV-Verfahren. Siemens AG Verlag Berlin und München, 1990.

[Endr92] Endres, A.; Uhl, J.: Objektorientierte Software-Entwicklung. In: Informatik-Spektrum 15 (1992) 5, S. 255-263.

[Fied91] Fieder, J.; Rix, K.; Zöller, H.: Objektorientierte Programmierung in der Automatisierung. VDI-Verlag, Düsseldorf, 1991.

[Flat90] Flatscher, R. G.: Design relationaler Datenbanken: Daten einer betrieblichen Auftragsabwicklung relational organisieren. IWT Verlag, Vaterstetten bei München, 1990.

[Flat93] Flatscher, W.: Objektorientiertes Kalkulieren und Ausschreiben. In: Bauinformatik (1993) 6, S. 246-251.

[Flat96] Flatscher, W.: Stolperstein zwischen AVA und CAD. In: Bauinformatik (1996) 3, S. 10-13.

[Gabb96] Gabbert, U.; Wehner, P.: Integration der Finite-Element-Methode im Produktdatenmodell. In: Forum Bauinformatik, Cottbus '96, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 135, 1996.

[Gehe00] Gehbauer, F.; Lennerts, K.: Modellierung und Entwicklung eines Wissensbasierten, Objektorientierten Systems zur Planung optimierter Baustellen-Layouts (ESBE). In: Hartmann, D. (Hrsg.): Objektorientierte Modellierung in Planung und Konstruktion. Deutsche Forschungsgemeinschaft, Forschungsbericht. Wiley-VCH GmbH, 2000.

[Gehr92] Gehri, M.: Computerunterstützte Baustellenführung. VDF Verlag der Fachvereine Zürich, 1992.

[Giel93] Gielingh, W. F.; Suhm, A. K.: IMPPACT Referenz Model - An Approach to Integrated Product and Process Modelling for Discrete Parts Manufacturing Reports ESPRIT Series. Springer-Verlag. 1993

[Grab79] Grabowski, H.; Eigner, M.: Anforderungen an CAD-Datenbanksysteme. VDI-Zeitschrift, 121 (1979) 12, S. 621-633, 1979.

[Grab91] Grabowski, H.; Anderl, R.: Advanced Modelling, Research Reports ESPRIT Project 322, CAD Interfaces (CAD*I). Springer-Verlag, 1991.

[Grab94] Grabowski, H.; Anderl, R. und Poolly, A.: Integriertes Produktmodell. Beuth-Verlag, 1994.

[Grün96] Grüneis, H.: Commerzbank-Hochhaus Frankfurt am Main: Projektentwicklung und Projektmanagement. Bauingenieur 71, (S. 305-313), Springer-Verlag, 1996.

[Haas97] Haas, W.; Endres, M.: STEP-CDS: CAD-Datenaustausch im Bauwesen. In: ProduktDaten Journal, Nr. 1, Dezember 1997, 4. Jahrgang.

[Hain99] Hain, K.; Meis, E.: Integriertes Produkt- und Produktionsmodell (PPM). Version 4.0, internes Arbeitspapier des SFB 346, Institut für Rechneranwendung in Planung und Konstruktion, Universität Karlsruhe, 1999.

[Hall94] Haller, H.-W.: Ein Produktmodell für den Stahlbau. Dissertation, Universität Fridericiana zu Karlsruhe, Fakultät Bauingenieur und Vermessungswesen, 1994.

[Hart00] Hartmann, D..: Grundlegende Betrachtungen zur Anwendung der Objektorientierung. In: Hartmann, D. (Hrsg.): Objektorientierte Modellierung in Planung und Konstruktion. Deutsche Forschungsgemeinschaft, Forschungsbericht. Wiley-VCH GmbH, 2000.

[Hess94] Hesse, W.; Barkow, G. et al: Terminologie der Softwaretechnik - Ein Begriffsystem für die Analyse und Modellierung von Anwendungssystemen. Teil 2: Tätigkeits- und ergebnisbezogene Elemente. Informatik-Spektrum (1994) S. 96-105.

[Heue91] Heuer A.: Konzepte Objektorientierter Datenmodelle. In: G. Vossen (Hrsg.), Entwicklungstendenzen bei Datenbanksystemen, Oldenbourg Verlag München Wien, 1991.

[Heue97a] Heuer, A.: Objektorientierte Datenbanken - Konzepte, Modelle, Standards und Systeme. Addison Wesley Verlag, 1997.

[Heue97b] Heuer, A.; Saake, G.: Datenbanken: Konzepte und Sprachen. MITP Verlag, Boon, 1997.

[Hilb97] Hilbert, K.H.: STEP verbindet Datenwelten. In: Produktdaten Journal Nr. 2/1997, S. 8-9.

[HOAI 91] HOAI: Verordnung über die Honorare für Leistungen der Architekten und der Ingenieure. Düsseldorf, Werner Verlag, 1991.

[Höre98] Hörenbaum, C.; Osterrieder, P.; Weichert, J.: Neue Entwicklungen bei der Produktmodellierung im Stahlbau und Holzbau. Festschrift Friedrich und Lochner GmbH, S. 60-65, Stuttgart, 1998.

[Huhn00] Huhnt, W.; Beucke, K.: Integrierte Bearbeitung technischer und betriebswirtschaftlicher Aufgaben im Bauwesen. In: Bauingenieur, 75 (2000), Nr. 2, S. 78-86, Springer Verlag, 2000.

[Hurs95] Hurschka, P.: OO-Design - ganz einfach oder sehr kompliziert. OBJEKTspektrum 5/95, S. 79-80.

[Hütt96] Hüttermann, R. et al: Die Entwicklung von Komponenten für den Einsatz bei bauingenieurspezifischen Problemen. In: Forum Bauinformatik, Cottbus '96, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 135, 1996.

[IDE092] IDEF0 (ICAM Definition Language 0): Federal Information Processing Standards, National Institut for Standard and Technology, Draft (Sept. 1992).

[Jell91] Jell, T.: Objektorientierte Programmierung mit C++. Hanser Verlag, München, Wien, 1991.

[Kala00] Kalantari, B.; Schäffler, H.; Diaz, J.: Datenaustausch im Bauwesen auf der Basis von XML. In: Forum Bauinformatik 2000, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 163, 2000.

[Kark97] Karkola, C.: Integration der Schnittstelle ISO 10303 (STEP) AP 225 in ein computer aided facility management system. In: 9. Forum Bauinformatik, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 140, 1997.

[Keil88] Keil, W.; Martinsen, U.: Einführung in die Kostenrechnung für Bauingenieure. Werner-Verlag GmbH, Düsseldorf, 1988.

[Kemp93] Kemper, A.; Moerkotte, G.: Basiskonzepte Objektorientierte Datenbanksysteme. In: Informatik Spektrum (1993) 16, S. 69-80.

[Kiew90] Kiewert, A.: Kostenfrüherkennung in der Konstruktion durch Kopplung von CAD und Kostenrechnung. In: A.-W. Sheer (Hrsg.), Rechnungswesen und EDV, 11. Saarbrücker Arbeitstagung, Physica-Verlag Heidelberg, 1990.

[Khos90] Khoshafian, S.; Abnous, R.: Object Orientation: Concepts, Languages, Databases, User Interfaces", John Wiley & Sons, Inc., 1990.

[Klau02] Klauer, T.: Integration von Sach- und Kosteninformationen in das Modellbasierte Projektkommunikationssystem BauKom-Online. In: Forum Bauinformatik 2002, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen, Nr.181, 2002.

[Koch95] Koch, R. et al: Konstruktionsbegleitende Kalkulation auf der Basis eines Prozesskostensatzes. In: Konstruktion 47 (1994), S. 427-433, Springer Verlag, 1994.

[Krau92] Krause, G.: Objektorientierte Programmierung und ihr Einsatz zur Entwicklung von Informationssystemen. Service Fachverlag, Wien, 1992.

[Krus96] Kruschwitz, E.: Euler-Modellierung dreidimensionaler Körper. In: Forum Bauinformatik, Cottbus '96, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 135, 1996.

[Laab98] Laabs, A.: Methoden für die Modellierung mit Objekten im Bauingenieurwesen. Dissertation, TU Berlin, Shaker Verlag, Aachen, 1998.

[Leis01] Leistner, C.: Automatisierung von Bauprozessen im Projektmanagement. In: Forum Bauinformatik 2001, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen, Nr.169, 2001.

[Lena90] Lenart, M.; Ketteler, G.: Expertensysteme für den Entwurf von Baukonstruktionen. In: J. Gauchel (Hrsg.), KI-Forschung im Baubereich, Ernst & Sohn Verlag, Berlin, 1990.

[Lock91] Lockemann, P.: Objektorientierte Datenbanksysteme. In: H. Schneider (Hrsg.), Lexikon der Informatik und Datenverarbeitung, Oldenbourg Verlag München Wien, 1991.

[Mart91] Marty, R.: Objektorientierte Softwareentwicklung - Strategische Perspektiven. Konferenz-Einzelbericht, 4. Kolloquium Technische Akademie Esslingen, 1991.

[Meiß95] Meißner, U., Rüppel, U.: Das Objektorientierte statische Modell als neutrale Statikschnittstelle für CAD- und Berechnungsprogramme. In Bauinformatik 4/1995, S. 140-144.

[Mins68] Minsky; M.: Matter, mind and Models. In: Semantic Information Processing, Minsky (ed.), Cambridge, Mass. 1968.

[Moch93] Mochel, T.: Objektorientierte Simulation- Ein neues Konzept zur Simulation diskreter Systeme. Verlag Shaker, Aachen , 1993.

[Muel92] Muellenbach, S.; Nieva, A.: EXPRESS Schema management and Instantiation Prototype. ESPRIT II Project NEUTRABAS, EXPRESS User's Group. Dallas, 1992.

[Nies93] Niestroj, A.: Objektorientierte Analyse für den bauteilorientierten Datenaustausch von der Objekt- zur Tragwerksplanung. Dissertation, TU Darmstadt, Berichtsheft Nr. 1/93, Institut für Numerische Methoden und Informatik im Bauwesen.

[Nowa83] Nowacki, H.: Geometrisches Modellieren - eine Kurzübersicht. VDI-Bericht 492, S. 321-328. Düsseldorf, VDI-Verlag, 1983.

[Ohts93] Ohtsubo, H.; Kawamura Y.; and Kubota, A.: Development of the Object-Oriented Finite Element Modelling System – MODIFY. In: Engineering with Computers (1993) 9, S. 187-197, Springer-Verlag London Limited, 1993.

[Oste97] Osterrieder, P.; Haller, H.; Saal, H.: Bauspezifische, fertigungsorientierte Produktmodellierung im Stahlbau und Holzbau. Bauingenieur 11/1997, (S. 305-313), Springer-Verlag, 1997.

[Oste00] Osterrieder, P.; Weichert, J.: Produktmodell DtH. Schriftreihe Statik und Dynamik 2/2000. USSN 1615-3952. BTU-Cottbus, Lehrstuhl Statik und Dynamik (Hrsg.).

[Ott91] Ott, H. J.: Software-Systementwicklung: Praxisorientierte Verfahren und Methoden. Hanser Verlag München Wien, 1991.

[Pall92] Pallischeck O.; Weichenhardt, F.: Objektorientierte Unternehmensmodellierung für CIM. In: CIM Management (1992) 4, S. 21-25.

[Pick89] Pickel, H.: Kostenmodelle als Hilfsmittel zum Kostengünstigen Konstruieren. Hanser Verlag, 1989.

[Poll96] Polly, A.: Methodische Entwicklung und Integration von Produktmodellen. Shaker Verlag, Aachen, 1996.

[Pran95] Prange, H.; Leimböck, E.; Klaus, U.: Baukalkulation unter Berücksichtigung der KLR Bau und der VOB. Bauverlag GmbH, Wiesbaden und Berlin, 9. Auflage 1995.

[Prin99] Prinz, P.; Kirch-Prinz, U.: Objektorientiertes Programmieren mit Ansi C++. Prentice Hall, 1999.

[REFA84] REFA in der Baupraxis, Teil 1: Grundlagen, G. Berg, Zeittechnik Verlag, Frankfurt/Main, 1984.

[Reym95] Reymendt, J.: Ein Beitrag zur Objektorientierten Modellierung im Massivbau. Dissertation. TH Darmstadt, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 128, 1995.

[Rich91] Richter, R.: wissensbasierte CAD-Systemkomponente zum Entwurf montagegerechter Produkte. Springer Verlag, 1991.

[Rist85] Ristan W.; Möller, U.: Baupläne als Ergebnis des Rechnerintegrierten Anordnungs-Planungs-Systems RAPS. VDI-Berichte Bd. 570.3, S. 37-52, Düsseldorf, VDI-Verlag, 1985.

[Roll95] Roller, D.: CAD Effiziente Anpassungs- und Variantenkonstruktion. Springer-Verlag, 1995.

[Rumb93] Rumbaugh, J., Blaha, M., Premerlani, W.; Eddy, F., Lorensen, W.: Objektorientiertes Modellieren und Entwerfen. Carl Hanser und Prentice Hall International, 1993.

[Rüpp94] Rüppel, U.: Objektorientiertes Management von Produktmodellen der Tragwerksplanung. Dissertation, TH Darmstadt, Berichtsheft Nr. 1/94, Institut für Numerische Methoden und Informatik im Bauwesen.

[Rüpp96] Rüppel, U., Meißner, U.: Integrierte Planung, Fertigung und Nutzung von Bauwerken auf der Basis von Produktmodellen. In: Bauingenieur, 71 (1996), S. 47-55, Springer Verlag, 1996.

[Rüpp98] Rüppel, U.: Ganzheitliches Management von Bauprojekten aus Auftragsgebersicht auf der Basis integrierter Ablauf- und Kostensteuerung. In: Bauingenieur, 73 (1998), Nr. 3, S. 105-110, Springer Verlag, 1998.

[Sche02] Scheler, S.: Ein wissensbasierter Ansatz zur Kostenanalyse von Baustelleneinrichtungen. In: Forum Bauinformatik 2002, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen, Nr.181, 2002.

[Sche00] Scherer, R.; Nollau, C.; Buchwalter, J.; Scheler, S.: Produktinformationssysteme unterstützt durch dynamische Klassifikation und Ähnlichkeitsbasierte Suche. In: Hartmann, D. (Hrsg.): Objektorientierte Modellierung in Planung und Konstruktion. Deutsche Forschungsgemeinschaft, Forschungsbericht. Wiley-VCH GmbH, 2000.

[Sche91] Scheer, A. -W.: Wirtschaftsinformatik- Informationssysteme im Industriebetrieb. Springer-Verlag Berlin Heidelberg, 1991.

[Schm91] Schmidt, D.: Persistente Objekte und Objektorientierte Datenbanken. Hanser Verlag, 1991.

[Schm96] Schmidt, M.; Alm, W.: Vorgangsmodellierungssystem mit AIMS und ihr Einsatz in einer Bauverwaltung. In: Forum Bauinformatik, Cottbus '96, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 135, 1996.

[Schn96] Schneider, U.: Bearbeitung fachübergreifender Daten. In: Forum Bauinformatik, Cottbus '96, VDI-Fortschrittberichte, Reihe 4: Bauingenieurwesen Nr. 135, 1996.

[Schu95] Schultz, H. (Hrsg.): Integrierte Konstruktionsumgebung auf der Basis von Fertigungsfeatures. Dissertation, TH Darmstadt, Hanser Verlag, 1995.

[Seil85] Seiler, W.: Technische Modellierungs- und Kommunikationsverfahren für das Konzipieren und Gestalten auf der Basis der Modell-Integration. Fortschritts-Berichte, reihe 10 Nr. 49. VDI-Verlag, Düsseldorf, 1985.

[Sinz91] Sinz, E. J.: Objektorientierte Analyse. In: Wirtschaftsinformatik 33, (1991) 5, S. 455-457.

[Spur91] Spur, G.; Krause, F.-L.: Modellieren geometrisches. In: H.-J. Schneider (Hrsg.), Lexikon der Informatik und Datenverarbeitung, Oldenbourg Verlag München Wien, 1991.

[Spur94] Spur, G. (Hrsg.): Modellierungsmethoden für rechnerintegrierte Produktionsprozesse. Hanser Verlag, 1994.

[Stac73] Stachowiak, H.: Allgemeine Modelltheorie. Springer Verlag, Wien, 1973.

[Stah89] Stahlknecht, P.: Einführung in die Wirtschaftsinformatik. Springer-Verlag Berlin Heidelberg, 1989.

[Ston99] Stonebraker, M.: Objektrelationale Datenbanken: die nächste große Welle. Hanser Verlag München Wien, 1999.

[Suhm93] Suhm, A.: Produktmodellierung in wissensbasierten Konstruktionssystemen auf der Basis von Lösungsmustern. Reihe Konstruktionstechnik, Shaker Verlag, Aachen, 1993.

[Thal91] Thalheim, B.: Konzepte des Datenbank-Entwurfs. In: G. Vossen (Hrsg.), Entwicklungstendenzen bei Datenbanksystemen, Oldenbourg Verlag München Wien, 1991.

[Tori93] Toriya, H.; Chiyokura, H: 3D CAD Principles and Applications. Springer-Verlag , 1993.

[Trau91] Trautloft, R.; Lindner, U.: Datenbanken: Entwurf und Anwendungen. Verlag-Technik Berlin, 1991.

[Voss92] Vossen, G.: On Formal Models for Object-Oriented Databases. In: EMISA-Forum 2/1992.

[Voss99] Vossen, G.: Datenbankmodelle, Datenbanksprachen und Datenbankmanagementsysteme. 3. Auflage, Oldenbourg Verlag, 1999.

[Wann98] Wanner, G.: Entwurf eines Objektorientierten Datenbankmodells für relationale Datenbanksysteme. Dissertation, BTU Cottbus, Hundt Druck GmbH Köln, 1998.

[Warn95] Warnecke, G. (Hrsg.): Konzept zur Gestaltung prozess- und integrationsgerechter Produktmodelle. Produktionstechnische Berichte, Lehrstuhl für Fertigungstechnik und Betriebsorganisation, Universität Kaiserslautern, 1995.

[Wass00] Wassermann, K.; Heck, P.: Integration von raum- und Bauteilorientierter Daten in der Gebäudeplanung in einem zentralen Objektmodell. In: Hartmann, D. (Hrsg.): Objektorientierte Modellierung in Planung und Konstruktion. Deutsche Forschungsgemeinschaft, Forschungsbericht. Wiley-VCH GmbH, 2000.

[Webe93] Weber, B.; Drozella, H.: Objektorientierung bei der automatischen Nachweisführung nach DIN 18 800 (11.90). In: Bauingenieur 68 (1993) S. 411-417, Springer-Verlag, 1993.

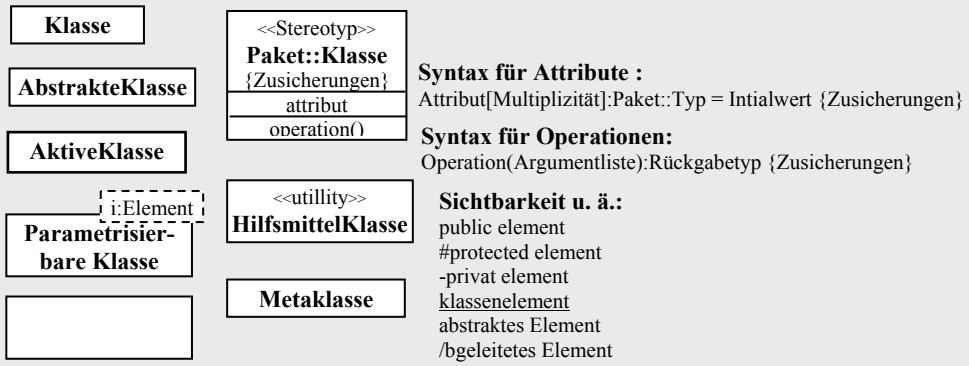
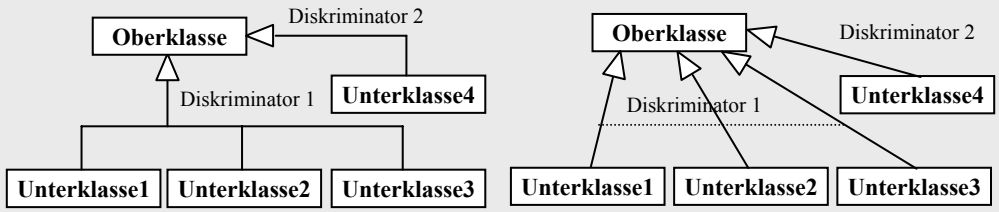
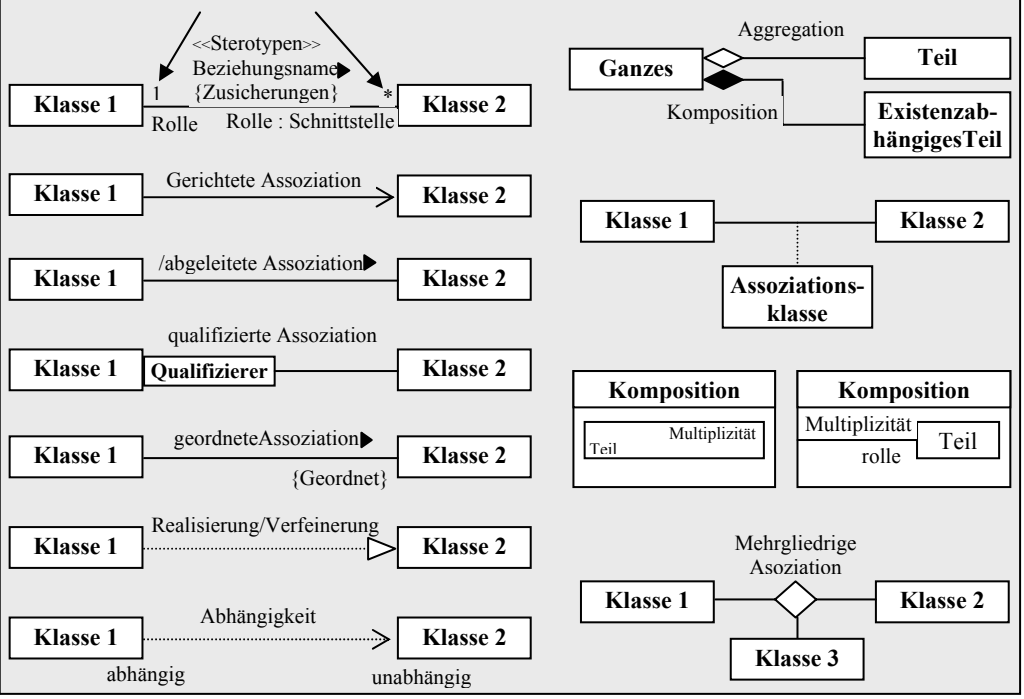
[Your89] Yourdon, E.: Modern Structured Analysis. Prentice-Hall Inc., 1989.

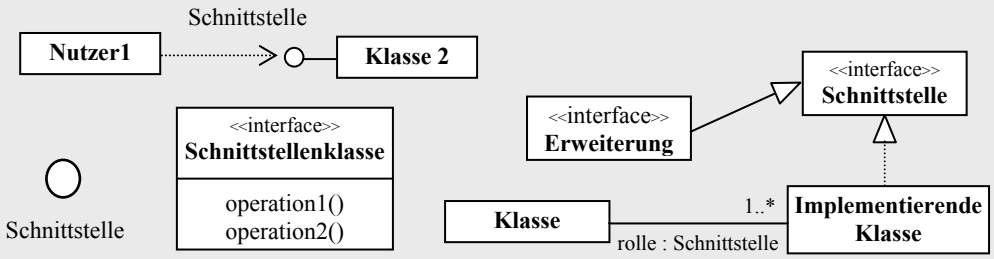
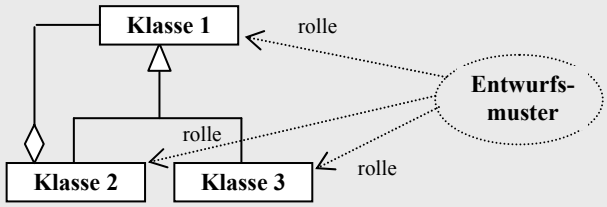
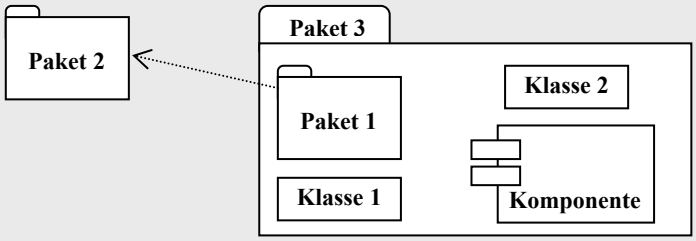
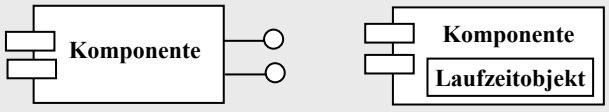
[Zang73] Zangenmeister, C.: Nutzwertanalyse in der Systemtechnik – Eine Methodik zur multidimensionalen Bewertung und Auswahl von Produktalternativen. Wittemannsche Buchhandlung, München, 1973.

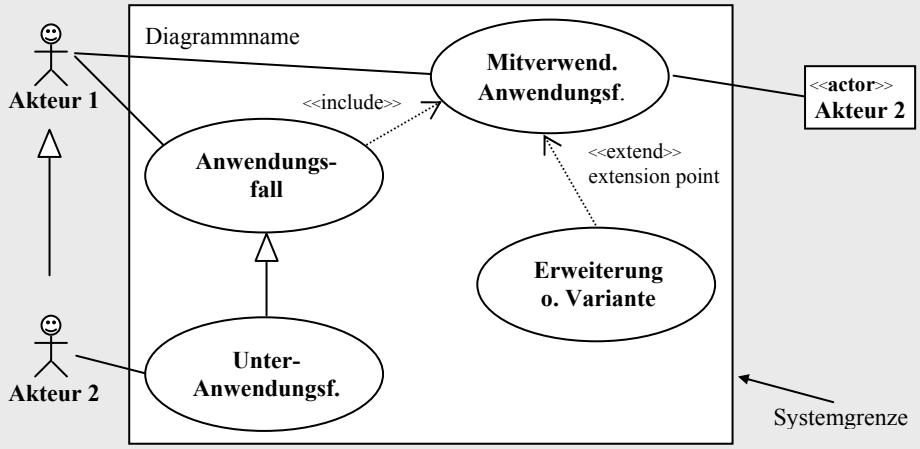
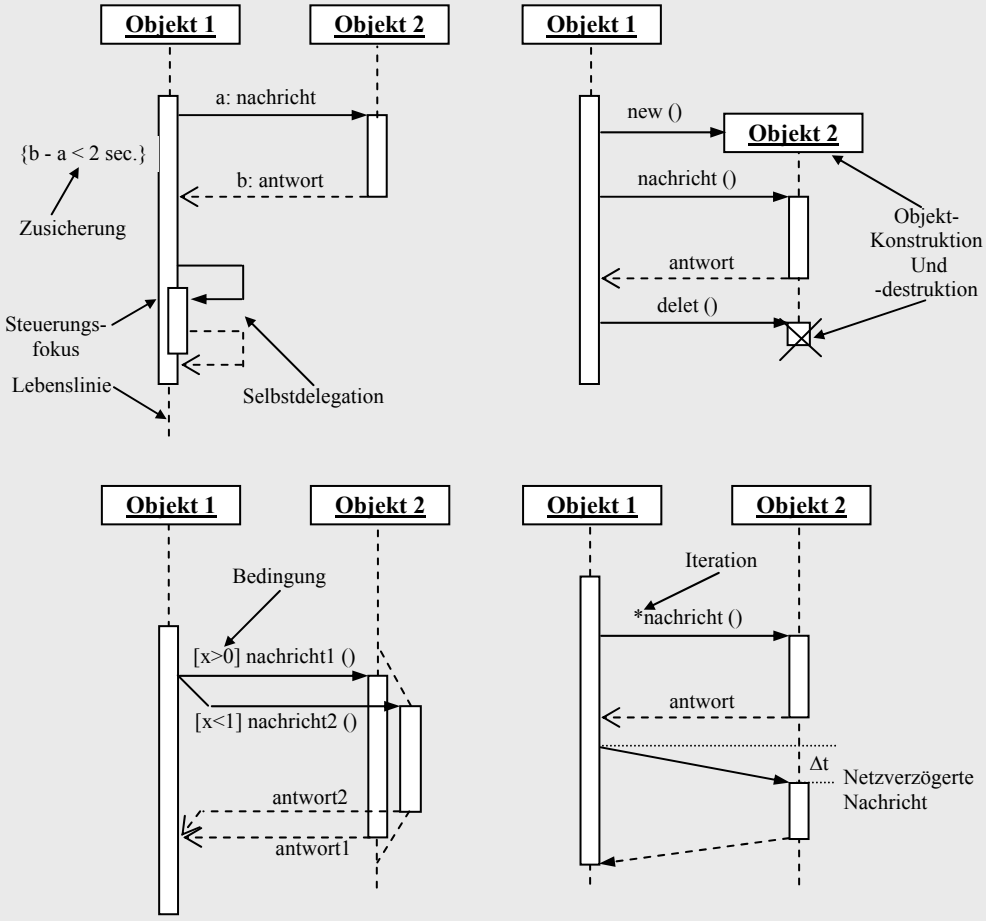
[Zimm90] Von Zimmermann, P.: Einsatz Objektorientierter Softwaretechnologie im Rechnungswesen. In: A.-W. Sheer (Hrsg.), Rechnungswesen und EDV, 11. Saarbrücker Arbeitstagung, Physica-Verlag Heidelberg, 1990.

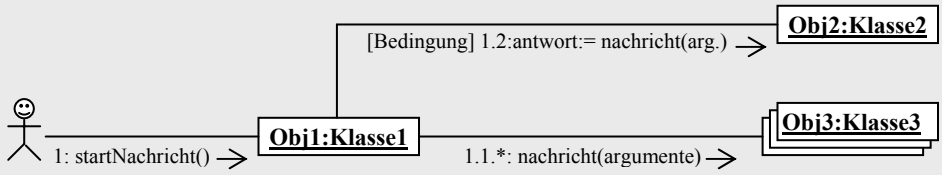
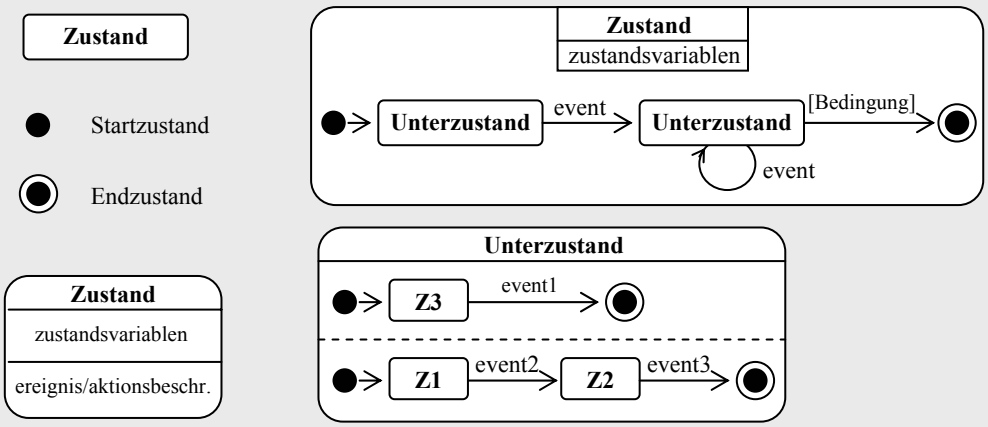
[Zöll91] Zöller, H.: Wiederverwendbare Software-Bausteine in der Automatisierung. VDI-Verlag, Düsseldorf, 1991.

Anhang: Unified Modeling Language (UML Version 1.3)

Begriff	Definition und Notation
Klasse	Der Begriff **Klasse** wird in UML 1.3 nicht neu definiert. 
Objekt	Der Begriff **Objekt** wird in UML 1.3 nicht neu definiert. 
Vererbung	Der Begriff **Vererbung** wird in UML 1.3 nicht neu definiert. 
Assoziationen	Eine **Assoziation** beschreibt eine Relation zwischen Klassen d.h. die gemeinsame Semantik und Struktur einer Menge von Objektbeziehungen. Es werden gerichtete Assoziationen (nur einseitig direkt navigierbar) und bidirektionale Assoziationen (beidseitig direkt navigierbar) unterschieden. Die beiden Enden einer Assoziation sind Assoziationsrollen. 

Begriff	Definition und Notation
Schnittstelle	Der Begriff **Schnittstelle** wird nicht in UML 1.3 neu definiert. **Schnittstellenklassen** sind abstrakte Klassen, die ausschließlich abstrakte Operationen definieren. Sie sind also Klassen, die mit dem Stereotype <<interface>> gekennzeichnet sind. Sie sind Spezifikationen des extern sichtbaren Verhalten von Klassen und beinhalten eine Menge von Signaturen für Operationen, die Klassen, die diese Schnittstelle bereitstellen wollen, implementiere müssen. 
Entwurfs-muster	Der Begriff **Entwurfsmuster** wird in UML 1.3 nicht neu definiert. 
Paket, Subsystem, System	**Pakete** sind Ansammlungen von Modellelementen beliebigen Typs, mit denen das Gesamtsystem in kleinere überschaubare Einheiten gegliedert wird. Ein Paket definiert Namensraum, d. h. innerhalb eines Paketes müssen die Namen der enthaltenen Elemente sein. Jedes Modellelement kann in anderen Paketen referenziert werden, gehört aber zu genau einem Paket. Pakete können wiederum Pakete beinhalten. Das oberste Paket beinhaltet das gesamte System. Ein **Subsystem** ist ein spezieller Form von Paket, dass einen unabhängigen Teil des ganzen zu modellierenden Systems beinhaltet. Ein **System** ist ein Paket, dass ganze modellierte System beinhaltet. 
Notiz	**Notizen** sind Kommentare zu einem Diagramm oder einem oder mehreren beliebigen Modellelementen ohne semantische Wirkung. 
	Eine **Komponente** ist ein ausführbares Softwaremodul mit eigener Identität und definierten Schnittstellen. 

Begriff	**Definition** und **Notation**
Anwendungs-falldiagramm	**Anwendungsfalldiagramme** zeigen die Beziehungen zwischen Akteuren und Anwendungsfälle. Ein **Anwendungsfall** beschreibt eine Menge konsistenter und zielgerichteter Interaktionen von Akteuren mit einem System, an deren Ende ein definiertes Ergebnis entstanden ist 
Sequenz-diagramme	Ein **Sequenzdiagramm** zeigt eine Menge von Interaktionen zwischen einer Menge von Objekte in einem Kontext unter Betonung der zeitlichen Abfolge. 

Begriff	Definition und Notation
Kollaborations-diagramm	Ein **Kollaborationsdiagramm** zeigt eine Menge von Interaktionen zwischen einer Menge von Objekte in einem Kontext unter Betonung der Beziehungen zwischen den Objekten und ihrer Topographie. Eine **Kollaboration** ist der Kontext einer Menge von Interaktionen. 
Zustands-diagramme	Ein **Zustandsdiagramm** zeigt eine Folge von Zuständen, die ein Objekt in seines Lebenslaufes einnehmen kann und aufgrund welcher Stimuli Zustandsänderungen stattfinden. Ein Zustandsdiagramm beschreibt eine hypothetische Maschine (endlicher Automat), die sich zu jedem Zeitpunkt in einer Menge endlicher Zustände befindet. Sie besteht aus einer endlichen, nicht leeren Menge von Zuständen, einer endlichen, nicht leeren Menge von Eingabesymbolen (Ereignissen), Funktionen, die den Übergang von einem Zustand in den nächsten darstellen, einen Anfangszustand und einer Menge von Endzustände. Ein **Zustand** ist eine Abstraktion der möglichen Attributwerte eines Objektes. 
Aktivitäts-diagramme	Ein **Aktivitätsdiagramm** ist eine spezielle Form des Zustandsdiagramms, das überwiegend oder ausschließlich Aktivitäten enthält. Es kann die Abläufe einer einzelnen Operationen näher spezifizieren oder einen Ablauf, an dem verschiedene Objekte beteiligt sind.