CARL HANSER VERLAG

Horst Neumann

Analyse und Entwurf von Softwaresystemen mit der UML

3-446-22038-0

www.hanser.de

5 Analyse des Systemverhaltens – Entwurf der Systemarchitektur

Ziel der Analyse des Systemverhaltens sind die operationellen Anforderungen, die sich auf den operationellen Betrieb des SW-Systems beziehen und die extern sichtbaren bzw. erkennbaren Wirkungen des SW-Systems bestimmen. Eine geeignete Methode zur Unterstützung der Analyse des Systemverhaltens, die sich im praktischen Einsatz bewährt hat, ist die Use-Case-Analyse.

Grundlage der Use-Case-Analyse sind die externen Objekte (Aktoren) und die Prozesse bzw. Vorgänge (Nutzungsfälle), die als Bestandteile des SW-Systems zur Erzeugung des gewünschten Systemverhaltens benötigt werden. Die Beziehungen zwischen Nutzungsfällen und die Beziehungen zu den relevanten Aktoren werden in Form von Diagrammen verdeutlicht. Wesentliche Aufgaben sind die Spezifikation der charakteristischen Merkmale der Nutzungsfälle und die Spezifikation der Interaktion der Nutzungsfälle mit den relevanten Aktoren in Form von Szenarios. Die Spezifikationen werden so weit ausgeführt, wie dies zur eindeutigen und vollständigen Beschreibung und zur Analyse des Systemverhaltens erforderlich ist.

Die Use-Case-Analyse wird bei Bedarf durch die Aktivitätenanalyse unterstützt. Dabei wird die Zuordnung von Aktivitäten zu übergeordneten Verantwortungen (Geschäftsobjekten) und der Fluss der Aktivitäten analysiert. Für die Darstellung werden spezielle Formen von Zustandsdiagrammen genutzt, die im Kapitel 6 behandelt werden.

An die Analyse des Systemverhaltens schließt sich der Entwurf der Systemarchitektur an. Dabei wird eine logische und physikalische Systemstruktur für die Zuordnung der Modelle konzipiert, die im Verlauf der Anforderungsanalyse und des Entwurfs entstehen.

Zunächst wird der generelle Prozess der Analyse skizziert. Die nachfolgenden Abschnitte sind nach den folgenden Themen gegliedert:

- Analyse der Nutzungsfälle (Use-Case-Analyse), d. h. Strukturierung, Spezifikation und Analyse der Nutzungsfälle und der zugeordneten Szenarios;
- Entwurf der Systemarchitektur, d. h. Konzipierung der logischen und physikalischen Systemstruktur.

Zuerst werden jeweils die Konzepte und Modellelemente und die resultierenden Modelle erläutert, soweit diese für die genannten Themen relevant sind. Danach wird die Vorgehensweise zur Konzipierung der Modelle beschrieben, welche die Grundlage für die Analyse bzw. den Entwurf bilden.

5.1 Prozess der Analyse

Charakteristik

Ziel der Analyse ist es, in einer strukturierten, eindeutigen und verständlichen Form darzustellen und zu beschreiben, *was* das zu entwickelnde SW-System leisten soll. Untersuchungen zum Thema, *wie* die Leistung mit den Mitteln der EDV erbracht werden kann, werden zunächst zurückgestellt oder nur dann ansatzweise durchgeführt, wenn die *Erfüllbarkeit* von Anforderungen strittig ist.

Gegenstand der Analyse sind der erwünschte Nutzeffekt des SW-Systems, das resultierende Systemverhalten und die relevanten Anforderungen. Eine besondere Bedeutung haben in diesem Zusammenhang der Problem- und Anwendungsbereich (Domäne) des SW-Systems, aber auch die projektspezifischen Rahmenbedingungen, insbesondere die verfügbaren Ressourcen.

Im Prinzip lassen sich folgende Arten der Analyse unterscheiden:

- Analyse des Problem- und Anwendungsbereichs (Domäne-Analyse)
- Ist-/Soll-Analyse der Prozesse und Vorgänge (Anwendungsanalyse)
- Anforderungsanalyse.

Die Analyse des Problem- und Anwendungsbereichs und die Ist-/Soll-Analyse der aktuellen Prozesse mit Konzipierung von Möglichkeiten zu ihrer Umgestaltung und Modernisierung durch die Nutzung der Datenverarbeitung sind ein wesentlicher Bestandteil der Konzeptphase. Dabei bieten sich Methoden an, die auf unterschiedlichen Konzepten zur Unterstützung des kreativen Gestaltungsprozesses basieren.

Die Anforderungsanalyse wird im Verlauf der Entwicklung des SW-Systems durchgeführt. Ihre Ergebnisse bilden die Grundlage für den Entwurf und die Realisierung des SW-Systems. Charakteristisch sind folgende Aktivitäten:

- die Erfassung und Dokumentation des Zwecks des SW-Systems: geforderter erwünschter Nutzeffekt sowie die Leistungen, die das System erbringen soll;
- die Erfassung bzw. Ableitung und Dokumentation der diversen Anforderungen an das SW-System;
- die Dekomposition des Anwendungsbereichs bzw. des SW-Systems in überschaubare Unterbereiche bzw. Subsysteme;
- Umsetzung der Anforderungen in geeignete Konzepte, welche auf die beabsichtigte Erfüllung der Anforderungen zugeschnitten sind;
- die Konzipierung von Modellen zur Analyse der diversen Anforderungen auf Widerspruchsfreiheit, Vollständigkeit und Erfüllbarkeit;
- die Validation der Analysemodelle.

Analysephasen

Die Anforderungsanalyse wird nicht in einer durchgängigen Makrophase durchgeführt, sondern sie ist in den Entwicklungsprozess zur iterativ-rekursiven bzw. evolutionären

Entwicklung eines SW-Systems eingebunden, der im Kapitel 2.2 behandelt wurde. Dabei werden zweckmäßig folgende Mikrophasen unterschieden:

- Analyse des Systemverhaltens
- Analyse der funktionalen Anforderungen
- Analyse der Realisierungsforderungen.

An jede Analysephase schließt sich nahtlos eine Entwurfsphase an (vgl. Abbildung 2.2-6 in Abschnitt 2.2.6), wobei die Aktivitäten in den beiden Phasen stark voneinander abhängen. Im Verlauf des Entwurfs können bei Bedarf Analyseaktivitäten nachgeholt oder vervollständigt werden.

Die *Analyse des Systemverhaltens* bildet die Voraussetzung für den *Entwurf der Systemarchitektur*. Sie ist die erste Phase im Entwicklungsprozess, wenn man die Vorphasen außer Betracht lässt. Wesentliche Aspekte sind die technischen Prozesse bzw. Geschäftsprozesse und die relevanten Aktionen und Reaktionen bzw. Aktivitäten und Arbeitsabläufe (work flow), die durch das SW-System unterstützt werden. Für die Entwicklung des SW-Systems hat die Analyse des Systemverhaltens eine herausragende und entscheidende Bedeutung.

Folgende *Methoden* zur Unterstützung der Analyse des Systemverhaltens haben sich im praktischen Einsatz bewährt:

- die Analyse der Nutzungsfälle (Use-Case-Analyse)
- die Analyse der Arbeitsabläufe (Aktivitäten- und Work-flow-Analyse)
- die Analyse der relevanten Systemzustände und der Reaktionen auf externe Ereignisse (Reaktionsanalyse).

Die *Analyse der Nutzungsfälle* und der sich anschließende *Entwurf der Systemarchitektur* zur Einbindung der Analysemodelle werden in den folgenden Abschnitten im Detail behandelt.

Für die Analyse der Nutzungsfälle spielen die systeminternen Objekte eine untergeordnete Rolle. Dagegen treten diese bei der *Analyse der funktionalen Anforderungen* in den Vordergrund der Betrachtung. Eine wesentliche Rolle für die Abstraktion und Modellbildung spielen deshalb hier die objektorientierten Konzepte, die auch für die *Analyse der realisierungsbedingten Anforderungen* relevant sind. Die Analyse der funktionalen und der realisierungsbedingten Anforderungen wird im Kapitel 6 unter der Überschrift „Objektorientierte Analyse“ behandelt.

Die Aktivitäten- und Work-flow-Analyse und die Reaktionsanalyse ergänzen die Use-Case-Analyse, da jeweils spezielle Aspekte des SW-Systems betrachtet werden. Die Work-flow-Analyse wird insbesondere bei betriebswirtschaftlichen und organisatorischen Problemstellungen eingesetzt, während die Reaktionsanalyse im Kontext von technischen Systemen eine wertvolle Unterstützung bietet. Beide Formen der Analyse nutzen Verfahren, die auch bei der Analyse der funktionalen Anforderungen angewendet werden. Sie werden deshalb im Zusammenhang mit der objektorientierten Analyse im erforderlichen Detail erläutert.

Den Schwerpunkt der nachfolgenden Darstellung bilden jeweils die Entwicklungsaktivitäten, d. h. die übergeordneten organisatorischen, planenden und kontrollierenden Aktivitäten werden bei der Behandlung weitgehend ausgeklammert.

5.2 Analyse der Nutzungsfälle (Use-Case-Analyse)

5.2.1 Einführung

Durch einen Nutzungsfall wird ein Ausschnitt aus dem extern sichtbaren bzw. erkennbaren Verhalten des SW-Systems spezifiziert, ohne dass dabei irgendwelche Realisierungsaspekte berücksichtigt werden. Die Gesamtheit der Nutzungsfälle beschreibt demnach genau das Systemverhalten, das für den Nutzer des Systems relevant ist und deshalb seinen Vorstellungen und Erwartungen entsprechen muss.

Aus den Nutzungsfällen ist das korrespondierende innere Verhalten des SW-Systems und damit letzten Endes die innere Funktionalität und der innere Ablauf ableitbar, der das äußere Verhalten herbeiführt.

Ausgangsdokumente

Die Darstellung in den nachfolgenden Abschnitten beruht auf der Annahme, dass die Analyse des Systemverhaltens auf den Ergebnissen einer *Konzeptphase* und einer sich anschließenden *Projekt-Definitionsphase* aufsetzt – vgl. Abschnitt 2.2. Im Verlauf dieser Vorphasen werden relevante Informationen gesammelt und gesichtet, Voruntersuchungen durchgeführt und das Entwicklungsprojekt geplant. Dabei entstehen Unterlagen, wie sie nachfolgend aufgeführt sind:

- die globale Beschreibung des SW-Systems: Zweck, wesentliche Aufgaben;
- die Beschreibung und Darstellung des Istzustands: Organisation, Rollenverteilung und Zuständigkeiten, technische Prozesse bzw. Geschäftsprozesse, zugeordnete Vorgänge bzw. Arbeitsabläufe, relevante Fragebögen und Besprechungsprotokolle;
- die Problembeschreibung;
- der Zielkatalog;
- eine *Vision* des zu entwickelnden Systems, d. h. ein operationelles Konzept und eine darauf abgestimmte Systemkonfiguration, z. B.
  - ein konkretes oder abstrahiertes Modell des Problem- und Anwendungsbereichs in leicht verständlicher Form, das durch die vorgesehenen Systemnutzer bzw. Systemexperten im Dialog mit den Entwicklern erstellt wurde
  - und/oder Konzepte der Benutzeroberfläche und Lösungsansätze bzw. experimentelle Prototypen mit Bewertungen;
- die Benutzerforderungen und relevanten Vorschriften und Richtlinien (Pflichtenheft);

- die Darstellung der Entwicklungsbasis: Hardware und Applikationen, die weiter genutzt oder eingesetzt werden sollen, existierende Prototypen von Teilsystemen, relevante Informations- und Ablaufmodelle, verfügbare Klassenbibliotheken bzw. Application Frameworks;
- Projekthandbuch und Entwicklungsplan mit der Risikoabschätzung;
- ein Verzeichnis der verfügbaren Dokumente und der relevanten Ansprechpartner.

Für die weiterführenden, im vorliegenden Buch behandelten Analyseaktivitäten werden die genannten Unterlagen, soweit sie im Einzelfall vorhanden sind, als Grundlage vorausgesetzt.

Ausführende

Die diversen Aktivitäten der Analyse erfordern Ausführende mit unterschiedlichen Erfahrungen und Kenntnissen, nämlich:

- *Domain-, System- und Anwendungsexperten* (i. f. kurz *Experten*), die den Problem- und Anwendungsbereich, das aktuelle System, die technischen Prozesse bzw. die Geschäftsprozesse und die aktuellen Abläufe bzw. Vorgänge kennen. Besonders wichtig ist, dass sie auch die technischen und wirtschaftlichen Engpässe und die Ansatzpunkte für die Überwindung der Engpässe durch den Einsatz der EDV erkennen.
- *System- und Anwendungsentwickler* (i. f. kurz *Entwickler*), welche die Möglichkeiten und den Einsatz der DV-Technologie zur Unterstützung und Automatisierung von Systemabläufen bzw. Geschäftsprozessen kennen.

Besonders wichtig ist die Expertise der System- und Anwendungsexperten, die deshalb die Analyse des Systemverhaltens durchführen sollten. Dies würde aber bedeuten, dass diese Erfahrungen in der Anwendung der Analysemethoden haben müssten, die auf der UML basieren und durch rechnergestütztes Werkzeug gut unterstützt werden. Die tägliche Praxis zeigt, dass diese Voraussetzung in der Regel nicht erfüllt ist.

Die Analyse wird deshalb zweckmäßig von den *Entwicklern* durchgeführt. Dies bedeutet aber, dass die Erfahrungen und Kenntnisse der Experten durch eine intensive Befragung und Beratung eingebracht und die Ergebnisse der Analyse von den Experten bewertet werden müssen. Es besteht sonst die Gefahr, dass die Entwickler das Verhalten des SW-Systems am Bedarf vorbei nach ihren eigenen Vorstellungen (und Phantasien) modellieren und spezifizieren.

Da sich die Experten und die Entwickler durch den Standpunkt der Systembetrachtung und durch ihr Verständnis der verwendeten Fachbegriffe unterscheiden, ist ein Abstimmungsprozess erforderlich. Eine wichtige Rolle spielt in diesem Zusammenhang die präzise Definition der Begriffe, die im Anwendungsbereich des SW-Systems gebräuchlich sind. Es zeigt sich nämlich immer wieder, dass gerade alltägliche Begriffe, die von den Experten als *selbstverständlich* angesehen werden, von den Entwicklern unterschiedlich verwendet bzw. interpretiert werden. Ein typisches Beispiel ist der

Begriff *Bestellung*. Was ist gemeint: eine Liste mit Artikeln, ein verbindlicher Kaufauftrag, ein Bestellformular auf dem Bildschirm?

Die Begriffe müssen deshalb mit allen Systemanwendern (Fachbereichen) abgestimmt werden, damit ihre einheitliche Verwendung sichergestellt ist. Die abgestimmten Begriffe werden zweckmäßig in einem *Begriffslexikon* zusammengestellt. Neben der präzisen Beschreibung der Bedeutung sollte auch der Bezug zu verwandten Begriffen (Unterbegriffe, Rollen) hergestellt werden. Die Begriffe repräsentieren unter Anderem relevante Aktoren und Entity-Objekte mit speziellem Informationsgehalt und korrespondierender Funktionalität.

Die erfolgreiche Zusammenarbeit zwischen den Entwicklern und Experten wird durch eine Reihe von bewährten Verfahren der Teamarbeit unterstützt. Dazu gehören die Nutzung von Präsentations- und Diskussionstechniken, Brainstorming, Assoziationstechniken, Workshops etc.

Wesentliche Ergebnisse

Die wesentlichen Ergebnisse der Analyse des Systemverhaltens sind:

- das *Begriffslexikon*
- die *Anforderungsspezifikation* mit folgenden Bestandteilen:
  - präzise Beschreibung der Anforderungen bzw. der Leistungen, die vom System erfüllt bzw. erbracht werden müssen
  - Ergebnisse der Use-Case-Analyse: Strukturierung und Spezifikationen der Nutzungsfälle und der zugeordneten Szenarios
  - Ergänzungen: Ergebnisse der Workflow- und der Reaktionsanalyse
- der *Validationsbericht* mit den Ergebnissen der Validation der Anforderungsspezifikation.

Die Ergebnisse der Analyse des Systemverhaltens bilden eine solide Grundlage für die objektorientierte Entwicklung des SW-Systems. Weitere Einzelheiten dazu werden im Abschnitt 5.2.5 erläutert.

5.2.2 Relevante Modellelemente: Aktor, Use-Case

Die wesentliche Grundlage der Use-Case-Analyse bilden die Modellelemente *Aktor* und *Use-Case (Nutzungsfall)*. Ihr Zusammenwirken wird in Use-Case-Diagrammen sichtbar gemacht.

Aktor

Ein Aktor repräsentiert ein *externes Objekt*, das der Systemumgebung zugeordnet ist und als aktiver Client oder als passiver Server oder in wechselnden Rollen mit dem SW-System interagiert. Dabei ist lediglich das Verhalten des Aktors im Zusammenwirken mit dem System relevant. Konkrete Aktoren mit gleichartigem Verhalten werden

Abbildung 5.2-1: Abbildung eines Aktors

durch eine Aktorklasse spezifiziert, von der konkrete Aktoren als Instanzen ableitbar sind.

Typische Instanzen von Aktoren sind:

- Benutzer von Applikationen des SW-Systems in einer bestimmten Rolle;
- periphere Geräte und externe DV-Systeme, die mit dem Zielrechnersystem verbunden sind.

Für einen Aktor ist in der UML kein spezielles grafisches Symbol definiert. Es wird das normale Klassensymbol in Verbindung mit einem Stereotyp verwendet. Vordefiniert ist der Stereotyp «actor», dem ein spezielles Ikon in Form eines Strichmännchens zugeordnet ist. Das Ikon stellt im Prinzip eine Person dar, die das SW-System nutzt. Klassen, die periphere Geräte bzw. externe DV-Systeme repräsentieren, können durch einen speziellen Stereotyp gekennzeichnet sein. Beispiele für Aktoren sind in Abbildung 5.2-1 dargestellt.

In einem interaktiven System sind die Benutzer, die mit dem System einen Dialog führen, die wesentlichen Aktoren. Im Allgemeinen haben die Benutzer verschiedene Rollen, d. h. sie repräsentieren Aktoren mit unterschiedlichem Verhalten. Dabei kann eine bestimmte Person auch in verschiedenen Rollen auftreten, z. B. zunächst als *Mitarbeiter*, bei Bedarf als *Operator*.

Aktorklassen mit Eigenschaften, die sich teilweise überdecken, können in eine Vererbungsstruktur eingebunden werden. Abbildung 5.2-2 zeigt ein Beispiel. *Benutzer* ist darin eine *abstrakte* Aktorklasse, die einen Benutzer mit Zugriffsrecht zur Durchführung allgemeiner Aufgaben repräsentiert. *Manager*, *Mitarbeiter* und *Operator* sind dagegen Klassen, aus denen konkrete Benutzer mit speziellen Aufgaben ableitbar sind.

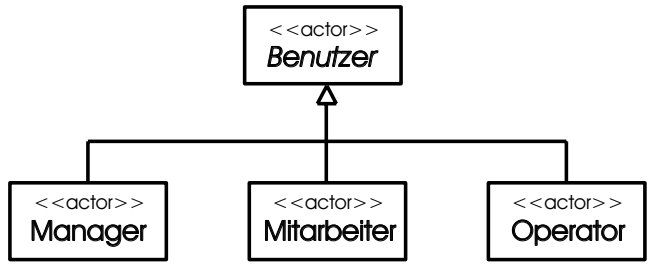

Abbildung 5.2-2: Struktur aus Aktoren

Zur Implementierung der Benutzerrechtsregelung oder für andere Zwecke müssen Daten der verschiedenen Benutzer verwaltet werden, sodass in einem realen SW-System den Aktoren zumeist auch *interne* Objekte zugeordnet sein müssen.

Benutzer sind für das System *direkte* oder *indirekte* Aktoren, je nachdem, ob sie das System unter Anwendung von adäquaten Eingabe- und Ausgabegeräten direkt nutzen oder die Nutzung nur beeinflussen, wie z. B. ein Kunde, der sich beraten lässt. Außerdem werden *primäre* und *sekundäre* Aktoren unterschieden. Erstere nutzen die Funktionalität, die das System unterstützt, Letztere tragen zur Aufrechterhaltung der Funktionalität bei. Beispiel: Verkäufer bzw. Systemverwalter.

Das SW-System benötigt für die Kommunikation mit einem Aktor eine Schnittstelle, z. B. eine grafische Benutzeroberfläche oder einen Treiber, wenn der Aktor ein Gerät repräsentiert. Die Schnittstellen, die wesentliche Bestandteile des zu entwickelnden SW-Systems darstellen, werden im weiteren Verlauf der Entwicklung als *Boundary-Klasse* bzw. als ein Aggregat aus Klassen konzipiert und realisiert.

Use-Case (Nutzungsfall)

Use-Case lässt sich sinngemäß mit *Nutzungsfall* oder auch mit *Anwendungsfall* übersetzen, wobei im vorliegenden Buch der erste Begriff bevorzugt wird.

Ein Nutzungsfall repräsentiert einen abgegrenzten *Vorgang* im SW-System, der ein für einen externen Beobachter erkennbares Verhalten bzw. eine überprüfbare Wirkung des SW-Systems definiert. Dies wird dadurch erreicht, dass als wesentlicher Bestandteil eines Nutzungsfalls logisch bzw. zeitlich zusammengehörige Abfolgen von extern sichtbaren Aktionen bzw. Reaktionen beschrieben werden, die im SW-System spontan oder durch die Wirkung von Aktoren ausgelöst werden. Das SW-System wird dabei im Prinzip als „black box“ betrachtet, sodass im Wesentlichen die geforderte operationelle Nutzung des Systems beschrieben wird. Die internen funktionalen Abläufe des SW-Systems und Aspekte der Realisierung bleiben unberücksichtigt!

Typische Nutzungsfälle sind:

- ein Vorgang bzw. eine Transaktion, der/die für den Benutzer ein Ergebnis von Wert erzeugt, und der dafür erforderliche interaktive Dialog zwischen dem Benutzer und dem SW-System;
- ein Vorgang, der einen externen Prozess steuert, und der dafür erforderliche Transfer von Signalen bzw. Daten von/zu peripheren Geräten.

Aus der Sicht des SW-Systems ist ein Nutzungsfall gekennzeichnet durch einlaufende Ereignisse sowie durch Zustandswechsel und Aktionen, die durch die Ereignisse ausgelöst werden, also durch Zustand und Verhalten. Ein Nutzungsfall ist demnach ein Teil des SW-Systems, das spezielle Objekte einschließt. Dies bedeutet, dass Nutzungsfälle mit gleichartigem Verhalten im Sinne einer Klasse spezifiziert werden können, von der konkrete, d. h. tatsächlich *ausführbare* Nutzungsfälle als Instanzen ableitbar sind. Die konkreten Nutzungsfälle interagieren mit konkreten Aktoren (Instanzen von

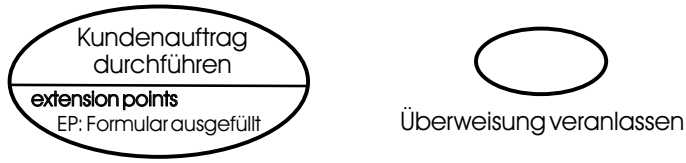

Abbildung 5.2-3: Abbildung eines Use-Case

Aktorklassen), wobei die Interaktionen durch bestimmte Ereignisse ausgelöst werden und den Austausch konkreter Signale und/oder Daten bewirken. Nutzungsfälle mit vergleichbarem Verhalten können in eine Vererbungsstruktur eingebunden werden.

Das durch die UML vordefinierte grafische Symbol eines Nutzungsfalls ist eine Ellipse mit einem durchgezogenen Rand in normaler Strichstärke. Die Ellipse ist in die folgenden horizontalen Abschnitte (compartments) unterteilt:

- ein Abschnitt mit einer Kurzbezeichnung, einem optionalen Stereotyp und optionalen Markierungen bzw. Beschränkungen;
- ein Abschnitt, der mit **extension points** überschrieben ist. Er enthält eine Liste mit Erweiterungspunkten, wobei jeder Erweiterungspunkt einen Namen und eine Beschreibung der Position für die Einfügung einer Sequenz von Aktionen eines anderen Nutzungsfalls umfasst, zu dem eine Nutzungsbeziehung besteht;
- ein Abschnitt mit einer Liste mit Merkmalen (z. B. Attributen, Operationen), die den Nutzungsfall näher spezifizieren.

Relevant ist der Abschnitt mit der Kurzbezeichnung, die mehrere Wörter umfassen kann. Die übrigen Abschnitte können entfallen. Ist dies der Fall, kann das grafische Symbol verkleinert und der Bezeichner unterhalb des Symbols an zentraler Position angezeigt werden. Beispiele für Nutzungsfälle sind in Abbildung 5.2-3 dargestellt.

Ein Nutzungsfall lässt sich in mehrere Nutzungsfälle unterteilen, wenn er z. B. ein Subsystem oder einen komplexen Prozess repräsentiert. Es resultiert eine hierarchische Struktur aus Nutzungsfällen mit mehreren Ebenen. Die unterste Ebene der Struktur bilden die *elementaren* Nutzungsfälle, die jeweils einen abgegrenzten Vorgang repräsentieren.

Ein nicht-elementarer Nutzungsfall hat ein charakteristisches Verhalten, das zweckmäßig in einer *Spezifikation des Nutzungsfalls* erfasst wird. Dazu gehören z. B. die Ereignisse, die den Nutzungsfall auslösen, die wesentlichen Ergebnisse, die der Nutzungsfall erzeugt, sowie mögliche Fehlerfälle und Ausnahmesituationen.

Für einen elementaren Nutzungsfall sind die Aktionen und Reaktionen des SW-Systems im Zusammenwirken mit den relevanten Aktoren charakteristisch. Die resultierende Vorgangsbeschreibung wird als *Szenario* bezeichnet. Für die von den Aktoren bzw. vom SW-System übermittelten Signale und Daten werden repräsentative Namen verwendet, d. h. die Spezifikation eines Szenarios repräsentiert eine Abstraktion der

möglichen konkreten Szenarios. Für die Spezifikation eines Szenarios bieten sich auch andere Formen an, z. B. Zustands- bzw. Ablaufdiagramme.

Ein Nutzungsfall besteht demnach im Allgemeinen aus:

- dem grafischen Symbol mit dem genannten Inhalt
- einer übergeordneten und/oder nachgeordneten Struktur
- einer übersichtlichen Spezifikation des charakteristischen Verhaltens
- den Spezifikationen der zugeordneten Szenarios.

Die Strukturierung von Szenarios wird dadurch unterstützt, dass Nutzungsfälle mit (unvollständigen) Teilszenarios gebildet werden können (siehe Abschnitt 5.2.3) und primäre und sekundäre Szenarios unterschieden werden.

Ein primäres Szenario entsteht unter der Annahme, dass der Nutzungsfall unter idealen Bedingungen abläuft. Die identifizierten Fehlerfälle (z. B. Eingabefehler oder Gerätefehler) und Ausnahmesituationen (z. B. Überlauf eines Eingabepuffers) bleiben unberücksichtigt. Die zu ihrer Kompensation jeweils vorgesehenen Reaktionen werden in einem sekundären Szenario dargestellt. Auf diese Weise wird eine klare Trennung der Spezifikation des regulären bzw. erwünschten Systemverhaltens gegen die Spezifikation des Fehlverhaltens erreicht. Die beiden Aspekte können dann auch bei der weiteren Entwicklung getrennt voneinander betrachtet werden. Dies ist allein schon deshalb von Vorteil, weil viele möglichen Fehlerquellen erst später bei der Analyse der Realisierungsforderungen im Detail erfasst und erst dann geeignete Gegenmaßnahmen konzipiert werden können.

Zusammenfassend lässt sich also feststellen, dass ein definierter Nutzungsfall durch ein Netz von Szenarios näher spezifiziert wird.

5.2.3 Modell der Systemnutzung

Das Modell der Systemnutzung resultiert aus einer Analyse des SW-Systems vom Standpunkt eines externen Betrachters. Dabei werden die operationellen Anforderungen analysiert, das extern sichtbare bzw. erkennbare Verhalten des Systems abgeleitet und dieses durch die Bildung einer geeigneten Struktur aus Nutzungsfällen und in Form von zugeordneten Szenarios spezifiziert.

Die Bestandteile des Modells der Systemnutzung sind:

- die identifizierten Nutzungsfälle und ihre strukturellen Abhängigkeiten
- Use-Case-Diagramme
- die Spezifikationen der Nutzungsfälle
- die Spezifikationen der zugeordneten Szenarios.

Ein Use-Case-Diagramm ist ein spezieller Graph, in dem Nutzungsfälle mit Aktoren und mit anderen Nutzungsfällen durch definierte Beziehungen verbunden sind. Durch das Diagramm wird dargestellt, wie die identifizierten Nutzungsfälle im SW-System mit den zugeordneten Aktoren interagieren und wie die Nutzungsfälle mit anderen

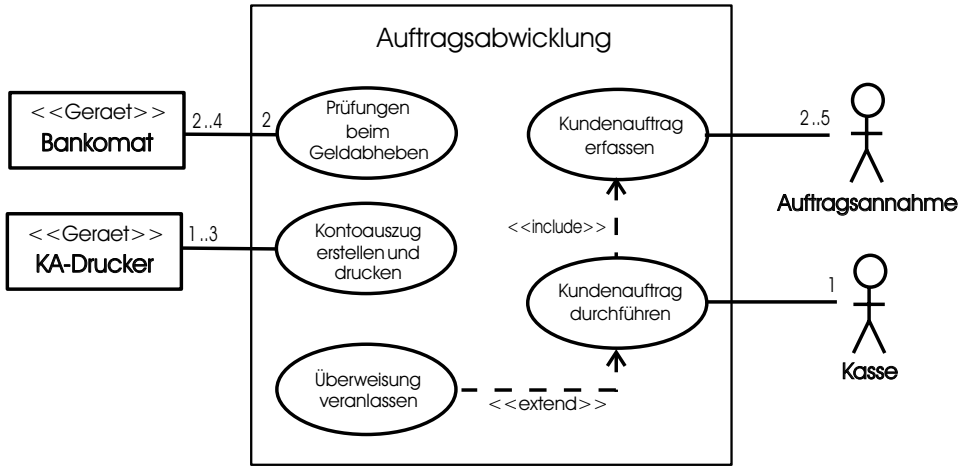

Abbildung 5.2-4: Use-Case-Diagramm: Auftragsabwicklung in einer Bank

Nutzungsfällen verknüpft sind. Die Nutzungsfälle können in ein Rechteck eingeschlossen werden, das die systembezogene Begrenzung (DV-System, Subsystem, Prozess) repräsentiert.

Ein Use-Case-Diagramm hat einen optionalen Namen. Es ist Bestandteil eines übergeordneten Nutzungsfalls oder eines Pakets, das den Gültigkeitsbereich für die identifizierbaren Elemente des Diagramms bestimmt. Abbildung 5.2-4 zeigt ein typisches Beispiel. Das große Rechteck verdeutlicht den Kontext des Diagramms, die darin eingebundenen Ellipsen repräsentieren Nutzungsfälle, die äußeren Strichmännchen bzw. Klassensymbole die zugeordneten Aktoren.

Die Beteiligung eines Aktors an einem Nutzungsfall wird durch eine einfache abstrakte Nutzungsbeziehung (Assoziation) spezifiziert. Das grafische Symbol ist eine durchgezogene Verbindungslinie zwischen dem Aktor und dem Nutzungsfall mit optionalem Richtungspfeil zur Kennzeichnung eines unidirektionalen Signal-/Datenflusses. Die Beziehung wird als Kommunikationsbeziehung interpretiert. Sie kann mit relevanten Kennungen (Multiplizität, Rolle) beim Aktor und Nutzungsfall versehen werden. Fehlt die Angabe einer Multiplizität, wird der Wert „1“ angenommen.

Zwischen den Nutzungsfällen auf einer bestimmten Ebene der Struktur bestehen Abhängigkeiten. Für diesen Zweck sind zwischen zwei Nutzungsfällen die nachfolgenden Abhängigkeitsbeziehungen definiert:

- *include-Beziehung*
  Ein spezieller Nutzungsfall *Ns* wird auf der Grundlage eines bereits definierten allgemeinen bzw. unvollständigen Nutzungsfalls *Na* spezifiziert. Dies entspricht der Vorstellung, dass der allgemeine bzw. unvollständige Nutzungsfall *Na* in den Nutzungsfall *Ns* eingefügt wird. Die Beziehung wird gemäß Abbildung 5.2-5 als

Abbildung 5.2-5: Einfügungsbeziehung zwischen Nutzungsfällen (*include*-Beziehung)

eine spezialisierte Form der Nutzungsbeziehung mit einem Richtungspfeil zum Nutzungsfall *Na* gezeichnet und durch ein Label mit dem vordefinierten Stereotyp «include» gekennzeichnet. Die genaue Einfügungsstelle (extension point) in der Sequenz der Aktionen von *Ns* kann gemäß Abbildung 5.2-3 im Abschnitt „extension points" von *Ns* bzw. im Szenario von *Ns* angegeben werden.
In Abbildung 5.2-5 ist *Kundenauftrag erfassen* ein unvollständiger Nutzungsfall. Erst in Verbindung mit dem Nutzungsfall *Kundenauftrag durchführen* wird ein brauchbares Ergebnis erzeugt.
Mit Hilfe der include-Beziehung können Nutzungsfälle, die sich in der Abfolge ihrer Aktionen überdecken, redundanzfrei spezifiziert werden. Die Korrektur der Nutzungsfälle bzw. ihre Anpassung an neue Anforderungen gestalten sich dadurch einfacher.

- *extend-Beziehung*
  Ein Nutzungsfall *Ns* wird durch einen anderen Nutzungsfall *Ne* erweitert, wenn eine bestimmte Bedingung erfüllt ist. Die Beziehung wird gemäß Abbildung 5.2-6 als eine spezialisierte Form der Nutzungsbeziehung mit einem Richtungspfeil zum Nutzungsfall *Ns* dargestellt und durch den Stereotyp «extend» markiert. Die Bedingung für die Erweiterung kann an den Richtungspfeil angefügt bzw. dem Szenario des Nutzungsfalls *Ne* vorangestellt werden. Die genaue Einfügungsstelle (extension point) in der Sequenz der Aktionen von *Ns* kann im Abschnitt „extension points" von *Ns* bzw. im Szenario von *Ns* angegeben werden. Der Nutzungsfall *Ns* ist unabhängig von *Ne* spezifiziert, d. h. er kann mit oder ohne die Erweiterung ausgeführt werden.

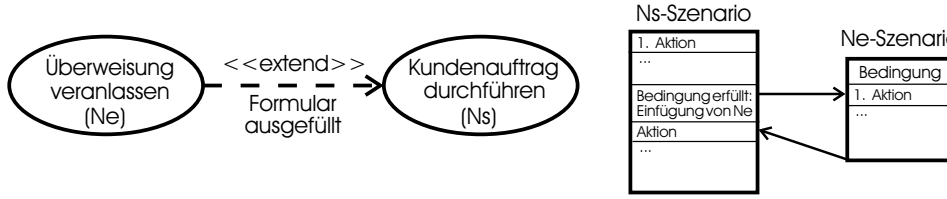

Abbildung 5.2-6: Erweiterungsbeziehung zwischen Nutzungsfällen (*extend*-Beziehung)

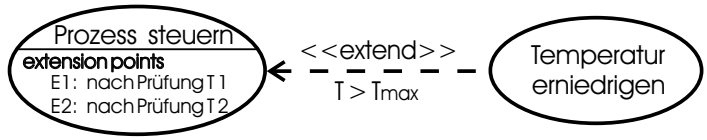

Abbildung 5.2-7: Erweiterung eines Nutzungsfalls durch eine Fehlerbehandlung

Die Erweiterungsbeziehung kann benutzt werden, um alternative Sequenzen von Aktionen oder Sequenzen zur Behandlung von Ausnahmesituationen in einen Nutzungsfall einzufügen. Abbildung 5.2-7 zeigt ein Beispiel.

Ein Nutzungsfall-Diagramm vermittelt lediglich einen Überblick über die Struktur der Nutzungsfälle und die beteiligten Aktoren. Es muss durch detaillierte Spezifikationen der Nutzungsfälle und der zugeordneten Szenarios ergänzt werden. Das Diagramm bietet allerdings ein Hilfsmittel zum schnellen Auffinden einer bestimmten Spezifikation. In diesem Zusammenhang wird erwartet, dass ein Werkzeug die interaktive Verknüpfung unterstützt, z. B. dadurch, dass nach dem Anklicken eines Use-Case-Symbols im Use-Case-Diagramm die zugehörige Spezifikation angezeigt wird.

Eine umfassende Syntax und präzise Semantik zur detaillierten Spezifikation eines Nutzungsfalls bzw. eines Szenarios sind *nicht* als Bestandteil der UML definiert. Es wird der Einsatz von Tabellen mit einer definierten Struktur empfohlen. Tabellen, die sich im praktischen Einsatz bewährt haben, werden in Abschnitt 5.2.4 vorgestellt. Insgesamt resultiert damit ein Dokument mit einer klaren Gliederung und Struktur. Die Spezifikationen der übergeordneten Nutzungsfälle vermitteln einen Überblick über das allgemeine Verhalten der Nutzungsfälle, während die detaillierten Interaktionen durch die zugeordneten Szenarios beschrieben werden. In der Regel haben die Spezifikationen der Nutzungsfälle einen hohen Grad der Stabilität, da die meisten Anpassungen und Änderungen eines Nutzungsfalls die Szenarios betreffen.

5.2.4 Aktivitäten

Die Analyse des Systemverhaltens umfasst folgende Aktivitäten:

- Abgrenzung des SW-Systems gegen die Systemumgebung
- Identifikation und Strukturierung der Nutzungsfälle
- Konstruktion der Use-Case-Diagramme
- Spezifikation und Analyse der Nutzungsfälle
- Spezifikation und Analyse der Szenarios.

Die Vorgehensweise der Analyse ist *iterativ* und *rekursiv*, d. h. die relevanten Aktivitäten werden zunächst nur grob in der angeführten Reihenfolge ausgeführt und danach so lange wiederholt, bis der nötige Detaillierungsgrad erreicht ist, der klare Aussagen zur Erfüllung und Erfüllbarkeit der operationellen Anforderungen zulässt.

Die genannten Aktivitäten werden in den nachfolgenden Abschnitten näher erläutert.

Abgrenzung des SW-Systems gegen die Systemumgebung

Ein System wird in der Regel nicht mit allen Bestandteilen neu entwickelt, sondern es müssen periphere Geräte bzw. externe DV-Systeme angekoppelt und vorhandene Teilsysteme eingebunden werden. Typische Beispiele sind:

- Standardgeräte (z. B. Drucker, Massenspeicher) bzw. spezielle Geräte mit spezifischer Anbindung (z. B. Sensoren, Aktuatoren, Converter);
- tragbare Computer bzw. externe DV-Systeme, die aktuelle Daten liefern bzw. mit aktuellen Daten versorgt werden müssen;
- Teilsysteme, die erprobte Algorithmen bereitstellen (Rechenkerne);
- Datenverwaltungssysteme.

Erster Schritt der Analyse des Systemverhaltens ist die eindeutige Abgrenzung des SW-Systems gegen die Systemumgebung. Dabei ist die Frage zu klären: Welche Bestandteile sind *interne Komponenten* bzw. *Subsysteme* des zu entwickelnden Systems, welches sind dagegen *Aktoren*, die mit dem System kommunizieren bzw. interagieren? Manchmal ist die eindeutige Klärung schwierig. Grundsätzlich sollten DV-Systeme, die bereits fertig entwickelt sind und für deren Anbindung lediglich die Entwicklung bzw. Anpassung einer Schnittstelle erforderlich ist, als Aktoren aufgefasst werden.

Die wesentliche Grundlage für die *Identifikation* und *Spezifikation* der Aktoren bilden die Ausgangsdokumente, die in Abschnitt 5.2.1 aufgelistet sind, vor allem das Begriffslexikon mit der präzisen Definition der typischen Begriffe des Anwendungsbereichs. In der Regel können daraus relevante Schlüsselbegriffe extrahiert werden, die Aktoren bezeichnen. Weitere Aktoren sind durch die Diskussion mit den System- und Anwendungsexperten zu ermitteln, wenn folgende Fragen geklärt werden:

- Wer benutzt direkt das System? Welche Rolle haben die Benutzer?
  Beispiel: Kundenberater, Verkäufer, Sachbearbeiter, Prozessüberwachung.
- Welche Daten werden vom System benötigt? Wer oder was liefert die Daten?
- Wer bzw. was muss vom System informiert werden?
- Welche Geräte bzw. externen DV-Systeme sollen an das System angekoppelt werden?
- Wer ist für die Überwachung und die Wartung des Systems verantwortlich?

Informationen, die für einen Aktor charakteristisch sind, werden zweckmäßig in einer Tabelle erfasst. Dazu gehören:

- die sog. *Sachmittel* (subjects), i. e. Informationen bzw. Dinge, die der Aktor übernimmt oder verwendet bzw. übergibt oder erzeugt, z. B. eine Bestellung, ein Scheck, der Barcode eines Artikels, eine Rechnung, ein Bild etc.;
- die Art der *Zuordnung* des Aktors, d. h. der Standort der Ein-/Ausgabegeräte für die Benutzer, der Standort der speziellen peripheren Geräte und der externen DV-Systeme und die Art der Verbindung.

Beispiel „Auftragsabwicklung in einer Bank“: Es wurden folgende Aktoren identifiziert: Auftragsannahme, Kasse, Automat zum Geldabheben (Bankomat) und Kontoauszugsdrucker – vgl. Abbildung 5.2-4 in Abschnitt 5.2.3.

Identifikation und Strukturierung der Nutzungsfälle

Umfangreiche Systeme sind durch eine große Zahl von Nutzungsfällen mit zahlreichen Varianten gekennzeichnet. Die Reduktion der Komplexität wird erreicht, wenn zunächst diverse Kategorien von Nutzungsfällen unterschieden werden, und zwar dadurch, dass charakteristische Aufgabenbereiche mit typischen Vorgängen definiert werden, z. B.:

- *Processing:*
  Transaktionen, Prozesssteuerung, Datenverarbeitung;
- *Datenverwaltung:*
  Datenerfassung, Plausibilitätsprüfungen, periodischer bzw. bedarfsgesteuerter Datentransfer, Aktualisierung des Datenbestandes;
- *Utilities:*
  Systemstart, Zugriffskontrolle, Statistik und Berichtswesen, Systemwartung etc.

Korrelierte Vorgänge werden dann in einem übergeordneten Nutzungsfall zusammengefasst, komplexe Vorgänge schrittweise in nachgeordnete Nutzungsfälle gegliedert. Die resultierende hierarchische Struktur wird entsprechend dem zunehmenden Systemverständnis verfeinert.

Eine andere Möglichkeit besteht darin, dass man von der typischen hierarchischen Struktur eines SW-Systems ausgeht, die in der linken Spalte von Tabelle 5.2-1 angedeutet ist.

Zunächst wird die übergeordnete Struktur des SW-Systems ermittelt, indem folgende Fragen geklärt werden: Welche Subsysteme lassen sich unterscheiden, welche Prozesse sind einem Subsystem zugeordnet, welche funktionalen Abläufe sind für einen Prozess charakteristisch? Der System-Struktur wird gemäß Tabelle 5.2-1 eine korrespondierende Nutzungsfall-Struktur gegenübergestellt.

Die wesentliche Grundlage für die *Identifikation* der Nutzungsfälle bilden die identifizierten Aktoren und die zugeordneten Sachmittel, ferner die Ausgangsdokumente, die in Abschnitt 5.2.1 aufgelistet sind, vor allem die Problembeschreibung und das Pflichtenheft mit den Benutzerforderungen. *Elementare* Nutzungsfälle sind insbesondere durch die Diskussion mit den System- und Anwendungsexperten zu ermitteln, wenn folgende Fragen geklärt werden:

Tabelle 5.2-1: Hierarchische Struktur aus Nutzungsfällen (repräsentatives Beispiel)

System-Struktur	Nutzungsfall-Struktur	Systemkontext
Subsysteme	Top-level-Nutzungsfälle	Software-System
→ Prozesse	→ nachgeordnete Nutzungsfälle	Subsystem
→ funktionale Abläufe	→ elementare Nutzungsfälle	Prozess

- Welche Arbeitsabläufe sind mit einem identifizierten Aktor verknüpft?
- Welcher Vorgang wird zur Bearbeitung bzw. Erzeugung eines identifizierten Sachmittels benötigt oder durch ein bestimmtes Ereignis ausgelöst?
- Wie interagiert das SW-System mit einem peripheren Gerät bzw. mit einem externen DV-System?

Beispiel „Auftragsabwicklung in einer Bank“: Es wurden folgende Nutzungsfälle identifiziert: Prüfungen beim Geldabheben (am Bankomat), Drucken des Kontoauszugs veranlassen, Kundenauftrag erfassen, Kundenauftrag durchführen, Überweisung veranlassen.

Konstruktion der Use-Case-Diagramme

Die Abhängigkeiten der Nutzungsfälle auf einer bestimmten Ebene der Struktur werden durch Use-Case-Diagramme verdeutlicht – vgl. Abschnitt 5.2.3. Ein bestimmtes Diagramm zeigt

- den Kontext (System, Subsystem, Prozess) des Diagramms;
- die eingeschlossenen Nutzungsfälle und die zugeordneten Aktoren;
- die Verknüpfung der Nutzungsfälle mit den Aktoren;
- die Abhängigkeiten von Nutzungsfällen im Sinne von *include*- und *extend*-Beziehungen.

Beispiel „Auftragsabwicklung in einer Bank“: siehe Abbildung 5.2-4 in Abschnitt 5.2.3.

Spezifikation und Analyse der Nutzungsfälle

Ein Nutzungsfall auf den oberen Ebenen der hierarchischen Struktur repräsentiert nicht direkt ein Szenario, sondern ist primär eine strukturelle Komponente. Es wurde aber bereits darauf hingewiesen, dass ein derartiger Nutzungsfall ein charakteristisches Verhalten hat, das zweckmäßig in einer Spezifikation erfasst wird.

Im praktischen Einsatz hat sich für die Spezifikation eines Nutzungsfalls ein formalisierter Rahmen bewährt, der entsprechend Tabelle 5.2-2 gegliedert ist.

Die Spezifikation eines Nutzungsfalls gilt im Allgemeinen auch für die *nachgeordneten* Nutzungsfälle, sodass in deren Spezifikationen lediglich das spezifische Verhalten ergänzt werden sollte.

Tabelle 5.2-3 zeigt ein Beispiel für die Spezifikation eines Nutzungsfalls entsprechend der Gliederung gemäß Tabelle 5.2-2.

Spezifikation und Analyse der Szenarios

Die Struktur der Nutzungsfälle, die Use-Case-Diagramme und die Spezifikationen der Nutzungsfälle vermitteln das extern sichtbare Verhalten des SW-Systems in einer übersichtlichen Darstellung. Im weiteren Verlauf der Analyse des Systemverhaltens wird jeder *elementare* Nutzungsfall im Detail betrachtet und in Form von Vorgangs-

Tabelle 5.2-2: Spezifikation eines Nutzungsfalls: Gliederung

Kennung	**Nutzungsfall:** Titel (Kurzbezeichnung)
Autor, Datum, Version	
Zweck (kurze, übersichtliche Beschreibung)	
Kontext: NameSystem | NameSubsystem | NameProzess	
Namen der beteiligten Aktoren	
Auslösende Ereignisse	
Bedingungen, die zur Auslösung erfüllt sein müssen (Vorbedingungen)	
Wesentliche Eingangs- und Ausgangsdaten und -signale (Verweis auf die Schnittstellen-Spezifikation)	
Mögliche Ausnahmesituationen und Fehlerfälle	
Wesentliche Ergebnisse des Nutzungsfalls	
Nachbedingungen	
Verweise auf nachgeordnete Nutzungsfälle bzw. auf die zugeordneten Szenarios	
Option: Relevante funktionale Anforderungen	

Tabelle 5.2-3: Spezifikation eines Nutzungsfalls (Beispiel)

B1	**Nutzungsfall:** Prüfungen beim Geldabheben am Bankomat.
Zweck: Das zentrale DV-System führt die notwendigen Prüfungen durch, wenn ein Kunde am Bankomat Geld abhebt.	
Kontext: Auftragsbearbeitung	
Beteiligte Aktoren: direkt: Bankomat; indirekt: Kunde	
Auslösendes Ereignis: Der Kunde schiebt die Kontokarte ein.	
Vorbedingung: Der Bankomat ist in Betrieb. Der zentrale Rechner ist aktiv.	
Relevante Eingangsdaten: Kundenkennung, Geheimnummer, Geldbetrag	
Ausnahmen: Kontokarte ist ungültig, Geheimnummer ist falsch, max. Kredit wird überzogen, Betrag nicht auszahlbar, Bankomat oder zentraler Rechner antwortet nicht.	
Wesentliches Ergebnis: Der gewünschte Geldbetrag ist ausgezahlt und abgebucht.	
Nachbedingung: keine	
Verweis auf Szenario: primäres Szenario: B1-P1	
Relevante Anforderungen: Signale und Daten vom Bankomat erfassen, Prüfungen durchführen, Signale zum Bankomat erzeugen.	

beschreibungen (Szenarios) spezifiziert. Dabei werden zunächst die *primären* Szenarios entwickelt, d. h. das reguläre Verhalten beschrieben. Nach und nach werden die *sekundären* Szenarios ergänzt, in welchen die Reaktionen auf die identifizierten Fehlerfälle und Ausnahmesituationen dargestellt werden.

Ein Szenario liefert die genaue Beschreibung der Aktionen und Reaktionen der Teilvorgänge, die in ihrer Gesamtheit den elementaren Nutzungsfall bilden, in logischer oder zeitlicher Ordnung. Typisch für die erste Aktion eines Teilvorgangs ist eine Bedingung, deren Erfüllung als ein Ereignis interpretiert werden kann, das den Teilvorgang auslöst.

Für die Spezifikation eines Szenrios bietet sich demnach eine *Sequenz* aus einzelnen Schritten mit folgender Struktur an:

fortlaufendeNummer [Bedingung:] Aktion

Darin bedeuten:

- fortlaufendeNummer: die Kennzeichnung der Aktion durch eine fortlaufende Nummer;
- [Bedingung:] optionale Bedingung, die für die Aktion erfüllt sein muss;
- Aktion: die Beschreibung der Aktion.

Für die Beschreibung der Aktionen müssen eindeutige und widerspruchsfreie Aussagen angestrebt werden. Das Ziel wird erreicht, wenn zur Beschreibung einer einzelnen Aktion ein einfacher Satz gebildet wird, und zwar entsprechend der folgenden Charakteristik:

- Auslöser der Aktion (Aktor bzw. Vorgang im SW-System): Subjekt im Satz;
- ausgeführte Aktion in *aktiver* Formulierung: Prädikat im Satz;
- Ziel der Aktion: Objekt im Satz in der *Einzahl*, es sei denn, die Aktion bezieht sich tatsächlich auf mehrere Objekte;
- Qualifizierung des Objekts: Adjektiv oder einfacher Relativsatz beim Objekt.

Die Eindeutigkeit der Aussagen wird erhöht, wenn bei der Beschreibung der einzelnen Aktionen

- eindeutig definierte Begriffe aus dem Begriffslexikon verwendet werden;
- lösungsbezogene Begriffe (wie z. B. Datei), Synonyme zur sprachlichen Abwechslung, Substantive zur Beschreibung von Aktionen und erläuternde Nebensätze vermieden werden.

Dagegen wird die Übersichtlichkeit verbessert, wenn

- die Aktionen, die zu einem bestimmten Teilvorgang gehören, durch eine Einrückung kenntlich gemacht werden;
- eine längere Sequenz von Aktionen, die unter einer bestimmten Bedingung ausgeführt wird, zu einem eigenen Teilszenario zusammengefasst und unter Anwendung der *include*- bzw. *extend*-Beziehung einbezogen wird.

Wenn die Regeln beachtet werden, wird mit großer Wahrscheinlichkeit erreicht, dass die Semantik der Spezifikation eines Szenarios eindeutig ist und von einem Leser

Tabelle 5.2-4: Spezifikation eines Szenarios: Gliederung

Kennung	**[Primäres | Sekundäres] Szenario:** Titel und Kennung des zugehörigen Nutzungsfalls
[Bedingung] -- Bedingung für die Einfügung des nachfolgenden Szenarios	
{fortlaufendeNummer [Bedingung:] Aktion -- Spezifikation einer einzelnen Aktion	
[fortlaufendeNummer einfügen: KNf[: Ai – Aj]]} -- Einfügung des Szenarios eines anderen Nutzungsfalls	

ohne spezielle methodische Kenntnisse unmittelbar verstanden und korrekt interpretiert wird.

Im praktischen Einsatz hat sich für die Spezifikation eines Szenarios ein formalisierter Rahmen bewährt, der entsprechend Tabelle 5.2-4 gegliedert ist.

Handelt es sich bei dem spezifizierten Szenario um eine mögliche Erweiterung des Szenarios eines Nutzungsfalls, zu dem eine *extend*-Beziehung definiert ist, wird zunächst die Bedingung angeführt, die für die Einfügung des Szenarios erfüllt sein muss. Daran anschließend folgen die einzelnen Aktionen des spezifizierten Szenarios. An Stelle einer Aktion kann auch ein anderes Szenario oder ein Ausschnitt aus einem anderen Szenario eingefügt werden. Voraussetzung ist, dass zum korrespondierenden Nutzungsfall eine include-Beziehung bzw. von diesem eine extend-Beziehung spezifiziert ist.

Es bedeuten:

- KNf: Kennung eines elementaren Nutzungsfalls (Szenarios);
- Ai bzw. Aj: Nummer der ersten bzw. letzten Aktion der Sequenz von Aktionen, die eingefügt werden sollen.

Durch die Anwendung der *include*-Beziehung wird die wiederholte Spezifikation von gemeinsamen Sequenzen von Aktionen vermieden. Änderungen in einem Szenario wirken sich dann unmittelbar auf alle nutzenden Szenarios aus. Die ständigen Korrekturen und Erweiterungen, die für die Spezifikation von Szenarios typisch sind, vereinfachen sich dadurch beträchtlich.

Tabelle 5.2-5 zeigt ein einfaches Beispiel für die Spezifikation eines Szenarios.

In Tabelle 5.2-5 ist der elementare Vorgang im SW-System, der in einer Aktion als Subjekt oder Objekt agiert bzw. reagiert, mit „S.“ abgekürzt. Aktionen, die zu einem nachgeordneten Teilvorgang gehören, sind durch eine Einrückung kenntlich gemacht. Die eingefügten Kommentare machen jeweils deutlich, welche Funktionen im Bankomat ablaufen, sodass dafür keine Unterstützung durch das zentrale SW-System erforderlich ist.

Obwohl die einzelnen Schritte eines Teilvorgangs nacheinander ablaufen, repräsentiert ein Szenario keine Beschreibung eines Ablaufs im Sinne eines prozeduralen Programms.

Tabelle 5.2-5: Spezifikation des primären Szenarios zum Nutzungsfall gemäß Tabelle 5.2-3.

B1	**Primäres Szenario:** Prüfungen beim Geldabheben am Bankomat
Nr.	**Aktion**
1	Kontokarte ist eingezogen: Bankomat übermittelt die Kennung des Kunden und die Kontonummer an das relevante Objekt im SW-System (S.).
	Kommentar: Der Bankomat liest die verschlüsselten Daten auf der Kontokarte.
2	S. überprüft die übermittelten Daten.
3	S. verlangt die Eingabe der Geheimnummer.
	Kommentar: Die Aufforderung wird am Bankomat-Display angezeigt.
4	Die Geheimnummer ist eingegeben: Bankomat übermittelt die Geheimnummer an S.
5	S. überprüft die Geheimnummer.
6	S. fordert den gewünschten Geldbetrag an.
	Kommentar: Die Aufforderung wird am Bankomat-Display angezeigt.
7	Der Geldbetrag ist eingegeben: Bankomat übermittelt den Betrag an S.
8	S. überprüft den Betrag gegen den Kontostand und den maximalen Kredit.
9	S. übermittelt „ok“ zur Auszahlung des Betrags an den Bankomat.
	Kommentar: Der Bankomat überprüft den Betrag gegen den Geldscheinvorrat, ermittelt die Geldscheine entsprechend dem Betrag, gibt diese aus und druckt einen Auszahlungsbeleg mit Betrag, Datum und Uhrzeit.
10	Bankomat übermittelt „ok“ zur Abbuchung des Betrags.
11	S. bucht den Betrag vom Girokonto ab.
12	S. vermerkt die Abbuchung mit Betrag, Datum und Uhrzeit als Vorgang.
13	S. veranlasst den Bankomat, die Kontokarte auszugeben.

Teilvorgänge, deren einleitende Bedingung zu einem bestimmten Zeitpunkt erfüllt ist, können im Prinzip zeitgleich zueinander ablaufen.

Die Umsetzung eines Szenarios in eine grafische Repräsentation und die im Verlauf der Analyse ständig anfallenden Änderungen und Anpassungen der Grafik rechtfertigen in der Regel nicht den dafür notwendigen Aufwand. Im Allgemeinen ist das Ergebnis kaum übersichtlicher als die tabellarisch-textuelle Darstellung mit einfachen Sätzen. Häufig wird empfohlen, die Interaktionen der Aktoren mit einem Nutzungsfall in Form eines Sequenzdiagramms darzustellen. Dabei muss man allerdings beachten, dass

ein Sequenzdiagramm das Ergebnis der Konzipierung eines dynamischen Ablaufmodells ist, das eine *objektorientierte Umsetzung* eines Szenarios darstellt. Die Umsetzung ist aber erst dann sinnvoll, wenn auf der Grundlage der Szenarios die diversen Klassen identifiziert, diese zueinander in Beziehung gesetzt und die Interaktion der Objekte abgeleitet ist.

Bei betriebswirtschaftlichen Anwendungen stehen im Allgemeinen der interaktive Dialog der Benutzer mit dem System und die Vorgänge im Sinne von Transaktionen im Vordergrund der Betrachtung. Dabei bietet die Spezifikation der erforderlichen Eingabedaten und der dargestellten Ergebnisdaten ein wichtiges zusätzliches Hilfsmittel für die Analyse der operationellen Anforderungen. Es empfiehlt sich deshalb, einen (übergeordneten) Nutzungsfall durch eine präzise Spezifikation der relevanten Daten zu ergänzen.

Die Spezifikation der Daten bildet zusammen mit den Szenarios die Grundlage für die Gestaltung der Dialog- und Repräsentationselemente (Views) der Benutzeroberfläche. Erste Vorschläge zur Gestaltung der Views tragen auch zur Klärung grundsätzlicher Fragen bei. Dabei geht es unter anderem um die notwendige Unterstützung der spontanen Umschaltung der Views, die jeweils einem bestimmten Nutzungsfall zugeordnet sind, sowie um erforderliche Undo- und Redo-Funktionen. Obwohl View-Elemente die Analyse der operationellen Anforderungen unterstützen, sind diese *nicht* Bestandteil der Spezifikation eines Nutzungsfalls.

In technischen Systemen treten die Wechselwirkungen mit speziellen Endgeräten bzw. mit eingebetteten Systemen oder kontrollierten Prozessen in den Vordergrund der Betrachtung. Die Szenarios beziehen sich dabei hauptsächlich auf die Wechselwirkungen, die insbesondere durch die Sequenz der Daten- und Signalflüsse von/zu den peripheren Geräten bestimmt sind. Die Beschreibung der Systeminteraktion muss deshalb durch die präzise Spezifikation der Eingangs- und Ausgangsdaten/-signale in Gestalt von Schnittstellenbeschreibungen (Interface-Dokumenten) ergänzt werden.

5.2.5 Resultierende Vorzüge

Verbindliche Grundlage für die Entwicklung

Die Ergebnisse der Use-Case-Analyse sowie der Aktivitäten- und Reaktionsanalyse, insbesondere die Spezifikationen der Nutzungsfälle und Szenarios, bilden die wesentliche Grundlage für die weitere Entwicklung des SW-Systems. Die resultierenden Dokumente schaffen nämlich die Voraussetzung für

- die *Abstimmung* des geforderten Systemverhaltens zwischen Systemnutzern bzw. System- und Anwendungsexperten und den Entwicklern;
- die *weiterführende Strukturierung der Anwendung*, wodurch die parallele Entwicklung und diesbezügliche Arbeitsteilung gefördert wird;
- die *Gestaltung der Benutzeroberfläche*, d. h. der Views und der grafischen Elemente zur Unterstützung des Benutzerdialogs;

- die *Konzipierung der Schnittstellen* zu den peripheren Geräten und externen DV-Systemen;
- die *Identifikation der Klassen für relevante Objekte* und deren Beziehungen;
- die *Spezifikation* der *dynamischen Objekt-Interaktion* und des *Objektverhaltens;*
- die *Validation* der Analysemodelle;
- die *Ableitung der Testpläne und der Testprozeduren* als Basis für den Ablauftest (Test der Szenarios) und den Test des integrierten Systems (End-to-end-Test);
- die *Erstellung der Benutzerhandbücher;*
- die *Abschätzung des Aufwands* für die weitere Systementwicklung.

Durch die Zustimmung des Auftraggebers als Vertreter der Systemnutzer und der Anwendungsexperten und durch die Gegenzeichnung wird das Dokument zum *Vertrag*, an den die Entwickler gebunden sind.

Änderungen und Erweiterungen der operationellen Anforderungen

Im Verlauf der zyklischen Entwicklung des Sofware-Systems und auch später beim operationellen Einsatz ergeben sich Änderungen und Erweiterungen der operationellen Anforderungen. Hauptsächliche Gründe dafür sind:

- die Beseitigung von Fehlern;
- die Erweiterung der Funktionalität;
- die Verbesserung des Benutzerdialogs und/oder der Datenverwaltung;
- der Anschluss eines neuen Geräts bzw. die Verbindung mit einem DV-System;
- die Verbesserung der Reaktionen bei Fehlerfällen und der Ausnahmebehandlung.

Das Modell der Systemnutzung und das Work-flow-Modell bilden für die Analyse der neuen und geänderten Anforderungen die wesentliche Grundlage. Folgende Aktivitäten sind relevant:

- Erweiterung bzw. Modifikation der Struktur der Nutzungsfälle;
- Spezifikation der neuen Nutzungsfälle und Szenarios bzw. Modifikation der betroffenen Spezifikationen;
- Erweiterung bzw. Modifikation des Work-flow-Modells.

Die Änderungen haben Auswirkungen auf die logischen und physikalischen Modelle, aus denen auf die notwendigen Änderungen in der Realisierung des SW-Systems geschlossen werden kann.

5.3 Entwurf der Systemarchitektur

5.3.1 Einführung

Eine wesentliche Voraussetzung für den Erfolg eines Projekts zur Entwicklung eines SW-Systems ist die Unterteilung des Gesamtsystems in kleinere und letzten Endes überschaubare Einheiten: Subsysteme, Prozesse und SW-Komponenten mit Klassen

bzw. Objekten, welche die benötigte Funktionalität bieten. Das gestalterische Konzept, das der Unterteilung bzw. der resultierenden Aufbau- und Ablaufstruktur zugrunde liegt, wird als *Architektur des SW-Systems* bezeichnet.

Die Systemarchitektur bildet das Fundament, auf dem das SW-System entwickelt wird. Sie muss so konzipiert sein, dass eine flexible Erweiterung und Anpassung der diversen Systemprototypen möglich ist, die im iterativen und evolutionären Entwicklungsprozess entstehen. Eine gute und in hohem Maße stabile Systemarchitektur ist für die Entwicklung des SW-Systems eine wesentliche Voraussetzung. Eine Entwicklung, die nicht auf der Grundlage einer soliden Systemarchitektur erfolgt, ist immer mit zusätzlichen Kosten und Terminüberschreitungen verbunden.

Der Entwurf der Systemarchitektur schließt sich nahtlos an die *Analyse des Systemverhaltens* an. Ziel des Entwurfs ist eine flexible System- und Modellstruktur, in welche die logischen Analyse- und Entwurfsmodelle und die physikalischen Modelle eingebunden werden.

Die Systemarchitektur hat folgende wesentliche Bestandteile:

- *logische* Systemstruktur, bestehend aus der
  - statischen Aufbaustruktur
  - dynamischen Ablaufstruktur
  - Verhaltensstruktur (prototypischen Aufbau- und Ablaufstruktur).
- *physikalische* Systemstruktur, bestehend aus der
  - modularen SW-Struktur: Struktur aus SW-Komponenten;
  - konfigurierten Struktur: Struktur aus Hardware-Komponenten mit residenten SW-Komponenten.

Die logische und physikalische Systemstruktur und ihre Bestandteile repräsentieren komplementäre Sichten (Views) des SW-Systems. Das Modell der Systemnutzung, das als Ergebnis der Use-Case-Analyse entsteht, repräsentiert eine Struktur aus Spezifikationselementen, aus der die Verhaltensstruktur bzw. die Aufbau- und Ablaufstruktur ableitbar sind.

Ausgangsdokumente

Die Grundlagen für den Entwurf der Systemarchitektur sind:

- die Abgrenzung des SW-Systems gegen die Systemumgebung und die Abgrenzung der Problembereiche des SW-Systems;
- das Modell der Systemnutzung, d. h. die Ergebnisse der Use-Case-Analyse, insbesondere die Struktur der Nutzungsfälle;
- die Struktur des Zielrechnersystems, soweit diese bereits vorgegeben ist.

Ausführende

Dem Projektleiter sollte ein Entwurfsexperte als *Systemarchitekt* zur Seite stehen, der für die Konstruktion der Systemarchitektur und für die Erhaltung ihrer Konsistenz und

Integrität verantwortlich ist. Er sollte durch erfahrene Systementwickler unterstützt werden, die gute Erfahrungen im Entwurf von SW-Systemen haben. Wesentlich ist, dass die Ausführenden mit dem objektorientierten Entwurf vertraut sind, der auf der UML basiert und durch rechnergestützte Werkzeuge gut unterstützt wird.

Relevante Ergebnisse

Die wesentlichen Ergebnisse des Entwurfs der Systemarchitektur sind:

- die logische Systemstruktur;
- die korrespondierende physikalische Systemstruktur.

5.3.2 Relevante Modellelemente: Paket, Subsystem, Modell

Paket

Ein *Paket* ist ein zusammengesetztes Modellelement mit einer definierten inneren Struktur und mit eindeutiger Abgrenzung zu anderen Elementen. Es bietet einen Mechanismus zur Gruppierung und Verknüpfung von Modellelementen, die in ihrer Gesamtheit ein Analyse- bzw. Entwurfsmodell bzw. einen Ausschnitt aus einem Modell repräsentieren.

Die relevanten Modellelemente sind:

- *Klassifizierer*, nämlich die diversen Arten von Klassen, Use-Case, Kollaboration;
- Abhängigkeits-, Vererbungs- und Nutzungsbeziehungen zwischen Klassifizierern;
- Objekte sowie die Links zwischen Objekten und die Botschaften zur Verdeutlichung der Kommunikation und Interaktion der Objekte;
- SW-Komponenten, die einen Bestandteil der physikalischen Systemstruktur repräsentieren;
- andere Pakete und Abhängigkeitsbeziehungen zwischen diesen.

Pakete bilden den organisatorischen Rahmen für die Entwicklung und Verwaltung der Modelle, die in ihrer Gesamtheit die Spezifikationsstruktur sowie die logische und physikalische Systemstruktur repräsentieren.

Das durch die UML vordefinierte grafische Symbol eines Pakets ist ein großes Rechteck mit durchgezogenem Rand, an das links oben ein kleineres, schmales Rechteck aufgesetzt ist. Das Symbol repräsentiert eine Karteikarte mit einem aufgesetzten Schild zur Aufnahme einer Identifikation. Abbildung 5.3-1 zeigt ein Beispiel.

Der Name des Pakets wird in das aufgesetzte Rechteck eingetragen. Im großen Rechteck (Paketkörper) werden die Bestandteile des Pakets durch ihre Symbole dargestellt. Dazu gehören auch die Verbindungen zwischen den Symbolen. Ein Paket kann bei Verwendung in einem anderen Modell oder durch Zooming auf seinen Namen reduziert und dabei deutlich verkleinert werden. Der Paketname wird dann innerhalb des großen Rechtecks dargestellt – siehe Abbildung 5.3-1.

Grundlage eines Pakets ist die *besitzt*-Beziehung. Dies bedeutet, dass ein Paket folgende Eigenschaften hat:

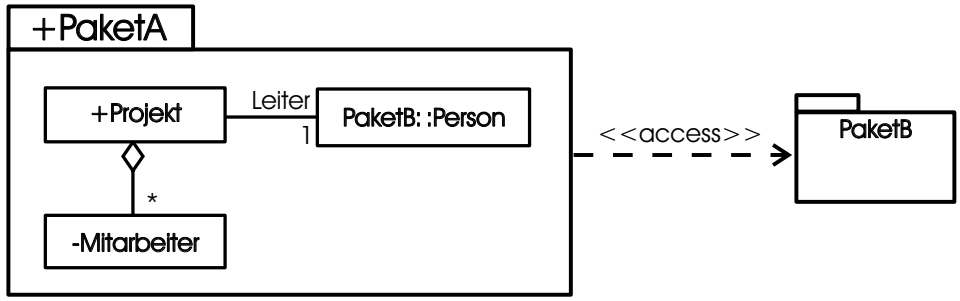

Abbildung 5.3-1: Paket mit Klassen und ihren Beziehungen als Bestandteile

- Ein Paket definiert einen Namensraum für die eingebundenen Elemente, d. h. die Elemente müssen durch ihren Namen eindeutig identifizierbar sein. Ein bestimmtes Element in einem Paket kann durch einen Pfadausdruck nach folgendem Muster adressiert werden:

  NamePaket{::NameInneresPaket}::NameElement

- Ein bestimmtes Element darf nur *einem* Paket als dessen fester Bestandteil zugeordnet werden. Die mögliche Sichtbarkeit eines Elements innerhalb und außerhalb des Pakets wird durch ein Sichtbarkeitssymbol festgelegt, das vor den Namen des Elements gesetzt wird. Alternativen sind:

  „+“: Das Element ist öffentlich, d. h. es ist außerhalb des Pakets sichtbar.
  „-“: Das Element ist privat, d. h. es ist nur für seine Bestandteile sichtbar.
  „#“: Das Element ist privat, ist aber auch für die nachgeordneten Pakete in einer Vererbungsbeziehung sichtbar (siehe unten).
  „~“: Das Element ist für alle anderen Elemente im Paket und transitiv für die Elemente in eingebundenen Paketen sichtbar (Paket-Sichtbarkeit).

  Abbildung 5.3.1 zeigt ein Paket *PaketA* mit einem Minimodell, das zwei dem Paket zugeordnete Klassen und eine Klasse aus einem anderen Paket mit dem Namen *PaketB* umfasst. Das Paket *PaketA* und die dem Paket zugeordneten Klassen haben Sichtbarkeitssymbole.
  Es wird erwartet, dass unter der Kontrolle eines Werkzeugs die Darstellung des Paketinhalts auf die öffentlichen Bestandteile beschränkt oder selektiv eingestellt werden kann.

- Ein Paket *P* kann andere Pakete *Pi* als Elemente enthalten. Auf diese Weise entstehen hierarchische Strukturen aus Paketen. Dabei sind die öffentlichen Elemente in einem *inneren* Paket für das Paket *P* direkt sichtbar. Dagegen sind für das Paket *P* die öffentlichen Elemente eines *externen* Pakets *Pe* nur dann tatsächlich nutzbar, wenn vom Paket *P* zum Paket *Pe* eine spezialisierte Nutzungsbeziehung definiert ist.

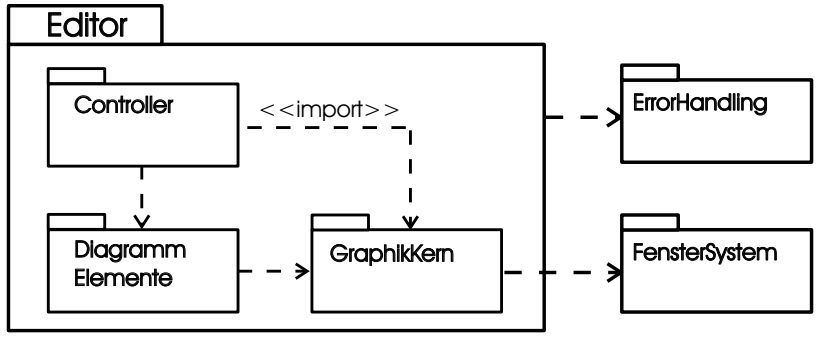

Abbildung 5.3-2: Importbeziehung zwischen Paketen

Zwischen zwei Paketen sind folgende Beziehungen definiert:

- *Nutzungsbeziehungen*
  Die Nutzungsbeziehung bildet die Voraussetzung dafür, dass in einem Modell, das in ein Paket *P* eingebunden ist, eine Beziehung zu einem Modellelement hergestellt werden kann, das einem externen Paket *Pe* zugeordnet ist. Es sind zwei Alternativen definiert:
  - *Zugriffsbeziehung:* Nutzungsbeziehung vom Paket *P* zum Paket *Pe* mit Stereotyp «access»;
    Das Paket *P* kann öffentliche Modellelemente aus dem Paket *Pe* referenzieren, d. h. Kopien der Modellelemente mit dem vollen Pfadnamen können in das Modell des Pakets *P* einbezogen werden. In Abbildung 5.3-1 wird z. B. im Paket *PaketA* die Klasse *Person* aus dem Paket *PaketB* referenziert.
  - *Importbeziehung:* Nutzungsbeziehung vom Paket *P* zum Paket *Pe* mit Stereotyp «import»;
    Die öffentlichen Modellelemente aus dem Paket *Pe* werden zusätzlich dem Namensraum des Pakets *P* zugeordnet, d. h. Kopien der Modellelemente können in das Modell des Pakets P einbezogen werden. Es wird erwartet, dass ein Werkzeug eine spezielle Kennzeichnung in das grafische Symbol eines importierten Modellelements einträgt, z. B. *(fromNameImportiertesPaket)*.
    Abbildung 5.3-2 zeigt diverse Beispiele. Die öffentlichen Elemente des Pakets *FensterSystem* sind ausschließlich im Paket *GrafikKern* nutzbar. Entsprechend ist der öffentliche Inhalt des Pakets *ErrorHandling* im Paket *Editor* nutzbar.

  Eine Nutzungsbeziehung zwischen zwei Paketen, die durch keinen Stereotyp gekennzeichnet ist, wird als Import-Beziehung interpretiert.
- *Vererbungsbeziehung*
  Zwei Pakete können durch eine Vererbungsbeziehung miteinander verknüpft sein. Im abgeleiteten Paket sind die nicht-privaten Modellelemente und Beziehungen des

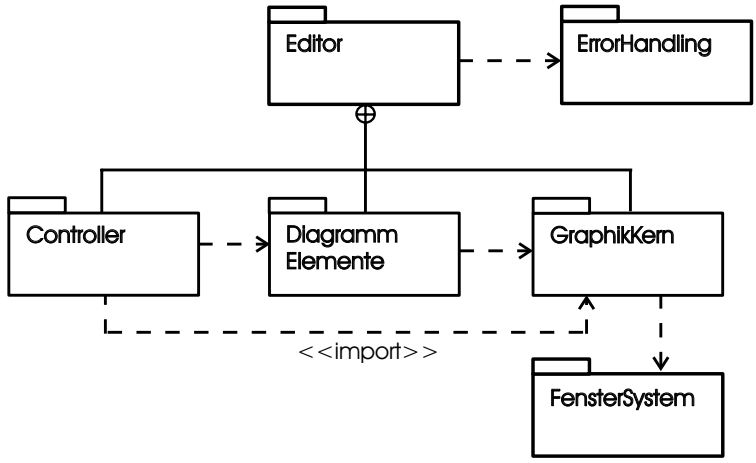

Abbildung 5.3-3: Darstellung einer Paketstruktur in Form einer Baumstruktur

vererbenden Pakets sichtbar. Diese können durch weitere Modellelemente und Beziehungen ergänzt werden.

Ein Paket und seine Bestandteile können auch in Form einer Baumstruktur dargestellt werden. Ein Anker-Symbol (Kreis mit kreuzenden Linien) beim übergeordneten Paket verdeutlicht die Zuordnung der Bestandteile. Abbildung 5.3-3 zeigt ein Beispiel.

Die Bildung von Paketen sollte auf eine hohe innere Kohäsion und die Minimierung der Kopplung zwischen den Paketen ausgerichtet sein. Dies wird erreicht, wenn jeweils stark korrelierte Elemente in einem Paket zusammengefasst werden.

Eine allgemein verwendbare Gruppierung von Modellelementen kann durch einen Stereotyp des Pakets angezeigt werden, der über dem Paketnamen angegeben wird. Die für den Basistyp „Paket“ vordefinierten Stereotypen sind in Tabelle 5.3-1 aufgelistet.

Ein Profil-Paket kann mit einem anderen Paket verknüpft werden. Für diesen Zweck sind zwei Stereotypen vordefiniert. Der Stereotyp «appliedProfile» zeigt an, welches Profil für ein Paket relevant ist, der Stereotyp «modelLibrary» verbindet eine Modellbibliothek mit einem Profil.

Subsystem

Ein *Subsystem* repräsentiert eine Einheit mit einer definierten Verhaltensstruktur in einem Modell und stellt eine Verbindung der Spezifikationsstruktur mit der logischen und physikalischen Systemstruktur her.

Die grafische Repräsentation ist das Paketsymbol, das durch ein Ikon in Form einer kleinen Gabel (fork) markiert wird. Das Ikon wird rechts neben dem Namen in das Schild des Pakets bzw. rechts oberhalb des Namens im Paketfeld eingetragen.

Tabelle 5.3-1: Vordefinierte Stereotypen für die Metaklassen „Paket“ bzw. „Model“ (siehe unten)

NameStereotyp	**Bedeutung:** Das Paket …
profile	repräsentiert das Profil eines Modells, d. h. enthält benutzerdefinierte Stereotypen und spezialisierte Modellelemente – vgl. Abschnitt 4.2.2.
topLevel	ist die Wurzel einer Paketstruktur.
stub	repräsentiert nur öffentliche Bestandteile anderer Pakete.
facade	referenziert nur Elemente in anderen Paketen.
framework	enthält generische Einheiten und repräsentiert eine wieder verwendbare Architektur für ein System oder ein Teilsystem.
modelLibrary	umfasst eine Bibliothek aus Modellelementen.
systemModel	umfasst die Modelle eines SW-Systems.
metamodel	repräsentiert die Abstraktion eines Modells.

In einem Modell wird ein Subsystem-Paket wie ein Paket eingesetzt, d. h. es wird entweder mit seinem Inhalt dargestellt (vgl. Abbildung 5.3-4) oder es ist ohne Inhalt Bestandteil einer Paketstruktur.

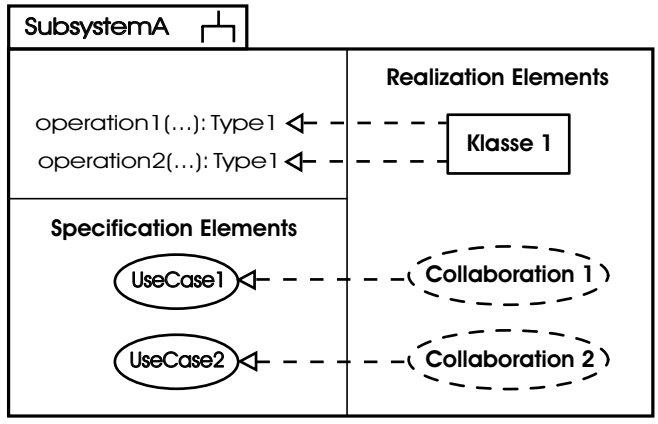

Abbildung 5.3-4: Struktur eines Subsystem-Pakets (abstraktes Beispiel)

Das Paketfeld eines Subsystem-Pakets ist gemäß Abbildung 5.3-4 in drei optionale Abschnitte unterteilt, die folgenden Inhalt haben:

- eine links angeordnete Liste mit Spezifikationen von Operationen, die vom Subsystem ausgeführt werden;

- ein links unterhalb der Operationen angeordneter Abschnitt mit der Überschrift „Specification Elements“ für die Aufnahme von sog. Spezifikationselementen. Typische Spezifikationselemente sind Use-Cases sowie korrespondierende Darstellungen in Form von Zustands- bzw. Ablaufmodellen;
- ein rechter Abschnitt mit der Überschrift „Realization Elements“, in den sog. Realisierungselemente eingetragen werden. Typische Realisierungselemente sind Kollaborationen, Aggregate aus Klassen oder einzelne Klassen.

Die Realisierungselemente repräsentieren (vorgesehene) Realisierungen von Operationen und Spezifikationselementen. Die relevanten Verknüpfungen können innerhalb des Pakets durch *Realisierungsbeziehungen* verdeutlicht werden, wie dies im abstrakten Beispiel gemäß Abbildung 5.3-4 gezeigt ist.

Einem Systempaket können externe Schnittstellen zugeordnet sein, die außerhalb des Pakets (z. B. in Form von Pins) dargestellt und durch Elemente im Paket (z. B. durch Klassen) realsiert werden. Einzelheiten zur Spezifikation und Nutzung von Schnittstellen werden im Kapitel 7 (Entwurf) behandelt.

Die diversen Elemente eines System-Pakets können auch in ungeordneter Form ohne Begrenzungen und ohne Realisierungsbeziehungen oder auch in Form einer Baumstruktur dargestellt werden. Sie sind auch auf mehrere Pakete in unterschiedlicher Darstellungsform aufteilbar. Ein Subsystem-Paket kann demnach auch eine reine Spezifikationseinheit oder eine instantiierbare Realisierungseinheit (mit dem Strereotyp «instantiable») darstellen.

Relevante Beziehungen zwischen Subsystem-Paketen sind die Import- und die Verfeinerungsbeziehung.

Modell

Ein *Modell* repräsentiert eine abstrakte Einheit eines physikalischen Systems, die einen bestimmten Zweck erfüllt. Die grafische Repräsentation ist das Paketsymbol, das durch ein Ikon in Form eines kleinen Dreiecks gekennzeichnet wird. Das Ikon wird rechts neben dem Namen in das Schild des Pakets bzw. rechts oberhalb des Namens im Paketfeld eingetragen.

Vordefinierte Stereotypen für ein Modell sind «systemModel» und «metamodel» – siehe bei Paket. Relevante Beziehungen zwischen Modell-Paketen sind neben der Import- und Verfeinerungsbeziehung die Verknüpfung zwischen unterschiedlichen Modellen (trace-Beziehung) – siehe Tabelle 4.2-4 in Abschnitt 4.2.3.

5.3.3 Prinzipielle Systemstruktur

Umfangreiche Analysemodelle werden in überschaubare Teilmodelle gegliedert. Dabei wird jedes Teilmodell in ein Paket eingebunden oder einem Paket logisch zugeordnet. Die Abhängigkeiten zwischen den Teilmodellen werden als Nutzungsbeziehungen

zwischen den Paketen oder durch die Zuordnung sichtbar. Insgesamt resultiert eine hierarchische Struktur aus Paketen, wobei ein Paket auf einer höheren Ebene der Struktur jeweils eine Struktur aus korrelierten Modellen repräsentiert. Auf den unteren Ebenen der Struktur sind die Analysemodelle angesiedelt, welche aus den jeweils relevanten Modellelementen zusammengesetzt sind.

Statische Systemstruktur und Struktur der Nutzungsfälle

Die Struktur der Nutzungsfälle repräsentiert eine statische Struktur, die auf die übersichtliche Spezifikation der dynamischen Interaktionen zwischen externen Objekten (Aktoren) und elementaren Nutzungsfällen zugeschnitten ist. Zwischen der *statischen Systemstruktur* und der *Struktur der Nutzungsfälle* besteht deshalb ein Zusammenhang, der in Tabelle 5.3-2 verdeutlicht ist.

Tabelle 5.3-2 zeigt eine hierarchische Struktur mit vier Ebenen. Die oberste Ebene repräsentiert das gesamte SW-System, das in Subsyteme unterteilt ist. Jedes Subsystem ist in Prozesse gegliedert, jedem Prozess sind funktionale Abläufe nachgeordnet.

Die statische Systemstruktur ist als Paketstruktur konzipiert, wobei ein Paket auf einer bestimmten Ebene jeweils ein Diagramm einschließt, das eine Struktur aus Paketen mit Importbeziehungen darstellt. Auf jeder Ebene ist der Systemstruktur die korrespondierende Struktur aus Nutzungsfällen gegenübergestellt.

Tabelle 5.3-2: Statische Systemstruktur und korrespondierende Struktur der Nutzungsfälle

Ebene	Statische Systemstruktur	Struktur der Nutzungsfälle (Spezifikationsstruktur)
System	Paket, das das gesamte Software-System repräsentiert	Nutzungsfall, der das gesamte SW-System repräsentiert.
	Inhalt des Pakets: Subsystemstruktur (Subsysteme und Beziehungen)	
Subsysteme	Struktur aus Paketen, die jeweils ein Subsystem repräsentieren	Struktur aus Top-level-Nutzungsfällen
	Inhalt eines Pakets: Prozess-Struktur	
Prozesse	Struktur aus Paketen, die jeweils einen Prozess repräsentieren	Struktur aus nachgeordneten Nutzungsfällen
	Inhalt eines Pakets: Struktur aus aktiven Objekten, denen jeweils funktionale Abläufe zugeordnet sind	
Funktionale Abläufe	Pakete, die jeweils einen funktionalen Ablauf repräsentieren	Struktur aus elementaren Nutzungsfällen

Typische Subsysteme eines SW-Systems sind:

- die Teilsysteme, die spezielle, stark korrelierte Aufgaben durchführen und gegebenenfalls auf verschiedenen Prozessoren bzw. Systemknoten ablaufen;
- die Teilsysteme, die allgemeine Aufgaben durchführen: Login, Zugriffsrechtsregelung, Berichterstellung, zentrale Ausnahmebehandlung, Systemrekonfiguration nach Hardware-Ausfällen etc.;
- die grafische Benutzeroberfläche zur Unterstützung des Dialogs mit den Benutzern;
- das Kommunikationssystem zur Ankoppelung der peripheren Geräte und externen DV-Systeme;
- der Transaktionsmanager und das Datenbanksystem zur Nutzung und Verwaltung der persistenten Objekte.

Den genannten Teilsystemen stehen in der Regel korrespondierende Top-level-Nutzungsfälle gegenüber. Die übrigen Subsysteme haben eine unterstützende Funktion. Die diesbezüglichen Anforderungen werden aus den Ergebnissen der Use-Case-Analyse und/oder der objektorientierten Analyse abgeleitet.

Es wird zunächst angenommen, dass die Subsysteme und die darin eingebundenen Prozesse zeitparallel zueinander agieren und bidirektional miteinander verknüpft sein können.

Strukturierung der Analysemodelle

Im Verlauf der objektorientierten Analyse werden konzeptionelle Modelle gebildet, die auf unterschiedliche Aspekte des zu entwickelnden SW-Systems zugeschnitten sind. Tabelle 5.3-3 vermittelt einen Überblick über die Modelle, die sich grundsätzlich unterscheiden lassen.

Ein *Informationsmodell* zeigt die statischen Beziehungen zwischen Informationseinheiten, die durch Klassen für Entity-Objekte repräsentiert werden.

Das *Prozessmodell* ist ein konkretes Modell, das die dynamische Interaktion von (aktiven) Controller-Objekten verdeutlicht. Dagegen bezeichnet ein *Prozessrahmenmodell* ein abstraktes Modell, das die Klassen der Controller-Objekte und die statischen Be-

Tabelle 5.3-3: Konzeptionelle Modelle der objektorientierten Analyse

Statische Modelle	Dynamische Modelle
Informationsmodell	---
Prozessrahmenmodell	Prozessmodell Zustandsmodell
Ablaufrahmenmodell ---	Ablaufmodell Zustandsmodell

ziehungen zwischen diesen Klassen verdeutlicht. Es repräsentiert demnach eine *Abstraktion* eines korrespondierenden Prozessmodells.

Ein *Ablaufmodell* ist ein konkretes Modell aus Objekten, deren dynamische Interaktion einen funktionalen Ablauf im Sinne eines elementaren Vorgangs verdeutlicht. Dagegen bezeichnet ein *Ablaufrahmenmodell* ein abstraktes Modell, das die Klassen der Objekte und die statischen Beziehungen zwischen diesen Klassen verdeutlicht. Es repräsentiert demnach eine *Abstraktion* eines korrespondierenden Ablaufmodells. Typisch für ein Ablaufrahmenmodell ist, dass darin Klassen für Controller-, Boundary- und Services-Objekte einbezogen sind, die steuernde und koordinierende Aufgaben übernehmen, sowie Klassen für Entity-Objekte, die dem Informationsmodell zugeordnet sind. Einzelheiten zu den genannten Arten von Klassen werden im Kapitel „Objektorientierte Analyse“ behandelt.

Das *Zustandsmodell* ist ein Modell, welches das interne Verhalten der Objekte einer bestimmten Klasse definiert.

Eine Klasse wird als Bestandteil eines Rahmen- bzw. Informationsmodells einem Paket und damit einem Teilmodell fest zugeordnet. Sind Klassen bzw. Objekte aus zwei verschiedenen Paketen assoziiert, besteht zwischen den beiden Paketen eine Abhängigkeit, die durch eine Importbeziehung verdeutlicht werden muss.

Man könnte einwenden, dass die Aufspaltung des Analysemodells in Informationsmodelle einerseits und Prozess- und Ablaufrahmenmodelle andererseits eine Trennung von Daten und Operationen impliziert. Dies ist jedoch nicht der Fall, da in jeder Entity-Klasse nicht nur die relevanten Attribute spezifiziert sind, sondern die Klasse auch alle Dienste anbietet, welche die Werte von Attributen in Klasseninstanzen verarbeiten. Die Boundary-, Controller- und Services-Objekte bilden lediglich das Mittel zur Verbindung von Subsystemen bzw. Prozessen und zur flexiblen Strukturierung des Ablaufs.

5.3.4 Logische und physikalische Systemstruktur

In den nachfolgenden Abschnitten wird eine prinzipielle logische und physikalische Systemstruktur vorgestellt, die als Grundlage zur Konzipierung einer geeigneten Systemarchitektur des zu entwickelnden SW-Systems verwendet werden kann.

Prinzipielle logische Systemstruktur

Werden in die statische Systemstruktur gemäß Tabelle 5.3-2 die konzeptionellen Modelle der objektorientierten Analyse gemäß Tabelle 5.3-3 einbezogen, so resultiert die prinzipielle logische Systemstruktur, die in Tabelle 5.3-4 dargestellt ist.

Die *statische Aufbaustruktur* eines umfangreichen SW-Systems gliedert sich entsprechend Tabelle 5.3-4 in folgende Substrukturen:

- die *Subsystemstruktur*, eine Struktur aus Paketen, wobei jedes Paket eine Prozessstruktur repräsentiert;

- die *Prozessstruktur*, eine Struktur aus Paketen, wobei jedes Paket eine Struktur aus Klassen für aktive Objekte einschließt, die mindestens ein abstraktes Muster für ein konkretes Prozessmodell und für Zustandsmodelle darstellt;
- die *funktionale Ablaufstruktur*, eine Struktur aus Paketen, wobei jedes Paket ein Ablaufrahmenmodell einschließt, das ein abstraktes Muster für konkrete Ablaufmodelle darstellt;
- die Struktur aus Paketen mit *Informationsmodellen*, wobei jedes Paket einem bestimmten Prozess und den nachgeordneten Ablaufmodellen zugeordnet ist.

Tabelle 5.3-4: Prinzipielle logische Systemstruktur

Ebene	**Statische Aufbaustruktur**	**Dynamische Ablaufstruktur (mit *Spezifikationsstruktur*)**
System	Paket, das das gesamte SW-System repräsentiert	
Subsysteme	Struktur aus Paketen, die jeweils ein Subsystem repräsentieren	Spezifikationsstruktur: Struktur aus Top-level-Nutzungsfällen
Prozesse	Struktur aus Paketen, die jeweils einen Prozess repräsentieren	Spezifikationsstruktur: Struktur aus übergeordneten Nutzungsfällen
aktive Objekte	Struktur aus Paketen mit Klassen für aktive Objekte, denen jeweils funktionale Abläufe zugeordnet sind	Spezifikationsstruktur: Spezieller übergeordneter Nutzungsfall bzw. spezielle Struktur aus Nutzungsfällen
	Inhalt eines Pakets: **Prozessrahmenmodell** (Klassendiagramm)	Zuordnung: ■ **Prozessmodell** (Kollaborationsdiagramm) ■ **Zustandsmodelle** (Zustandsdiagramme)
	Inhalt von Bezugspaketen: **Informationsmodelle** (Klassendiagramme)	
funktionale Abläufe	Struktur aus Paketen, die jeweils einen funktionalen Ablauf repräsentieren	Spezifikationsstruktur: Struktur aus elementaren Nutzungsfällen.
	Inhalt eines Pakets: **Ablaufrahmenmodell** (Klassendiagramm)	Zuordnung: ■ **Ablaufmodelle** (Sequenzdiagramme) ■ **Zustandsmodelle** (Zustandsdiagramme)
Bausteine	Klassen, Beziehungen	Objekte, Links, Botschaften

Tabelle 5.3-5: Inhalt der Tabelle zur Dokumentation eines Pakets

Definition eines Pakets
Name
Zweck: kurze Beschreibung
Version
Bestandteil von: NameStruktur
Klassifizierung: Subsystem | Prozess | …
Sichtbarkeit: public | protected | private
Inhalt: NameDiagramm, …
Zuordnung: Nutzungsfall, Modell

Ein Paket auf einer bestimmten Ebene der Struktur sollte jeweils für wohldefinierte und stark korrelierte Aufgaben vorgesehen werden, sodass einerseits eine klare Aufgabenteilung, andererseits eine schwache Kopplung der Pakete resultiert.

Es empfiehlt sich, die statische Aufbaustruktur dadurch zu dokumentieren, dass für jedes Paket eine Tabelle mit den wichtigsten Daten angelegt wird. Den typischen Inhalt zeigt Tabelle 5.3-5. Die Tabellen sollten durch eine Beschreibung der Gründe für die gewählte Aufbaustruktur ergänzt werden.

Die Subsystem- und Prozessstruktur repräsentieren Strukturen aus *aktiven* Einheiten, während die Ablauf- bzw. Informationsmodelle im Wesentlichen *passive* Einheiten umfassen. Ein Ablaufrahmenmodell ist der Klasse eines aktiven Controllers untergeordnet, der den funktionalen Ablauf steuert. Ein umfangreiches Informationsmodell kann bei Bedarf in eine nachgeordnete Struktur aufgespalten werden. Es kann auch eine übergeordnete Struktur haben, wenn mehrere Prozesse bzw. Subsysteme ein gemeinsames Informationsmodell nutzen.

Wie aus Tabelle 5.3-4 ersichtlich ist, umfasst ein Prozessrahmenmodell Abstraktionen für aktive Objekte, denen jeweils einerseits ein Ablaufrahmenmodell, andererseits ein *übergeordneter* Nutzungsfall zugeordnet ist. Das Ablaufrahmenmodell bildet eine gemeinsame Abstraktion für Ablaufmodelle, welche eine *objektorientierte* Umsetzung der elementaren Nutzungsfälle repräsentieren, die dem übergeordneten Nutzungsfall nachgeordnet sind.

Die Verknüpfung von statischen und dynamischen Modellen mit dem Nutzungsfall, zu dessen Realisierung die Modelle beitragen, kann durch *Verfeinerungsbeziehungen* mit dem Stereotyp «realize» verdeutlicht werden. Ein geeignetes Werkzeug kann die Beziehungen durch unsichtbare Verbindungen (Hyperlinks) verwalten. Bei geeigneter Anwahl eines Nutzungsfalls können dann automatisch die relevanten Modelle dargestellt werden.

Weitere Einzelheiten der logischen Systemstruktur werden in Abschnitt 7.3.3 behandelt.

Verhaltensstruktur

Als *Verhaltensstruktur* wird eine logische Systemstruktur (bzw. eine Teilstruktur) bezeichnet, die im Sinne einer prototypischen bzw. generischen Aufbau- und Ablaufstruktur konzipiert wird. Dadurch wird die Wartbarkeit, d. h. die Anpassbarkeit und Erweiterbarkeit der Systemstruktur an geänderte und neue Anforderungen sowie die Wiederverwendbarkeit für Anwendungen mit ähnlichen Aufgabenstellungen erhöht.

Tabelle 5.3-6 zeigt das Prinzip einer Verhaltensstruktur. Einer Struktur aus Spezifikationselementen wird eine Struktur aus Realisierungselementen zugeordnet. Die wesentlichen Spezifikationselemente sind Nutzungsfälle, die durch Aktivitäten und Workflow- sowie Reaktionsmodelle ergänzt sind. Die Grundlage für die Konzipierung der Realisierungsstruktur bilden Kollaborationen mit zugeordneten Interaktionen, die zusammen mit den Nutzungsfällen in Subsystem-Pakete eingebunden und auf nachgeordneten Strukturebenen verfeinert werden. Einzelheiten der Verhaltensstruktur werden im Zusammenhang mit dem objektorientierten Entwurf in Kapitel 7 behandelt.

Tabelle 5.3-6: Verhaltensstruktur: Prinzip

Ebene	Spezifikationsstruktur	Realisierungsstruktur
Subsysteme	Top-level-Nutzungsfälle	Struktur aus Subsystem-Paketen mit optionalen Schnittstellen
Prozesse	Struktur aus Nutzungsfällen	Kollaborationen und zugeordnete Interaktionen
aktive Objekte	Struktur aus Nutzungsfällen	Kollaborationen, Strukturen aus aktiven Klassen und zugeordnete Interaktionen
funktionale Abläufe	Struktur aus elementaren Nutzungsfällen	Strukturen aus passiven Klassen und zugeordnete Interaktionen

Prinzipielle physikalische Systemstruktur

Die prinzipielle physikalische Systemstruktur setzt sich zusammen aus der *modularen SW-Struktur* und der *konfigurierten Struktur*. Die modulare SW-Struktur umfasst die kompilierbaren, integrierbaren und ausführbaren SW-Komponenten. Die konfigurierte Struktur zeigt das Zielrechnersystem und die Zuordnung von SW-Komponenten zu relevanten Hardware-Komponenten. Die beiden Strukturen sind in Tabelle 5.3-7 gegenübergestellt.

Die physikalische Systemstruktur ist eine hierarchische Struktur, die einerseits das SW-System, andererseits das Hardware-System, d. h. das Zielrechnersystem mit den Hard-

Tabelle 5.3-7: Prinzipielle physikalische Systemstruktur

Ebene	Modulare SW-Struktur	Konfigurierte Struktur (HW mit residenten SW-Komponenten)	Ebene
SW-System	Systemkontext (freies Blockdiagramm)	vernetzte Knoten (Knotendiagramm)	HW-System
Subsysteme	Subsystemstruktur (Blockdiagramm bzw. Komponentendiagramm)	vernetzte Knoten (Knotendiagramm)	lokale Knoten
Prozesse	Struktur aus Software-Komponenten (Komponentendiagramm)	lokaler Knoten (Knotendiagramm)	Prozessoren Speichergeräte
Funktionale Abläufe	Struktur aus Software-Komponenten (Komponentendiagramme)	Knoten mit residenten Software-Komponenten	residente SW-Komponenten

ware-Komponenten, repräsentiert. Die Ebenen der Struktur entsprechen denen der logischen Systemstruktur.

Die modulare SW-Struktur ist eine statische Struktur aus SW-Komponenten, die Kompilierungs-, Integrations- und Ladeeinheiten, aber auch Klassen- und Komponentenbibliotheken repräsentieren, die für die Kompilierung, Integration bzw. für den Ablauf des SW-Systems benötigt werden.

Der Systemkontext macht das Zusammenwirken der peripheren Geräte und externen DV-Systeme mit dem SW-System sichtbar. Ein wesentlicher Aspekt ist die Definition der dafür erforderlichen Schnittstellen, die als Bestandteil des SW-Systems bereitgestellt werden müssen. Zur grafischen Gestaltung des physikalischen Systemkontexts ist in der UML kein Diagramm mit spezifischer Semantik vorgesehen. Es wird deshalb empfohlen, den Systemkontext mit Hilfe eines geeigneten grafischen Editors als übersichtliches und leicht verständliches Blockdiagramm zu gestalten und dieses in die physikalische Systemstruktur zu integrieren.

Die physikalische Subsystemstruktur zeigt die Gliederung des SW-Systems in Subsysteme entsprechend der statischen Paketstruktur. Dabei wird im ersten Ansatz vorausgesetzt, dass im Falle eines auf mehrere Knoten verteilten Systems ein Subsystem die strukturelle Einheit ist, die einem bestimmten Knoten zugeordnet wird. Ein wesentlicher Aspekt ist auch hier die Definition der Schnittstellen, die als Bestandteil der Subsysteme vorgesehen werden müssen. Alternativen der Darstellung sind ein Komponentendiagramm, in dem die Subsysteme durch SW-Komponenten mit expliziten Schnittstellen dargestellt sind, oder ein freies Blockdiagramm. Bei einfachen Systemstrukturen kann die Subsystemstruktur in das Kontextdiagramm integriert werden.

Für die physikalische Strukturierung eines Subsystems in Prozesse wird angenommen, dass bei Einsatz eines *Mehrprozessorsystems* ein Prozess die strukturelle Einheit ist, die einem bestimmten Prozessor zugeordnet wird. Darüber hinaus sind für die Strukturierung eines Subsystems in SW-Komponenten physikalische Gemeinsamkeiten entscheidend, z. B. die Nutzung der gleichen Klassenbibliothek für die Implementierung, die Zuordnung zum gleichen Speichermodul etc.

Weitere Einzelheiten der physikalischen Systemstruktur werden in Abschnitt 7.3.3 behandelt.

5.3.5 Aktivitäten

Die folgenden Aktivitäten sind für den Entwurf der Systemarchitektur charakteristisch:

- Konzipierung der übergeordneten logischen Systemstruktur
- Konzipierung des Systemkontexts;
- Konzipierung der physikalischen Subsystem- und Prozessstruktur.

Der Entwurf der Systemarchitektur wird dadurch abgeschlossen, dass eine Grundlage für Entwurfsentscheidungen geschaffen wird. Wesentlich sind vor allem folgende Aspekte:

- die Berücksichtigung der Entwurfskriterien mit Festlegung von Prioritäten – siehe Kapitel 7;
- die Berücksichtigung der voraussichtlichen Änderungen und ihrer Auswirkungen auf die Modifizierbarkeit und Erweiterbarkeit der Struktur (Perspektive der Änderbarkeit);
- die Strategie beim Entwurf der logischen und physikalischen Modelle;
- die Planung der Reihenfolge und des jeweils vorgesehenen Ausbaus der diversen Systemprototypen (Zwischenprodukte), die im Zuge der evolutionären Entwicklung des SW-Systems entwickelt und auf dem aktuellen Zielrechnersystem evaluiert werden.

Während bei einem wenig umfangreichen Projekt (das maximal drei Mannjahre Aufwand umfasst) in der Regel ein zweiter evolutionärer Zyklus zur Verbesserung des *ersten Versuchs* ausreicht, sollte man bei einer anspruchsvollen Entwicklung je nach Umfang, Komplexität und Risiko drei bis maximal fünf Entwicklungszyklen vorsehen.

Die Systemarchitektur und die Entwurfsentscheidungen, deren Realisierung mit einem gewissen Risiko verbunden sein kann, sollten möglichst bereits im ersten Systemprototyp erprobt werden. In Extremfällen kann es notwendig werden, dass durch *exploratives* Prototyping alternative Lösungen ausprobiert und die *trade-offs* evaluiert werden.

Konzipierung der übergeordneten logischen Systemstruktur

Die Subsystem- und Prozessstruktur und die nachgeordnete funktionale Struktur können entsprechend Tabelle 5.3-4 aus der Struktur der Nutzungsfälle abgeleitet werden – vgl. dazu Abschnitt 5.2.4.

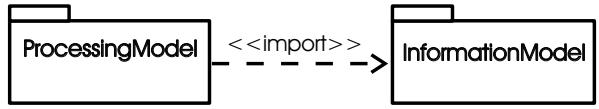

Abbildung 5.3-5: Prozess und zugeordnetes Informationsmodell

Ein Prozess ist im Allgemeinen mit einem Informationsmodell verknüpft. Dies kann im Prinzip entsprechend Abbildung 5.3-5 modelliert werden. Darin umfasst das Paket *ProcessingModel* die Pakete, die das Prozessrahmenmodell und die davon abhängigen Ablaufrahmenmodelle repräsentieren. Dagegen schließt das Paket *InformationModel* die Pakete ein, die insgesamt das Informationsmodell bilden.

Es empfiehlt sich, die *funktionale Struktur* zunächst entsprechend den auf der Prozessebene definierten Nutzungsfällen zu organisieren. Dies bedeutet, dass den Nutzungsfällen, die einem Prozess zugeordnet sind, jeweils ein Prozessrahmenmodell mit nachgeordneten Ablaufrahmenmodellen gegenübersteht. Einem Ablaufrahmenmodell ist ein übergeordneter Nutzungsfall zugeordnet, einem konkreten Ablaufmodell ein elementarer Nutzungsfall. Es ist denkbar, dass sich im weiteren Verlauf der Analyse bzw. des Entwurfs zeigt, dass die Organisation der funktionalen Struktur nach anderen Gesichtspunkten Vorteile bietet.

Abhängigkeiten zwischen Paketen dergestalt, dass ein Paket Dienste bzw. Informationen aus einem anderen Paket benötigt, werden durch Importbeziehungen verdeutlicht. Dabei sollte man sich zunächst auf die Beziehungen beschränken, die offensichtlich sind. Abbildung 5.3-6 zeigt ein Beispiel einer statischen Systemstruktur für eine betriebswirtschaftliche Anwendung.

Konzipierung des Systemkontexts

Eine besondere Rolle für die Konzipierung der physikalischen Systemstruktur spielen der *Systemkontext* und die darauf abgestimmte Gestaltung der *physikalischen Subsystem-* und *Prozessstruktur*. Zweck des Systemkontexts ist die Darstellung des Zusammenwirkens der peripheren Geräte und externen DV-Systeme mit dem SW-System. Besonders geeignet dafür ist ein frei gestaltetes Blockdiagramm.

Das SW-System wird dabei durch einen zentralen Block dargestellt, der mit externen Blöcken verbunden ist, welche die peripheren Geräte und externen Systeme repräsentieren. Die Schnittstellen, die als Bestandteile des SW-Systems entwickelt werden müssen, werden zweckmäßig innerhalb des zentralen Blocks am Rand positioniert. Pfeilspitzen am Ende der Verbindungen markieren einen uni- bzw. bidirektionalen Ereignis- bzw. Botschaftenfluss, der relevante Signale bzw. Eingangsdaten und Ergebnisse transportiert. Ist als Verbindung zu einem Standardgerät, z. B. zu einem Drucker, eine Standard-Schnittstelle vorgesehen, sollte diese speziell gekennzeichnet oder weggelassen werden.

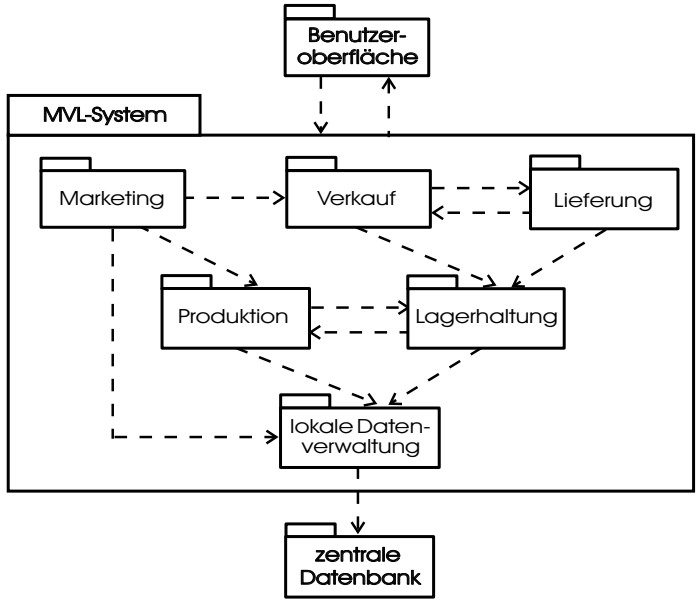

Abbildung 5.3-6: Beispiel einer statischen Systemstruktur

In das Diagramm können auch zusätzliche Aspekte einbezogen werden, die für den Systemkontext relevant sind und das Verständnis der Systemnutzung erhöhen, z. B.:

- die direkten und indirekten *Systembenutzer;*
- *Sachmittel*, welche an Benutzer bzw. von diesen übermittelt werden, wie z. B. Aufträge, Quittungen etc.;
- *Signale, Daten und Aufträge*, die über die Schnittstelle transportiert werden;
- die *lokalen Bereiche*, in denen die Systemnutzer tätig bzw. wo die angeschlossenen Geräte und externe DV-Systeme verfügbar sind.

Abbildung 5.3-7 zeigt ein Beispiel eines frei gestalteten Kontextdiagramms.

Das Kontextdiagramm wird durch die zunächst informale Spezifikation der Schnittstellen ergänzt. Dabei werden die Signale, Daten und Aufträge, die über die Schnittstelle transportiert werden, beschrieben. Falls spezielle Geräte angeschlossen sind, werden auch die Operationen, die zur Transformation, Zwischenspeicherung und Weiterleitung der Daten benötigt werden, spezifiziert und Klassen zu ihrer Realisierung vorgesehen.

Konzipierung der physikalischen Subsystem- und Prozessstruktur

Die Gliederung des Systems in Subsysteme bzw. Prozesse muss der statischen Paketstruktur entsprechen. Typisch für das Zusammenwirken der Subsysteme bzw.

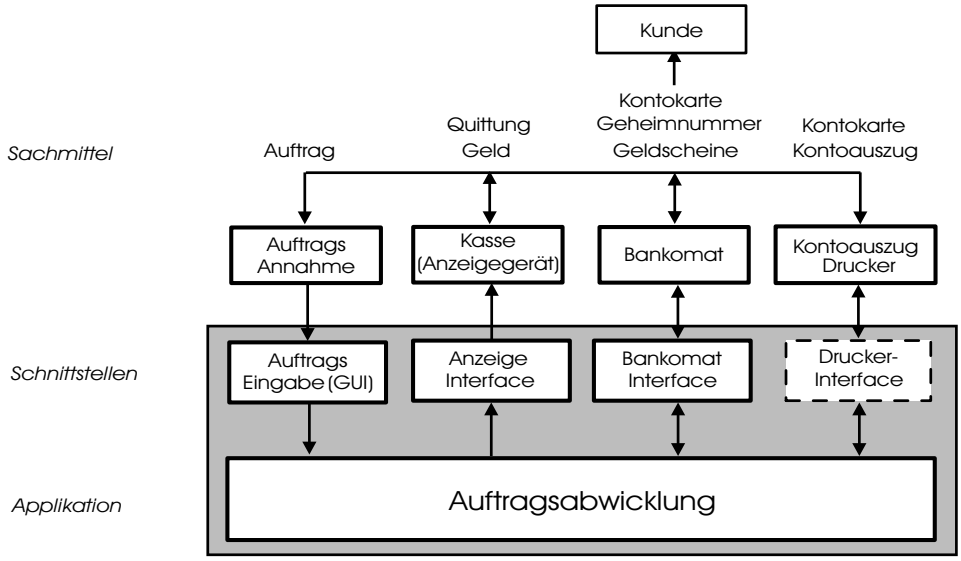

Abbildung 5.3-7: Physikalischer Systemkontext (Beispiel)

Prozesse sind netzförmige Strukturen. Alternativen der Darstellung sind ein freies Blockdiagramm oder ein Komponentendiagramm.

5.4 Zusammenfassung

Die Analyse des Systemverhaltens bezieht sich auf die operationellen Anforderungen, welche die extern sichtbaren Wirkungen des SW-Systems bestimmen. Eine geeignete Methode zur Unterstützung der Analyse ist die Use-Case-Analyse. Dabei werden die Objekte der Systemumgebung (Aktoren) und relevante Vorgänge im SW-System (Nutzungsfälle) identifiziert und zueinander in Beziehung gesetzt. Wesentliche Erkenntnisse für die weitere Systementwicklung gewinnt man dadurch, dass die Charakteristika der einzelnen Nutzungsfälle ermittelt und die Interaktionen der elementaren Nutzungsfälle mit den relevanten Aktoren analysiert und in Form von Szenarios spezifiziert werden.

Die Ergebnisse der Analyse des Systemverhaltens bilden die wesentliche Grundlage für die eigentliche Entwicklung des SW-Systems, die in den nachfolgenden Kapiteln erläutert wird. Das resultierende Dokument repräsentiert deshalb einen Vertrag zwischen dem Auftraggeber bzw. dem Fachbereich und dem Bereich, der das Entwicklungsprojekt durchführt.

An die Analyse des Systemverhaltens schließt sich der Entwurf der Systemarchitektur an. Dabei wird eine logische und physikalische Systemstruktur entwickelt, in welche

die Modelle eingebunden werden, die im weiteren Verlauf der Analyse und des Entwurfs entstehen.

In den nachfolgenden Kapiteln 6 und 7 werden die Abhängigkeiten zwischen dem Modell der Nutzungsfälle (Use-Case-Modell) und den logischen und physikalischen Modellen im Detail beschrieben. Die Abbildung im vorderen Einschlag des vorliegenden Buchs vermittelt einen zusammenfassenden Überblick, auf den der Leser einen schnellen Zugriff hat.