Gründen der Lesbarkeit und Verständlichkeit einem *fast* sequenziellen Erklärungsmuster und beginnen nun mit der Anwendungsfallmodellierung.

4.1 Anwendungsfallmodellierung

In diesem Schritt werden die vom Kunden gewünschten Systemoperationen identifiziert. Dies geschieht entweder durch Analyse einer vom Kunden zur Verfügung gestellten, in der Regel textuellen Problemstellung und/oder durch Diskussion mit den Kunden. Systemoperationen werden in UML als Anwendungsfälle bezeichnet. Anwendungsfälle wurden von Jacobson in der objektorientierten Modellierungsmethode OOSE eingeführt ([Jaco92]). Damit wurde erfolgreich der bei Vertretern der objektorientierten Entwicklung weit verbreiteten »alles ist ein Objekt und strukturierte Entwicklung ist schlecht«-Mentalität entgegengewirkt. Anwendungsfälle entsprechen den Funktionen bzw. Prozessen der strukturierten Entwicklung, die bei erstmaliger Anwendung der funktionalen Zerlegung des zu entwickelnden Systems generiert werden. Anwendungsfälle repräsentieren die Anforderungen der Kunden. Damit wird auch der Erfahrung Rechnung getragen, dass Kunden in erster Linie an der Funktionalität des zu entwickelnden Systems und nicht an Objekten interessiert sind.

Anwendungsfallmodell (use case model)

Das Ergebnis der Anwendungsfallmodellierung ist das *Anwendungsfallmodell*. Dies umfasst die identifizierten Anwendungsfälle, die Akteure, d.h. Personen und andere externe Systeme, die mit diesen Anwendungsfällen interagieren, und das zu entwickelnde System als solches. Diese Information findet in unterschiedlichen Dokumenten ihren Niederschlag, nämlich in einem globalen Anwendungsfalldiagramm und in detaillierten, sowohl textuellen als auch grafischen Beschreibungen für jeden Anwendungsfall.

Das zu beschreibende System muss nicht notwendigerweise implementiert werden. Die Anwendungsfallmodellierung kann auch z.B. für eine Istanalyse eines Betriebs und seiner Abläufe eingesetzt werden.

4.1.1 Identifikation von Akteuren

Wer mit dem System interagieren soll, wird in Form von Akteuren festgelegt. Obwohl UML per se keine Klassifikation von Akteuren vorsieht, ist eine solche im Umgang mit Akteuren sehr hilfreich. Akteure können unterschiedlich klassifiziert werden. Eine Klassifikation unterscheidet zwischen menschlichen und nicht menschlichen Akteuren. Letztere sind u.a. andere Softwaresysteme (z.B. ein E-Mail-System), Ein- und Ausgabegeräte (z.B. eine Faxkarte) und externe Systeme (z.B. ein Bankinstitut). Menschliche

Klassifikation von Akteuren:

menschlich / nicht menschlich

Akteure im CALENDARIUM sind *Benutzer* und *Administrator*, nicht menschliche Akteure sind *Faxsystem* und *E-Mail-System*).

primär / sekundär

Eine zweite Klassifikation unterscheidet zwischen primären und sekundären Akteuren. Ein *primärer Akteur* ist ein Hauptnutznießer des zu implementierenden Systems. Er aktiviert die wesentlichen Systemoperationen, d.h. Anwendungsfälle, um ihre Funktionalität in Anspruch zu nehmen. Im Gegensatz dazu wird der *sekundäre Akteur* gebraucht, um das korrekte Funktionieren des Systems sicherzustellen. Ein sekundärer Akteur ist niemals »Kunde« eines Systems, sondern immer »Mittel zum Zweck«. Im CALENDARIUM ist der Benutzer ein primärer Akteur. Der Administrator, das E-Mail-System und das Faxsystem sind sekundäre Akteure, die dazu beitragen, die gewünschte Systemfunktionalität zu realisieren.

aktiv / passiv

Eine dritte Klassifikation unterscheidet zwischen *aktiven Akteuren*, die selbst Systemoperationen anstoßen können, und *passiven Akteuren*, die ausschließlich vom System angestoßen werden. Primäre Akteure werden in der Regel auch aktiv sein, während sekundäre Akteure entweder aktiv oder passiv sind. So ist der Administrator ein sekundärer, aktiver Akteur, der Systemoperationen zur Verwaltung der Benutzer aufruft, während das E-Mail-System und das Faxsystem in unserem Kontext eindeutig sekundäre, passive Akteure sind.

Finden von Akteuren

Obwohl wir im CALENDARIUM vorerst alle Benutzer des Systems nur durch einen einzigen Akteur repräsentieren, wird man in der Regel mehrere Kategorien von Benutzern unterscheiden, z.B. abhängig von ihrer Autorisierung, mit dem System zu arbeiten (z.B. »*superuser*« und »Standardbenutzer«), oder abhängig von ihrem Wissensstand (z.B. geübter Benutzer und naiver Benutzer). Diese Kategorien beeinflussen die Benutzerschnittstellendefinitionen, und es ist daher wesentlich, sich Gedanken darüber zu machen. Generell ist die Frage nach den Akteuren eines zu implementierenden Systems sehr relevant, da man von den Akteuren bzw. ihren Anforderungen auf die unterschiedlichen Anwendungsfälle des Systems schließen kann.

Benutzerschnittstelle *S. 228*

Die Beantwortung der folgenden Fragen hilft ebenfalls, die Akteure eines Systems zu identifizieren (nach [Erik98]):

- ? Wer benötigt Unterstützung für die tägliche Arbeit?
- ? Wer ist für die Systemadministration zuständig?
- ? Mit welchen externen Geräten muss das System kommunizieren können?
- ? Mit welchen externen (Software-)Systemen muss das System kommunizieren können?
- ? Wer interessiert sich für die Ergebnisse des Systems?

4.1.2 Identifikation von Anwendungsfällen

Ein Anwendungsfall beschreibt ein bestimmtes Verhalten, das von einem zu entwickelnden System erwartet wird. Alle Anwendungsfälle zusammen machen die Systemfunktionalität des Gesamtsystems aus. Anwendungsfälle können auf verschiedene Weise identifiziert werden. Erstens durch Sammeln der Kundenwünsche und zweitens durch Analyse der textuellen Problemstellung. Für diesen zweiten Ansatz adaptieren wir die von Abbott vorgeschlagene Vorgangsweise zur Erstellung eines ersten Klassendiagramms ([Abbo83]), so dass sie für die Erkennung von Anwendungsfällen eingesetzt werden kann.

Viele der benutzerinitiierten Anwendungsfälle sind in der Regel auch als Zeitwortphrasen in der textuellen Problemstellung versteckt. Um nun den Vorgang des Herausfilterns von Anwendungsfällen aus der Problemstellung zu veranschaulichen, fasst Tabelle 4–1 die wesentlichen Zeitwortphrasen der Problembeschreibung aus Unterkapitel 1.5 und ihre Interpretation zusammen. Die Zeitwortphrasen sind nach den vorkommenden Zeitwörtern alphabetisch sortiert. Die Interpretation konzentriert sich auf die Entscheidung, ob die jeweilige Zeitwortphrase als (Teil eines) Anwendungsfall(s) weiter untersucht werden soll oder nicht.

Problemstellung ↗S. 13

Natürlich kann nicht postuliert werden, dass diese Liste von Anwendungsfällen die Funktionalität des Systems CALENDARIUM vollständig beschreibt. Aber die vom Kunden geforderten Systemoperationen sollten bei hinreichender Vollständigkeit der Problembeschreibung durch dieses Verfahren abgedeckt werden. Das Finden weiterer Anwendungsfälle, die zur Erfüllung der Funktionalität der bereits identifizierten Anwendungsfälle benötigt werden, ist u.a. Aufgabe der folgenden Verfeinerungsschritte.

Tab. 4–1 *Extrahieren von Anwendungsfällen aus der textuellen Problemstellung*

Zeitwortphrase	Modellierungsentscheidung
abfragen von Details eines Termins	*Termin abfragen*
anbieten von Ansichten	*Auswahl von Ansichten*
ändern eines Termins	*Termin ändern*
Notifikation **angeben**	Teil von *Termin erfassen*
Wiederholungsdauer und -frequenz **angeben**	Tritt in Zusammenhang mit dem Erfassen eines Serientermins auf; *Serientermin erfassen* ist eine Erweiterung von *Termin erfassen*

Identifizierte Anwendungsfälle werden kursiv dargestellt

Erweiterung eines Anwendungsfalls: extend-Beziehung ↗S. 107

Zeitwortphrase	Modellierungsentscheidung
Fälligkeitszeitpunkt von *to-do*-Eintrag in Kalender **aufnehmen**	*to-do-Eintrag erfassen*; ist hinlänglich unterschiedlich zu *Termin erfassen* und daher keine Erweiterung davon
auftreten einer Terminkollision	Hat den Charakter eines Hinweises in der Problemstellung und wird daher nicht als Anwendungsfall modelliert
Teilnehmer im System **bekannt machen**	*Benutzer verwalten*
Teilnehmer von Termin **benachrichtigen**	*Teilnehmer verständigen*
Dauer **bestimmen**	Teil von *Terminvorschlag finden*
mehrere Terminkalender gleichzeitig **betrachten**	Wird innerhalb von *Auswahl von Ansichten* behandelt
Termine **charakterisieren**	Teil von *Termin erfassen* bzw. *ändern*
to-do-Liste nach Fälligkeit sortiert **darstellen**	Teil von *Auswahl von Ansichten*
Beginnzeitpunkt der *to-do*-Listendarstellung **definieren**	Teil von *to-do-Eintrag erfassen*
Notifikation **definieren**	Synonym zu »*Notifikation angeben*«
einen Kalender eines anderen Benutzers **einsehen**	Teil von *Auswahl von Ansichten*
Darstellungsaspekte **einstellen**	Teil von *Einstellungen verwalten*
neuen Termin **eintragen**	*Termin erfassen*
Terminvorschlag **ermitteln**	*Terminvorschlag finden*
Termine **exportieren**	*Termine exportieren*
Termine des Kalenderbesitzers **finden statt**	Beschreibt einen Zustand, der keine spezifische Aktion zur Folge hat
Tage in der Jahresansicht grafisch **hervorheben**	Teil von *Auswahl von Ansichten*
Details eines Termins **lesen**	Synonym zu »**abfragen** von Details eines Termins«
Termin **löschen**	*Termin löschen*

Zeitwortphrase	Modellierungsentscheidung
Verschiebungsvorschläge **machen**	Teil von *Terminvorschlag finden*
Ausnahmeregelungen bei Serienterminen **organisieren**	Da Ausnahmen auf der Ebene von Einzelterminen spezifiziert werden, wird dies innerhalb von *Termin ändern* erledigt
Teilnehmer von einem Eintrag **unterrichten**	Synonym zu »*Teilnehmern von Termin benachrichtigen*«
Werktagstermine und allgemeine Termine **unterscheiden**	Teil von *Serientermin erfassen*
Fixtermine und verschiebbare Termine optisch **unterscheiden**	Teil von *Auswahl von Ansichten*; die Voreinstellungen dazu werden in *Einstellungen verwalten* festgelegt
Termintypen optisch **unterscheiden**	dito
Exportformate **unterstützen**	Teilaufgabe von *Termine exportieren*
Definition von Personengruppen **unterstützen**	*globale Benutzergruppen verwalten* und *private Benutzergruppen verwalten*
Termin **vereinbaren**	*Termin erfassen*
über Zugriffsrechte **verfügen**	Dieser Zustand wird mittelbar durch die beiden Anwendungsfälle *Zugriffsrechte verwalten* und *Benutzer verwalten* sichergestellt
Termintypen **verwalten**	*Termintypen verwalten*
CALENDARIUM **verwaltet** Termine	Kein eigener Anwendungsfall, weil dies die gesamte Systemfunktionalität von CALENDARIUM umfasst
Termin **vorankündigen**	*Termin vorankündigen*
Suchzeitintervall **vorgeben**	Teil von *Terminvorschlag finden*
anderen Benutzern Berechtigungsstufen **zuordnen**	*Zugriffsrechte verwalten*
Teilnehmer einem Termin **zuordnen**	Teil von *Termin erfassen*
Teilnehmer einer Notifikation eines Termins **zuordnen**	Teil von *Termin erfassen*
Termintypen **zuordnen**	Teil von *Termin erfassen*

Aus Platzgründen werden im Folgenden nur einige ausgewählte Anwendungsfälle aus Tabelle 4–1 weiter untersucht.

Für jeden Anwendungsfall soll eine Kurzbeschreibung die Ziele des Anwendungsfalls umreißen. In Abbildung 4–1 sind beispielhaft der Anwendungsfall *Termin erfassen* und eine Kurzbeschreibung desselben in einer Notiz angegeben.

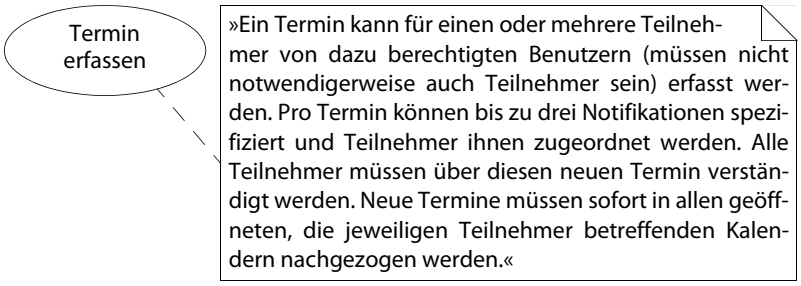


Abb. 4–1
Darstellung und Kurzbeschreibung eines Anwendungsfalls

Diese Kurzbeschreibung stellt natürlich nur eine »erste Näherung« einer genauen Spezifikation eines Anwendungsfalls dar, die in der Folge verfeinert werden muss – siehe dazu Abschnitt 4.1.4. Dennoch ist bereits ein interessantes Detail zu erkennen. Obwohl in der Problemstellung die Aktualisierung von Kalendern nicht explizit erwähnt ist, wird diese Forderung in der Kurzbeschreibung berücksichtigt. Der Grund ist, dass sich diese Anforderung mittelbar aus zwei anderen Anforderungen aus der Problemstellung ergibt, nämlich »Entwicklung eines mehrbenutzerfähigen Kalenderprogramms« und »einen Kalender eines anderen Benutzers einsehen«. Demzufolge können mehrere Benutzer gleichzeitig den Terminkalender eines anderen Benutzers geöffnet haben. Im Sinne einer konsistenten und korrekten Wiedergabe eines Terminkalenders müssen daher neue Termine sofort in allen geöffneten Kalendern nachgetragen werden.

Verfeinerung der Beschreibung des Anwendungsfalls
S. 200

Die identifizierten Anwendungsfälle, das sie umschließende System und die zugehörigen Akteure werden in einem Anwendungsfalldiagramm zusammengefasst. Abbildung 2–56 auf Seite 105 stellt bereits die erste Fassung des Anwendungsfalldiagramms für das CALENDARIUM dar. Mit »erster Fassung« soll angedeutet werden, dass das Anwendungsfalldiagramm als Ergebnis der Anwendungsfallmodellierung nicht »auf einmal« entsteht, sondern iterativ in mehreren Zyklen verfeinert wird (vgl. die Verfeinerung des Anwendungsfalldiagramms in Abb. 4–2 auf S. 198 sowie in Unterkapitel 4.3).

Erinnerung an Abb. 2–56

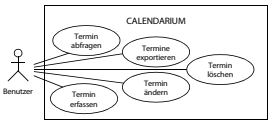


Abbildung 2–56 zeigt fünf Anwendungsfälle, die durch Analyse der textuellen Problemstellung identifiziert wurden (vgl. Tab. 4–1) und den Kern von CALENDARIUM darstellen. Diese sind in der Reihenfolge ihrer Nennung in Tabelle 4–1 *Termin abfragen*, *Termin ändern*, *Termin erfassen*, *Termine exportieren* und *Termin löschen*. Alle fünf Anwendungsfälle werden vom vorläufig einzigen primären Akteur *Benutzer* initiiert. Sie stellen also in erster Linie die vom Benutzer gewünschten Systemoperationen dar.

Analysiert man nun z.B. Tabelle 4–1 und die Kurzbeschreibung des Anwendungsfalls *Termin erfassen* aus Abbildung 4–1 genauer, so lassen sich sogleich weitere Anwendungsfälle identifizieren, die von *Termin erfassen* zur Erfüllung der geforderten Funktionalität benötigt werden. Diese sind *Teilnehmer verständigen* und *Kalender aktualisieren*.

Abb. 4–2
Anwendungsfalldiagramm für CALENDARIUM (Verfeinerung)

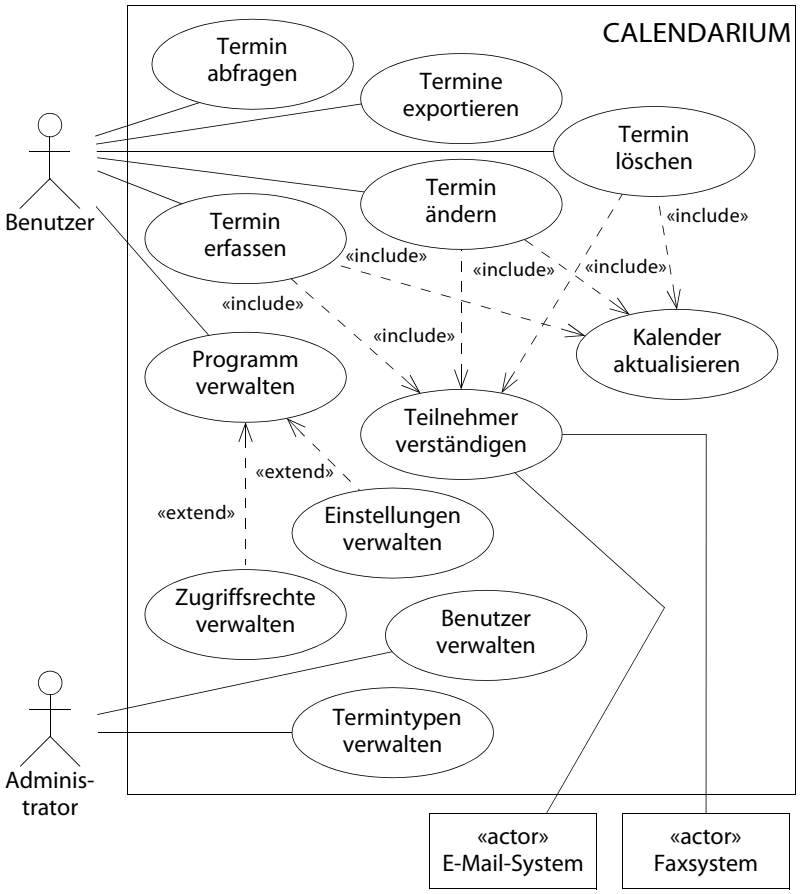


Abbildung 4–2 zeigt zusätzlich zum Benutzer weitere Akteure und ihre Kommunikationsbeziehungen mit relevanten Anwendungsfällen. *E-Mail-System* und *Faxsystem* werden benötigt, um die Teilnehmer über neue Termine bzw. das Herannahen eines Termins zu verständigen. Der *Administrator* gewährleistet das Funktionieren des Gesamtsystems, indem er die Benutzerverwaltung übernimmt und Termintypen (z.B. private Termine, Geschäftstermine) einrichtet.

Abschließend noch ein Wort zur Granularität von Anwendungsfällen. Was genau einen Anwendungsfall ausmacht, ist zurzeit eher eine Frage des Geschmacks als die einer präzisen Definition (vgl. auch die entsprechende Diskussion über Ablaufmodellierung in Unterkapitel 2.7). Erst wenn ge-

Ablaufmodellierung ↗S. 160

nug Erfahrung im Einsatz anwendungsfallorientierter Vorgehensmodelle für unterschiedliche Anwendungsbereiche vorliegt, wird man hierfür genauere Faustregeln aufstellen können.

4.1.3 Überarbeitung des Anwendungsfalldiagramms

Für nichttriviale Problemstellungen kann das Anwendungsfalldiagramm schnell sehr unübersichtlich werden. Dies trifft bis zu einem gewissen Grad bereits für die erste Verfeinerung des CALENDARIUM-Diagramms (Abb. 4–2) zu. Für derartige Fälle bietet UML einen Abstraktionsmechanismus in Form von Paketen an. Im Sinne der im Exkurs über die Verwendung von Paketen definierten *Bottom-up*-Vorgangsweise werden nun jene Anwendungsfälle, die ein gemeinsames Ziel verfolgen, jeweils in einem eigenen Paket gekapselt. Die beiden resultierenden Pakete sind in den Abbildungen 4–3 und 4–4 dargestellt.

Verwendung von Paketen ↗ *S. 101*

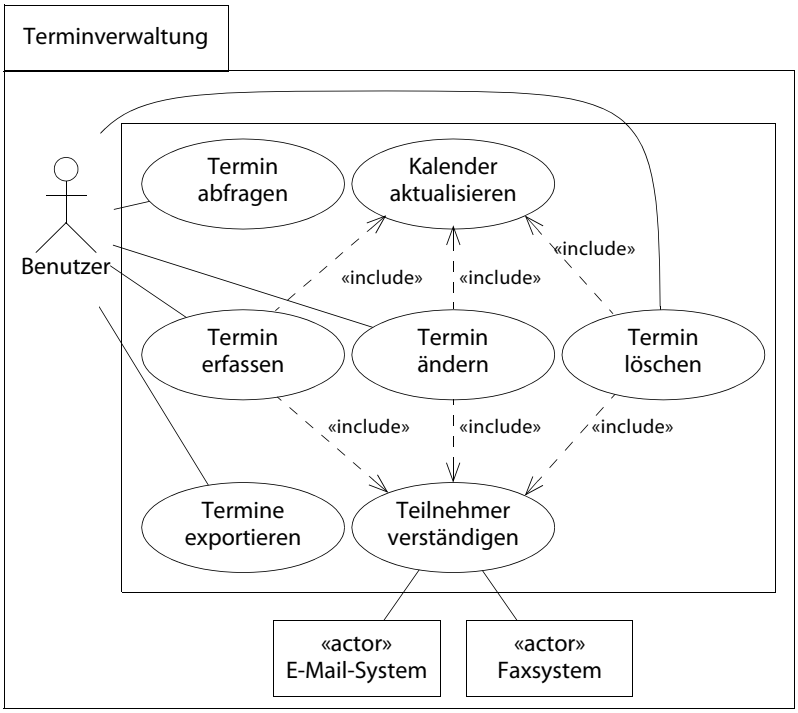
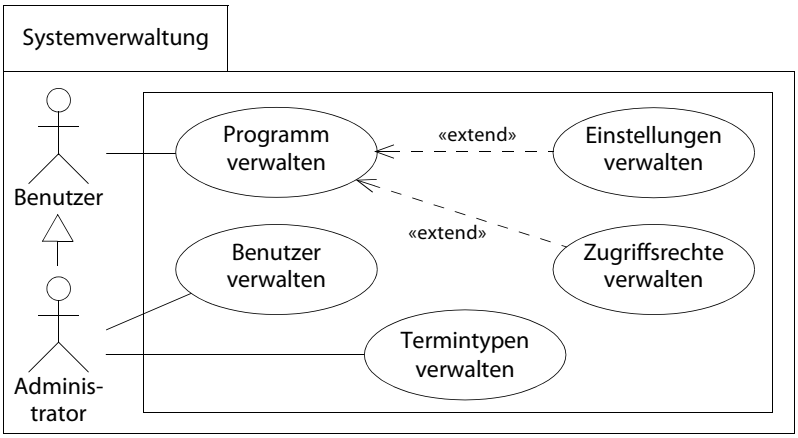


Abb. 4–3
Paket **Terminverwaltung** *(Kapselung von terminrelevanten Anwendungsfällen)*

Es werden neben den Anwendungsfällen auch die zugeordneten Akteure und alle Beziehungen der Anwendungsfälle im jeweiligen Paket angezeigt. So beinhaltet das Paket *Terminverwaltung* in Abbildung 4–3 alle jene Anwendungsfälle, die sich mit Terminen im engeren Sinn beschäftigen, während Abbildung 4–4 im Paket *Systemverwaltung* alle jene Anwendungs-

fälle zusammenfasst, die das Verwalten des Kalenderprogramms als solches zur Aufgabe haben, wie z.B. das Anpassen diverser Einstellungen an Benutzerwünsche und das Erfassen neuer Benutzer.

Abb. 4–4
Paket **Systemverwaltung** *(Kapselung von kalendersystemrelevanten Anwendungsfällen)*

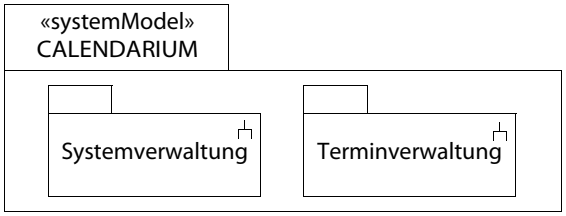


«systemModel»-Paket S. 101

Subsystem S. 99

Das zu entwickelnde System wird als Paket mit dem vordefinierten Stereotyp *«systemModel»* modelliert. Abbildung 4–5 zeigt das *«systemModel»*-Paket CALENDARIUM mit den eingebetteten Subsystemen *Systemverwaltung* und *Terminverwaltung*. Derzeit stellt es natürlich nur jene Sicht auf unser Gesamtsystem dar, die die Anwendungsfälle betrifft. Weitere Sichten werden beim Einführen der jeweiligen Modellelemente nachgezogen, z.B. nach der Strukturmodellierung in der Analyse.

Abb. 4–5
«systemModel»-Paket CALENDARIUM



4.1.4 Beschreibung eines Anwendungsfalls

Abhängig von seiner Komplexität sollte für jeden Anwendungsfall, ausgehend von seiner textuellen Kurzbeschreibung, eine detaillierte Beschreibung erzeugt werden, die auch die einzelnen auszuführenden Schritte bereits verbal identifiziert. Dies kann einerseits in Form von Freitext erfolgen, andererseits in Form von strukturiertem Text und – etwas formaler – in Form eines Aktivitätsdiagramms. Die beiden letzten Varianten werden im Folgenden vorgestellt.

Strukturierter Text

UML macht keine Vorgaben bezüglich des Einsatzes von natürlichsprachlichem Text zur Beschreibung von Anwendungsfällen. Die Erfahrung der Autoren hat aber gezeigt, dass sich die Spezifikation einer entsprechenden Dokumentvorlage (»leeres Formular«) lohnt. Nicht zuletzt deshalb, um eine einheitliche Darstellung, ein ähnliches Detaillierungsniveau und eine gemeinsame Gesprächsgrundlage zwischen unterschiedlichen Entwicklern, Kunden und Benutzern zu haben. In enger Anlehnung an [Cock97] schlagen die Autoren die Dokumentvorlage aus Tabelle 4–2 zur detaillierten Beschreibung eines Anwendungsfalls vor. Die einzelnen Einträge werden beispielhaft für den Anwendungsfall *Termin erfassen* erläutert.

Dokumentvorlage zur Beschreibung eines Anwendungsfalls

Tab. 4–2
Beschreibung eines Anwendungsfalls in Form von strukturiertem Text

Anwendungsfall	**Termin erfassen**
Kurzbeschreibung	Ein Termin kann für einen oder mehrere Teilnehmer von dazu berechtigten Benutzern (müssen nicht notwendigerweise auch Teilnehmer sein) erfasst werden. Pro Termin können bis zu drei Notifikationen spezifiziert und Teilnehmer ihnen zugeordnet werden. Alle Teilnehmer müssen über diesen neuen Termin verständigt werden. Neue Termine müssen sofort in allen geöffneten, die jeweiligen Teilnehmer betreffenden Kalendern nachgezogen werden.
Vorbedingung	Benutzer ist dem System bekannt.
Nachbedingung	Neuer Termin ist erfasst. Alle Teilnehmer sind verständigt und gegebenenfalls informiert, dass es aus Berechtigungsgründen nicht möglich war, ihren Terminkalender zu verändern. Alle Sichten sind aktualisiert.
Fehlersituationen	Benutzer hat für *keinen* der Teilnehmer die Berechtigung, den Termin zu erfassen, d.h. im Kalender des Teilnehmers einzutragen.
Nachzustand im Fehlerfall	Termin konnte nicht erfasst werden. Terminkalender der Teilnehmer und alle Sichten darauf sind im Status quo ante.
Akteure	Benutzer (primärer Akteur), E-Mail-System (sekundärer Akteur), Faxsystem (sekundärer Akteur).

Anwendungsfall	Termin erfassen
Standardablauf	1. Benutzer meldet sich an. 2. Eckdaten des neuen Termins (Zeit, Ort, voraussichtliche Dauer etc.) werden erfasst. 3. Alle Teilnehmer werden dem Termin zugeordnet. 4. Benutzer ist berechtigt, für alle Teilnehmer den Termin zu erfassen. 5. Termin führt bei keinem Teilnehmer zu Kollisionen und wird erzeugt. 6. Alle Teilnehmer am Termin werden gemäß ihrer bevorzugten Kommunikationsart verständigt. 7. Alle geöffneten Kalender von Teilnehmern werden aktualisiert. 8. Die entsprechenden Notifikationsanforderungen je Teilnehmer (z.B. 1h vor Terminbeginn ein akustisches Signal) werden registriert.
Alternativabläufe	4′. Benutzer ist für mindestens einen Teilnehmer *nicht* berechtigt, den Termin zu erfassen.
#1	5′. Analog zu 5. 6′. Analog zu 6, wobei jene Teilnehmer, deren Kalender nicht änderbar waren, zusätzlich davon informiert werden, dass es mangels Berechtigung nicht möglich war, ihren Terminkalender zu verändern. (Diese Benutzer haben also in der Realität einen Termin, der in ihrem Kalender nicht aufscheint.) 7′. Alle geöffneten Kalender von jenen Teilnehmern werden aktualisiert, für die eine Berechtigung zur Änderung des Kalenders vorliegt.
#2	5″. Termin führt bei mindestens einem Teilnehmer zu Kollisionen und es wird daher der Anwendungsfall *Terminvorschlag finden* ausgeführt. 5a″. Ein passender Ersatztermin wird gefunden und die Termin-Eckdaten werden entsprechend geändert.
#3	5‴. Analog zu 5″. 5a‴. Es wird kein passender Ersatztermin gefunden und der ursprüngliche Termin wird trotz Kollision eingetragen. 6‴. Analog zu 6, wobei jene Teilnehmer, für die sich eine Kollision ergibt, darauf gesondert hingewiesen werden.
Trigger	–

Name
Kurzbeschreibung

Jeder Anwendungsfall wird eindeutig durch seinen *Namen* im System identifiziert. Die *Kurzbeschreibung* soll die wesentlichen Aufgaben des Anwendungsfalls zusammenfassen.

Vorbedingung

In der *Vorbedingung* wird der Systemzustand spezifiziert, der eine notwendige Voraussetzung für ein erfolgreiches Ausführen des Anwendungsfalls darstellt. Ist die Vorbedingung nicht erfüllt, so kann der Anwendungsfall in der Regel nicht ausgeführt werden. In unserem Beispiel muss der Benutzer dem Kalendersystem bekannt sein, anderenfalls hat er keine Autorisierung zur Benutzung des Systems.

Nachbedingung

Analog dazu wird in der *Nachbedingung* der Systemzustand nach dem erfolgreichen Ausführen des Anwendungsfalls beschrieben. Im Wesentlichen wird in diesem Abschnitt die Erledigung der Aufgaben aus der Kurzbeschreibung postuliert.

Fehlersituationen

Der nächste Abschnitt beschreibt alle konzeptionellen *Fehlersituationen*, die auftreten und zu einem im Sinne der geforderten Nachbedingung nicht erfolgreichen Beenden des Anwendungsfalls führen können. Mit »konzeptionell« soll angedeutet werden, dass nur problembereichsrelevante Fehlersituationen beschrieben werden, aber keine technischen – z.B. wird der Systemabsturz aufgrund eines Hardwarefehlers in dieser Aufzählung nicht erwähnt. Zurückkommend auf unser Beispiel können wir festhalten, dass *Termin erfassen* nicht ausgeführt werden kann, wenn der Benutzer für keinen der Teilnehmer die Berechtigung zum Eintragen im Kalender des Teilnehmers hat.

Nachzustand im Fehlerfall

Der Systemzustand nach Auftreten von Fehlersituationen wird im Abschnitt *Nachzustand im Fehlerfall* charakterisiert. In unserem Fall bedeutet dies, dass der Termin nicht erfasst wurde und sich das System im Zustand vor Ausführung des Anwendungsfalls befindet.

Akteur

Im nächsten Abschnitt werden alle *Akteure* aufgelistet, die mit diesem Anwendungsfall kommunizieren.

Trigger

Kann der Anwendungsfall durch ein oder mehrere spezielle Ereignisse ausgelöst werden (z.B. Zeitereignis, Auftreten einer Ausnahmesituation in einem anderen Anwendungsfall), so wird dies im nächsten Abschnitt über *Trigger* beschrieben. In unserem Fall gibt es dazu keine speziellen Angaben, weil *Termin erfassen* bei Bedarf immer über einen entsprechenden Menüeintrag aufgerufen werden kann und nicht durch einen speziellen Trigger ausgelöst wird.

Standardablauf

Der nächste Abschnitt beschreibt das »Fleisch« des Anwendungsfalls, nämlich den *Standardablauf*. Die einzelnen Schritte, die im Normalfall durchlaufen werden, sind hier der Reihe nach aufgelistet. Ein Schritt kann selbst wieder einen weiteren Anwendungsfall (im Sinne eines Prozeduraufrufs) aktivieren.

Alternativabläufe

Abweichungen von diesem Standardablauf werden unter *Alternativabläufe* zusammengefasst. Die Nummer im Alternativablauf bezieht sich auf die jeweilige Nummer im Standardablauf, der zugeordnete Schritt redefi-

niert den entsprechenden Schritt im Standardablauf. Im Beispiel sind zwei »Abweichungen von der Norm« definiert. Erstens, falls der Benutzer für einen Teilnehmer keine Schreibberechtigung für dessen Kalender hat, so wird der Teilnehmer entsprechend informiert. Und zweitens, falls dieser neue Termin zu Terminkollisionen bei einem Teilnehmer führt, so kann vom System ein neuer Terminvorschlag automatisch erstellt werden. Da *Terminvorschlag finden* bereits als Anwendungsfall bei der Analyse der Zeitwortphrasen identifiziert wurde (Tab. 4–1), wird er in diesem Alternativzweig aufgerufen.

Die beschriebene Dokumentvorlage ist natürlich nur als Vorschlag zu sehen und kann abhängig von den jeweiligen Projektanforderungen geändert bzw. erweitert werden. In [Cock97] wird auch der Vorschlag gemacht, einen zusätzlichen Abschnitt *Weitere Informationen* hinzuzufügen, der z.B. nichtfunktionale Anforderungen, eine Liste der rufenden bzw. gerufenen Anwendungsfälle und offene Problempunkte enthält. Eine Entscheidung darüber, welche Information mit welchem Detaillierungsgrad in der Anwendungsfallbeschreibung aufzunehmen ist, ist sicher auch Aufgabe des jeweiligen Projektmanagements.

Aktivitätsdiagramm

Bei komplexen, variantenreichen Anwendungsfällen können die zuvor empfohlenen verbalen Ablaufbeschreibungen zu Mehrdeutigkeiten und Missverständnissen führen. Um dem vorzubeugen, empfiehlt es sich in einem solchen Fall, die Anwendungsfallbeschreibung durch ein Aktivitätsdiagramm zu formalisieren. Je nach Komplexität kann entweder für jede Ablaufvariante ein eigenes Aktivitätsdiagramm angegeben werden, oder es können auch einzelne Varianten durch Entscheidungsknoten innerhalb eines einzigen Diagramms dargestellt werden.

Aktivitätsdiagramm
S. 160

Für den im vorhergehenden Unterabschnitt erläuterten Anwendungsfall *Termin erfassen* wurde bereits in Kapitel 2 (Abb. 2–100, S. 171) ein Aktivitätsdiagramm angegeben, das der Ablaufbeschreibung in Tabelle 4–2 einigermaßen entspricht. Die Problematik der Berechtigung, einen Teilnehmer einem Termin zuzuordnen (und damit implizit dessen Kalender zu ändern), wurde dort jedoch außer Acht gelassen.

Abbildung 4–6 zeigt eine entsprechend erweiterte Darstellung. Der Ablauf beginnt mit dem Anmelden des Benutzers (Schritt 1 in Tab. 4–2). Zur Prüfung der eingegebenen Identifikation wird das entsprechende Benutzerobjekt im System herangezogen. Bei Übereinstimmung wird ein leeres Terminobjekt angelegt, dem anschließend die so genannten »Eckdaten« und die Teilnehmer zugeordnet werden. Während die entsprechenden Schritte 2 und 3 in Tabelle 4–2 sequenziell erfolgen, ist aufgrund der in Abbildung 4–6 spezifizierten Nebenläufigkeit dabei *keine* besondere Reihenfolge einzuhalten.

Abb. 4–6 *(Seite 197) Aktivitätsdiagramm zum Anwendungsfall »Termin erfassen«*