Das Use-Case-Konzept

Christoph Weiß
RWG GmbH
E-Mail: christoph_weiss@compuserve.com

Ein kurzer Abriß

Ein Use-Case ist eine Art Klasse für Anwendungsfälle von Softwaresystemen. Zum Beispiel wäre in einer Software für die Deutsche Bahn AG *»einen Platz reservieren«* ein denkbarer Use-Case. Ein konkreter Anwendungsfall davon könnte *»im 18.07 Uhr-Zug am 31.5.98 von Stuttgart nach Nürnberg einen Platz reservieren«* sein. Use-Cases eignen sich gut für die Beschreibung von Bedienabläufen bei interaktiven Systemen und haben ihre Schwächen bei der Beschreibung von parallelen bzw. nicht interaktiven Abläufen.

In Use-Case-Beschreibungen wird die Funktionalität einer Software aus beschrieben, wobei ein Anwender durchaus auch eine andere Software oder eine Hardware sein kann. Die Use-Case-Beschreibungen bilden dann den Kern eines Use-Case-Modells. Daher kann ein Use-Case-Modell im Rahmen einer ANSI/IEEE-konformen Spezifikation (vgl. [1]) die Rolle eines funktionalen Modells übernehmen.

Im Gegensatz zu den Funktionsbeschreibungen eines herkömmlichen funktionalen Modells werden Use-Cases konsequent aus Anwendersicht heraus beschrieben, wobei jeweils eine Kette von Aktionen beschrieben wird, die ein Anwender ausführen muß, um ein bestimmtes Ziel (z.B. *»einen Platz reservieren«*) zu erreichen. Dabei bleibt der Gesamtzusammenhang der einzelnen Funktionen einer Software erhalten. Zwar kommt unter den Use-Case-Beschreibungen oft eine funktionale Redundanz zustande, diese läßt sich aber noch im Use-Case-Modell selbst wegmodellieren, ohne daß dabei der jeweilige Gesamtzusammenhang verloren geht.

Use-Case-Beschreibungen sind sowohl für einen Auftraggeber als auch für einen Entwickler verständlich. Aus der Sicht eines Auftraggebers dienen Use-Case-Beschreibungen als Beschreibungen dessen, was er von einer Software erwartet. Aus der Sicht eines Entwicklers dienen Use-Case-Beschreibungen einerseits als Ausgangsbasis für den Entwurf einer Software und andererseits als Ausgangsbasis für das Testmodell für eine Software. Beides kann parallel entwickelt werden.

1 Einführung

Die Idee der Use-Cases stammt ursprünglich von Ivar Jacobson (vgl. [3], [6]), der inzwischen mit Booch und Rumbaugh an einer gemeinsamen Methode bastelt. Das erste Ergebnis dieser Zusammenarbeit ist die Unified Modeling Language (UML), in der auch die Notation von Use-Cases berücksichtigt wurde (vgl. [2]). Doch Booch und Rumbaugh hatten Use-Cases bereits vor ihrer Zusammenarbeit in ihre jeweilige Methode übernommen und auch Wirfs-Brock arbeitet seit geraumer Zeit mit Use-Cases (vgl. [4]). Es scheint sich die Verwendung von Use-Cases immer mehr zu einem Standard zu entwickeln.

Meist werden Use-Cases lediglich anhand von Beispielen erklärt. Ich will deshalb hier versuchen, den Use-Case-Begriff etwas genauer zu fassen. Unter einem **Use-Case** wird eine Reihe von Bedienungsabläufen zusammengefaßt, wobei mit jedem dieser Abläufe ein konkretes Ziel verfolgt wird und ein Ablauf entweder

- der normale Ablauf eines Anwendungsfalls (pro Use-Case genau einer) oder
- eine Variante des normalen Ablaufs oder
- ein zu dem normalen Ablauf verwandter Ablauf oder
- eine Ausnahmebehandlung im Fehlerfall in einer der anderen Abläufe des Use-Case

sein kann.

Zum Beispiel könnten unter dem Use-Case *»einen Platz reservieren«* folgende Abläufe zusammengefaßt werden:

- Normaler Ablauf:
  - ◊ eine Platzreservierung vornehmen
- Varianten:
  - ◊ eine Platzreservierung ändern
  - ◊ keine Reservierung mehr möglich*
- Verwandter Ablauf:
  - ◊ eine Platzreservierung löschen
- Ausnahmebehandlung:
  - ◊ das Datum wurde im falschen Format eingegeben

Als normaler Ablauf wird nicht unbedingt der häufigste Ablauf gewählt, sondern der aussagekräftigste, d.h. der Ablauf, der am deutlichsten macht, um was es eigentlich geht.

Im Kontext von Use-Cases wird eine Person, eine Software oder eine Hardware, die ein System auf eine bestimmte Weise benutzt, als **Akteur** (engl. actor) bezeichnet. Dabei ist der Begriff des Akteurs allerdings nicht an die besagte Person, Software oder Hardware gebunden, sondern an die Rolle, in welcher diese das

* Wenn keine Reservierung mehr möglich ist, so liegt ein legitimer Fall im normalen Ablauf und kein Bedienungsfehler vor. Deshalb fällt dieser Fall unter die Varianten und nicht unter die Ausnahmebehandlungen.

System benutzt. Ein Akteur läßt sich daher auch als Klasse für Anwender betrachten.

Ein beliebtes Beispiel ist in diesem Zusammenhang ein Bankangestellter, der gleichzeitig auch Kunde bei seiner Bank ist. Dieser kann das System der Bank entweder in seiner Rolle als Bankangestellter oder in seiner Rolle als Kunde benutzen und verkörpert dabei entweder den Akteur »Bankangestellter« oder den Akteur »Kunde«.

Use-Cases werden in Form von **Use-Case-Beschreibungen** festgehalten, in denen jeweils die Akteure eines Use-Case vermerkt und die einzelnen Abläufe eines Use-Case beschrieben werden. Dabei besteht die Beschreibung eines Ablaufs aus einer Folge von Benutzereingaben und den jeweiligen Reaktionen des Systems darauf. Eine Use-Case-Beschreibung wird immer aus der Anwendersicht heraus erstellt.

Ein **Use-Case-Modell** besteht aus Use-Cases und Akteuren. Jeder Use-Case wird zusammen mit seinen Akteuren in Form einer Use-Case-Beschreibung festgehalten. Ein Use-Case-Modell enthält Informationen darüber,

- was ein System können soll (Use-Case),
- wer ein System wie benutzen kann (Akteure eines Use-Case),
- wie ein System jeweils auf bestimmte Benutzereingaben reagieren soll (Beschreibungen der Abläufe).

Bei der Beschreibung des Use-Case-Konzepts wird immer großzügig darüber hinweggegangen, wie Regeln aus dem jeweiligen Anwendungsbereich einer Software gehandhabt werden. Wenn überhaupt, so tauchen Regeln implizit in den Ablaufbeschreibungen auf. Nach meiner bisherigen Erfahrung hat sich eine explizite Beschreibung von Regeln im Use-Case-Modell als sinnvoll erwiesen. In den Ablaufbeschreibungen wird dann an den Stellen, an denen Regeln zum Einsatz kommen, jeweils auf diese Regelbeschreibungen verwiesen und beschrieben, wie diese dort angewendet werden.

2 Die Rolle der Use-Cases im Verlauf einer Entwicklung

In Abbildung 2.1 wird ein Beispiel für die Umsetzung des Use-Case-Konzepts dargestellt. Darin übernimmt in einer ANSI/IEEE-konformen Spezifikation das Use-Case-Modell die Rolle eines funktionalen Modells (vgl. [1]). Gleichzeitig dient das Use-Case-Modell als Vertragsgrundlage für den Auftraggeber und den Entwickler.

Für den Auftraggeber bildet das Use-Case-Modell eine Beschreibung dessen, was er von einer Software erwartet. Für den Entwickler bildet das Use-Case-Modell den Ausgangspunkt sowohl für den Entwurf der Software als auch für das Test-Modell für die Software.

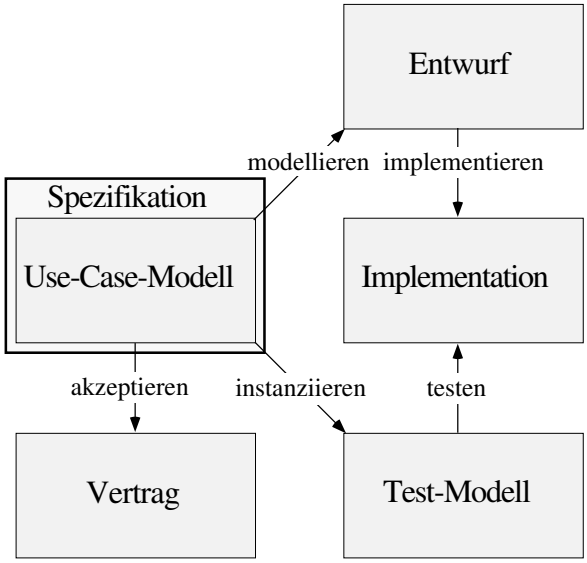


Abb. 2.1: Umsetzung des Use-Case-Konzepts

Als erster Entwurf werden zunächst die einzelnen Use-Cases des Use-Case-Modells modelliert. Dieser »naive« Entwurf wird anschließend im Hinblick auf bessere Wartbarkeit, höhere Flexibilität, etc. umstrukturiert. Dies kann zum Beispiel mit Hilfe von Entwurfsmustern geschehen.

Parallel dazu kann auf der Basis der Use-Cases mit der Entwicklung eines Test-Modells begonnen werden. Dazu werden die einzelnen Use-Case-Abläufe instanziiert, d.h. konkrete Anwendungsfälle konstruiert.

3 Vom Use-Case zum Entwurf

Ich möchte nun an einem Beispiel aus der Praxis demonstrieren, wie eine Use-Case-Beschreibung aufgebaut sein kann und wie man von einer Use-Case-Beschreibung zu einem Entwurf kommt. Der äußere Rahmen des Projekts war zunächst meine Diplomarbeit bei Prof. Ludewig an der Universität Stuttgart. Dabei ging es um die Entwicklung einer Klassenbibliothek, die Programmen eine graphische Darstellung von Noten am Bildschirm ermöglicht.

Herausgekommen ist eine objektorientierte Spezifikation und ein implementierungsunabhängiger Entwurf. Kernstück der ANSI/IEEE-konformen Spezifikation ist ein Use-Case-Modell, das aus 8 Use-Cases, 1 Akteur und 40 Regeln besteht. Der Entwurf der Klassenbibliothek besteht aus über 90 Klassen.

Das Projekt wird mittlerweile an der Universität Stuttgart fortgesetzt, wo der Entwurf im Rahmen von Sorftwarepraktika implementiert wird. Dies erfolgt in enger Zusammenarbeit mit der Musikhochschule Stuttgart, da dort ein konkreter Bedarf an dieser Software besteht.

3.1 Aufbau der Beschreibung eines Use-Case

In diesem Projekt, wie auch in anderen Projekten, hat sich der folgende Aufbau für die Beschreibung eines Use-Case bewährt:

Use-Case:

Eindeutige Bezeichnung des Use-Case.

Notwendigkeit:

Ein Use-Case besitzt einen der folgenden drei Grade an Notwendigkeit:

- *verbindlich*: Der Use-Case muß verwirklicht werden.
- *wünschenswert*: Der Use-Case würde in den eigentlichen Anwendungsbereich der Software passen und damit deren Funktionalität konsequent erweitern. Die Software kann aber auch ohne dessen Verwirklichung akzeptiert werden.
- *optional*: Der Use-Case paßt nicht in den eigentlichen Anwendungsbereich der Software, würde diesen aber evtl. sinnvoll ergänzen. Ein Use-Case dieser Art kann am ehesten weggelassen werden.

verwendet:

Auflistung der Use-Cases, die von diesem Use-Case verwendet werden. Dabei wird in der Beschreibung des Use-Case für jeden verwendeten Use-Case aufgeführt, wann dieser verwendete Use-Case zur Anwendung kommt.

ergänzt:

Auflistung der Use-Cases, die dieser Use-Case ergänzt. Dabei wird in der Beschreibung des ergänzenden Use-Case für jeden ergänzten Use-Case aufgeführt, wann dieser ergänzende Use-Case zur Anwendung kommt und von wo bis wo dieser in den jeweiligen ergänzten Use-Case eingefügt wird.

Zusammenfassung:

Zusammenfassung des Use-Case.

Akteure:

Akteure, die den Use-Case durchführen können.

Vorbedingungen:

Der Use-Case wird nur dann durchgeführt, wenn diese Bedingungen erfüllt sind.

Nachbedingungen:

Bedingungen, die nach erfolgreicher Durchführung des Use-Case garantiert erfüllt sind.

im Fehlerfall:

Fehlerbehandlung, d.h. Beschreibung dessen, was passiert, wenn während der Durchführung des Use-Case ein Fehler auftritt.

änderungsanfällig:

Gibt darüber Auskunft, ob im Lauf der Zeit Änderungen am Use-Case eher wahrscheinlich sind oder ob er eher stabil bleibt. Wenn Änderungen zu erwarten sind, wird angegeben, für welche Teile des Use-Case diese erwartet werden und wie diese aussehen könnten.

Regeln:

Auflistung der Regeln aus dem Anwendungsbereich, die, zusammen mit deren Ausnahmen, in dem Use-Case zur Anwendung kommen.

normaler Ablauf:

Beschreibung des Ablaufs, der für das Verständnis des Use-Case am besten geeignet ist.

Varianten:

Beschreibung von Varianten des normalen Ablaufs, sowie von Abläufen, die zu dem normalen Ablauf verwandt sind.

3.2 Auszug aus einer Use-Case-Beschreibung

Um einen besseren Überblick zu ermöglichen und die Aufmerksamkeit nicht vom Wesentlichen abzulenken, folgt hier nur ein vereinfachter Auszug aus einer Use-Case-Beschreibung. In diesem Use-Case geht es eigentlich um die Erstellung eines Liniensystems. Gezeigt wird eine Variante davon, nämlich die Änderung an einem Liniensystem. Bestimmte Änderungen an einem Liniensystem können allerdings noch wesentlich komplexere Folgen haben, als hier gezeigt wird.

Das Ganze spielt sich innerhalb eines Model-View-Controller-Kontexts ab, wobei das Modell in diesem Fall ein Liniensystem ist, das mittels einer oder mehrerer Views, den Layouts, dargestellt wird. Änderungen am Modell werden jeweils von dem Controller, dem Layout-Manager, an davon betroffene Views weitergeleitet. Die Beschreibung der vereinfachten Änderungsvariante sieht dann folgendermaßen aus:

Varianten:

Änderungen an einem Liniensystem vornehmen

Folgende Änderungen sind möglich:

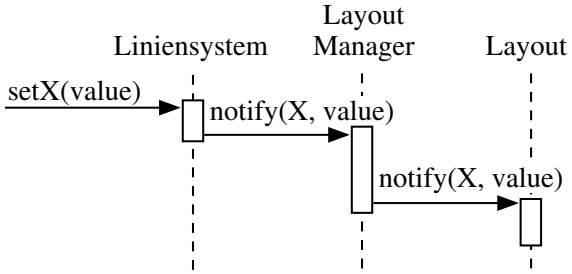
- *Tonart ändern*
- *Tonartwechsel ändern*
- *Tonartwechsel hinzufügen*
- *Tonartwechsel entfernen*

Dasselbe gilt analog für die Taktarten und den Notenschlüssel.
Der Layout-Manager wird über die jeweilige Änderung informiert. Dieser wiederum informiert betroffene Layouts über die jeweilige Änderung.

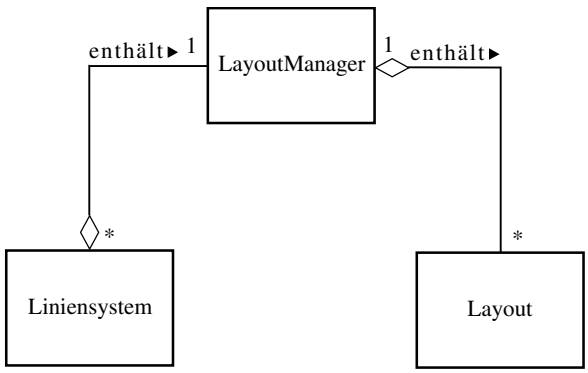
3.3 Realisierung eines Entwurfs

Informationen, die in einer Use-Case-Beschreibung enthalten sind, werden im Entwurf teilweise in Graphiken und teilweise in textuellen Beschreibungen festgehalten. So werden beispielsweise die Attribute, die geändert werden können, in der Beschreibung der Klasse *Liniensystem* in Form der Methoden dieser Klasse festgehalten. Die Beschreibung der Klassen wird hier nicht angegeben, da dies den Rahmen sprengen würde, aber ich denke, das Beispiel läßt sich auch so nachvollziehen.
Der eigentliche Ablauf eines Use-Case wird in Form eines Diagramms für dynamische Abläufe, wie zum Beispiel einem Sequenzdiagramm* (vgl. [2]), dargestellt.

Bei einem Sequenzdiagramm werden die beteiligten Instanzen jeweils durch eine vertikale gestrichelte Linie wiedergegeben. Aktivitätsphasen einzelner Instanzen werden durch vertikale Balken auf den gestrichelten Linien dargestellt. Nachrichten werden durch horizontale Pfeile symbolisiert. So ist die x-Achse die Nachrichtenachse und die y-Achse ist die Zeitachse. Da die Zeitachse nicht maßstabsgetreu ist, kann an ihr nur die Ausführungsreihenfolge abgelesen werden. Auf der linken Seite eines Sequenzdiagramms steht immer eine Beschreibung des Ablaufs. Diese wird hier aus Platzgründen nach dem Sequenzdiagramm angegeben. Das Sequenzdiagramm zu der beschriebenen Use-Case-Variante sieht dann folgendermaßen aus:



Wenn an einem Liniensystem der Wert von dessen Attribut X geändert wird, dann informiert das Liniensystem den LayoutManager über die Änderung.
Anschließend informiert der LayoutManager die betroffenen Layouts über die Änderung.



* Sequenzdiagramme hießen bisher Interaktionsdiagramme. In der UML wurde dies geändert (vgl. [2]).

Unter dem Sequenzdiagramm ist das dazugehörige statische Diagramm zu sehen, in dem die Beziehungen zwischen den einzelnen Klassen dargestellt werden. Für beide Diagramme wurde die UML (vgl. [2]) verwendet.

4 Modellierungsmöglichkeiten mit Use-Cases

Anwendungsfälle mit Hilfe von Use-Cases zu beschreiben, bringt oft eine Redundanz der beschriebenen Funktionalität einer Software mit sich. Eine Möglichkeit, Redundanzen wegzumodellieren, besteht darin, Teilabläufe, die mehrere Use-Cases gemeinsam haben, jeweils in Form eines eigenen Use-Case zu modellieren, auf den dann die Use-Cases, die den ausgelagerten Teilablauf benötigen, zurückgreifen.

Die Abbildung 4.1 zeigt ein Beispiel, wie es für ein Notensatzprogramm denkbar wäre. In einem Notensatzprogramm können meist außer den Noten auch noch ein Liedtext und Akkordbezeichnungen dargestellt werden. Dieser Tatsache werden die beiden Use-Cases *»Liedtext darstellen«* und *»Akkordbezeichnungen«* gerecht.

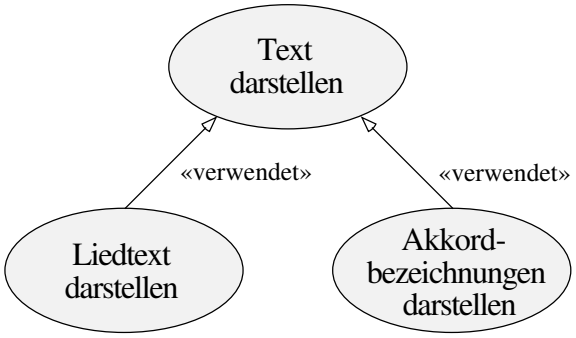


Abb. 4.1: Ein abstrakter Use-Case

Abstrahiert man deren jeweilige Semantik, so bleibt das grundsätzliche Problem des Darstellens eines Textes übrig. Deshalb verwenden die beiden erstgenannten Use-Cases einen dritten Use-Case *»Text darstellen«*, der diese Funktionalität beschreibt. Nun hat in einem Notensatzprogramm üblicherweise jede Art der Textdarstellung eine besondere Semantik, die jeweils im Kontext des darzustellenden Liedes zu sehen ist. Daher wird der Use-Case *»Text darstellen«*, nie direkt zum Einsatz kommen. Wenn ein Use-Case ausschließlich von anderen Use-Cases verwendet wird, so handelt es sich bei diesem Use-Case um einen abstrakten Use-Case.

Ein **abstrakter Use-Case** ist ein Use-Case, der selbst keine Akteure besitzt. Ein abstrakter Use-Case kann bewußt abstrakt modelliert sein, d.h. er beschreibt einen oder mehrere Vorgänge, die nicht von einem Anwender direkt ausgeführt werden können, da deren direkte Ausführung für den Anwender keinen Sinn ergibt.

Ein abstrakter Use-Case kann aber auch unbewußt abstrakt modelliert sein, wenn er im Moment noch keine

Akteure besitzt, es aber nicht ausgeschlossen ist, daß er einmal Akteure haben wird. Bekommt ein abstrakter Use-Case dann Akteure, so geht dessen Eigenschaft »abstrakt« verloren.

Eine **verwendet-Beziehung** wird eingesetzt, wenn sich ein Kernablauf aus mehreren Teilabläufen zusammensetzt. Ein verwendeter Use-Case beschreibt also einen Ablauf, der zu dem Ablauf eines anderen Use-Case hinzugefügt wird, um damit einen Kernablauf zusammenzubauen. Die Abbildung 4.2 zeigt ein Beispiel für einen verwendeten Use-Case, der nicht abstrakt ist. Darin verwendet der Use-Case *»Fußnote darstellen«* den Use-Case *»Text darstellen«*. Der Kontext des Beispiels ist diesmal aber kein Notensatzprogramm, sondern ein Textverarbeitungsprogramm. In diesem neuen Kontext ist eine direkte Verwendung des Use-Case *»Text darstellen«*. sinnvoll.

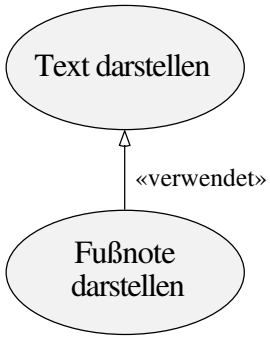


Abb. 4.2: Eine verwendet-Beziehung

In einem verwendenden Use-Case wird beschrieben, welche anderen Use-Cases wo und wie verwendet werden. Dagegen wird in einem verwendeten Use-Case nicht beschrieben, ob und von welchen Use-Cases er verwendet wird.

Neben der verwendet-Beziehung gibt es für Use-Cases noch einen zweiten Beziehungstyp, die ergänzt-Beziehung. Eine **ergänzt-Beziehung** wird eingesetzt, wenn ein bereits vollständiger Kernablauf in bestimmten Fällen noch zusätzlich erweitert wird. Ein ergänzender Use-Case beschreibt also einen Ablauf, um den der Kernablauf eines anderen Use-Case in bestimmten Fällen erweitert wird.

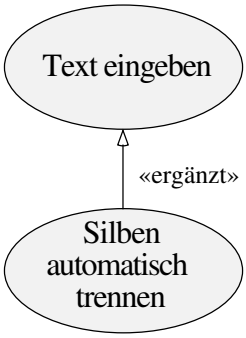


Abb. 4.3: Eine ergänzt-Beziehung

Die Abbildung 4.3 zeigt ein Beispiel für einen ergänzenden Use-Case. Der Kontext des Beispiels ist wieder das Textverarbeitungsprogramm. In dem Textverarbeitungsprogramm wird die Texteingabe von einer automatischen Silbentrennung unterstützt.

In einem ergänzenden Use-Case wird immer beschrieben, in welchen Fällen er welchen Use-Cases wie ergänzt. Dagegen wird in einem ergänzten Use-Case nie beschrieben, ob und von welchen Use-Cases er ergänzt wird. Daher kann im Beispiel die Texteingabe um die automatische Silbentrennung erweitert werden bzw. kann die automatische Silbentrennung bei Bedarf wieder entfernt werden, ohne daß die Texteingabe dazu geändert werden muß.

Nun stellt sich natürlich die Frage nach der Existenzberechtigung der ergänzt-Beziehung. Denn auf den ersten Blick läßt sich eigentlich alles mittels der verwendet-Beziehung modellieren. Aber dieser Schein trügt. So kann das letzte Beispiel nicht ohne semantischen Verlust mittels einer verwendet-Beziehung modelliert werden. Es geht dabei entweder die Semantik verloren, daß die Texteingabe der Kernablauf ist oder es geht die Semantik verloren, daß der Kernablauf beliebig erweitert werden kann, ohne daß an diesem etwas geändert werden muß.

Beide Beziehungstypen ähneln der herkömmlichen Vererbungsbeziehung. Da es sich jeweils um die »Vererbung« von Abläufen handelt, unterscheiden sich diese dann von einer herkömmlichen Vererbungsbeziehung, wenn Abläufe nicht mehr direkt »ererbt« werden.

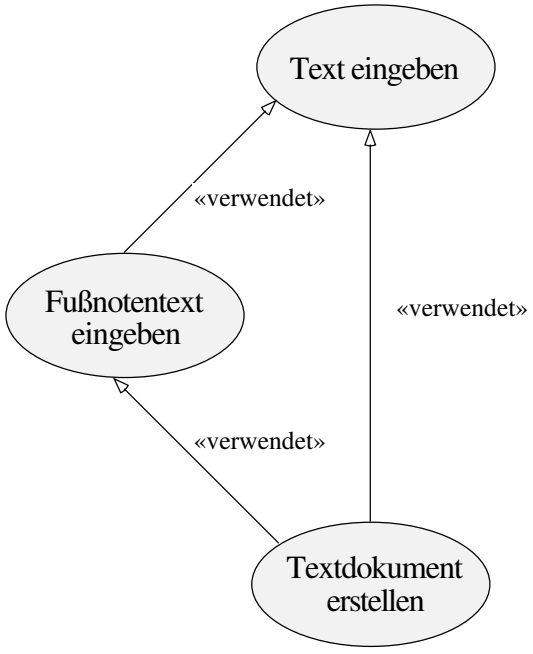


Abb. 4.4: Unterschied zur Vererbung

Die Abbildung 4.4 zeigt hierzu ein Beispiel. Beim Erstellen eines Textdokuments sollen Fußnoten eingegeben werden können. Daher verwendet der Use-

Case *»Textdokument erstellen«* den Use-Case *»Fußnotentext eingeben«*. Da das Eingeben von Fußnotentext nur eine besondere Form der Texteingabe ist, verwendet der Use-Case *»Fußnotentext eingeben«* den Use-Case *»Text eingeben«*.

Natürlich soll beim Erstellen eines Textdokuments auch normaler Text eingegeben werden können. Daher verwendet der Use-Case *»Textdokument erstellen«* ebenfalls den Use-Case *»Text eingeben«*. Bei einer echten Vererbungsbeziehung wäre diese verwendet-Beziehung nicht nötig, denn die Fähigkeit zur Texteingabe würde dann über *»Fußnotentext eingeben«* indirekt ererbt werden.

Es läßt sich unter diesem Gesichtspunkt auch darüber diskutieren, inwieweit Use-Cases nun wirklich Klassen für Anwendungsfälle sind und ob es sich dabei nicht eher um Schablonen für Anwendungsfälle handelt. Da diese Diskussion den Rahmen dieses Beitrags sprengen würde, gebe ich diesen Punkt hier lediglich zu bedenken und halte mich auch weiterhin an Jacobson.

5 Probleme und Risiken bei der Arbeit mit Use-Cases

Die Arbeit mit Use-Cases bietet nicht nur Vorteile, sondern bringt auch Probleme mit sich. Use-Cases werden aus Anwendersicht heraus beschrieben. Daher ist bei der Abschätzung des Arbeitsaufwands für ein Projekt und damit auch für die zeitliche Planung eines Projekts beim Umgang mit Use-Cases Vorsicht geboten. Denn Use-Cases sagen zwar etwas über den Funktionsumfang eines Systems aus, aber je größer der Anteil an internen Berechnungen innerhalb einzelner Use-Cases ist, desto schwieriger wird es, anhand der Use-Cases den Arbeitsaufwand abzuschätzen, der für die Realisierung des Funktionsumfangs benötigt wird. Wenn beispielsweise in einem Projekt mit zehn Use-Cases genau ein Use-Case vorkommt, der sehr aufwendige Berechnungen enthält, so kann dessen Realisierung im Entwurf durchaus mehr als 50% des gesamten Entwurfs ausmachen. In der Rechenzentrale Württembergischer Genossenschaften (RWG) begegnen wir dem Problem damit, daß es auch noch einen technischen Beschreibungstyp gibt. Falls nötig werden ergänzende technische Beschreibungen angefertigt, die dann nicht für den Anwender bestimmt sind.

Use-Cases sind Klassen für Anwendungsfälle. Daher können keine Konflikte zwischen Instanzen davon, also konkreten Abläufen von Use-Cases, modelliert werden. Dieses Problem läßt sich durch die Verwendung von Szenarien und Regeln umgehen. In den Szenarien werden dann konkret solche Situationen unter Anwendung entsprechender Regeln abgehandelt.

Use-Cases sind atomar und seriell. Daher können damit keine parallelen Situationen modelliert werden. Auch dieses Problem läßt sich durch Verwendung konkreter Szenarien und entsprechender Regeln umgehen.

6 Fazit

Der große Vorteil bei der Arbeit mit Use-Cases liegt zweifellos in deren Verständlichkeit für den Auftraggeber und für den Entwickler. Darüber hinaus helfen Use-Cases aber auch noch dem Entwickler, den funktionalen Gesamtzusammenhang eines Systems zu überblicken und bilden den »roten Faden« durch den Verlauf einer Entwicklung. Aber auch Use-Cases haben ihre Tücken und sind nicht universell einsetzbar. Sie haben ihre Schwächen bei stark automatisierten und bei parallelen Systemen.

Zum Schluß noch zwei kleine Anregungen. Es kann auch sehr nützlich sein, mit Ist- und Soll-Use-Cases zu arbeiten. Dabei wird eine Bestandsaufnahme in Form von Ist-Use-Cases gemacht, in dem Abläufe festgehalten werden, wie sie derzeitig vonstatten gehen. Dann wird in Form von Soll-Use-Cases festgehalten, wie die aktuellen Abläufe in Zukunft aussehen sollen. Diese Vorgehensweise kann beispielsweise beim Reengineering angewendet werden. Aber Use-Cases eignen sich nicht nur zur Beschreibung rechnerunterstützter Abläufe, sondern generell zur Beschreibung von Abläufen. So kann in einem Ist-Use-Case ein Ablauf festgehalten werden, wie er bisher stattfindet und in dem Soll-Use-Case wie dieser Ablauf später mit Rechnerunterstützung aussehen soll. Mit Use-Cases können auch bestehende Abläufe einfach nur optimiert werden, unabhängig davon ob diese rechnerunterstützt sind oder nicht. Beispielsweise lassen sich damit auch verwaltungstechnische Abläufe optimieren. Es können sogar ganze Firmen auf dieser Basis restrukturiert bzw. Arbeitsabläufe in Firmen optimiert werden (vgl. [6]).

Literatur

[1] ANSI/IEEE (1984): *An American National Standard: IEEE Guide to Software Requirements Specifications*. **ANSI/IEEE Std 830-1984**.

[2] Booch G., Rumbaugh J., Jacobson I. (1997): The ***Unified Modeling Language***. Documentation Set. Version 1.1 (Erhältlich bei Rational via Internet: http://www.rational.com).

[3] Jacobson et al. (1993): ***Object-Oriented Software Engineering*** *– A Use Case Driven Aprroach.* Addison-Wesley, Reading, Massachusetts.

[4] Jacobson I., Christerson M. (1995): *Modeling With Use Cases. A growing consensus on use cases*. **Journal of Object-Oriented Programming**, **March-April**, S. 15-19.

[5] Jacobson et al. (1995): ***The Object Advantage***– *Business Process Reengineering with Object Technology.* Addison-Wesley, Reading, Massachusetts.

[6] Jacobson et al. (1997): ***Software Reuse***– *Architecture, Process and Organization for Business Success.* ACM Press, Addison Wesley Longman, New York, New York.