Use Cases

Als ein populäres Mittel um Anforderungen zu erfassen und Systeme zu beschreiben, werden *Use Cases* benutzt. Sie bilden die Basis für eine umfassendere objektorientierte Entwurfsmethode (Object-Oriented Software Engineering, *OOSE*), welche von Ivar Jacobson mitentwickelt wurde.

Slide 1

Hauptsächlich beschreiben *Use Cases* das Verhalten eines Systems in Form von Anwendungsfällen oder auch Transaktionen.

Modelle

Die Modelle, die im *OOSE* benutzt werden, decken alle Phasen des Entwicklungsprozesses ab:

Slide 2

Anforderungsmodell: Beschreibung der funktionalen Anforderungen aus Benutzersicht

Analysemodell: Logisches Modell des Systems, welches auf stabilen Anforderungen und einer idealen Umgebung aufbaut

Designmodell: Umwandlung des Analysemodells in ein implementierbares Design

Implementierungsmodell: Eigentlicher Programmcode

Testmodell: Entwicklung von Prüfplänen und Durchführung der Tests

Architektur

Unter Architektur im OOSE wird die Basis oder auch Bedeutung aller entwickelten Modelle verstanden. Sie ist das Resultat der Anwendung einer *Methode* auf ein System.

Slide 3

Im OOSE wird hier eine Parallele zur objektorientierten Entwicklung gezogen: die Architektur stellt die beschreibende Klasse aller erzeugbaren Modelle oder Systeme (Instanzen) dar.

Prozeß

Im OOSE wird der Entwicklungsprozeß losgelöst vom Projekt eher produktbezogen betrachtet. Als Einzelprozesse werden dabei der

Slide 4

- Analyse-
- Konstruktions-
- Komponentenentwicklungs- und
- Testprozeß

beschrieben. Diese Prozesse kommunizieren dabei während der Entwicklungsphase miteinander.

Prozesse und Modelle

Der Entwicklungsprozeß wird im OOSE als Vorgehen verstanden, bei dem Modelle ineinander überführt werden. Grundsätzlich gibt es zwei Möglichkeiten:

operational: ein Modell wird als Ganzes in ein anderes Modell überführt, welches dann auf Korrektheit geprüft wird

Slide 5

transformational: einzelne Transformationen werden auf einer Ebene der Einzelbestandteile eines Modells durchgeführt, wobei die Transformationen jeweils geprüft werden

Im OOSE wird hier vornehmlich operational gearbeitet. Dabei wird verstärkt auf Durchgängigkeit geachtet, so daß sich Objekte über mehrere Umwandlungen hinweg verfolgen lassen.

Analyse

Innerhalb des Analyseprozesses werden die Anforderungen zunächst in ein Anforderungsmodell und danach ein Analysemodell überführt. Die beiden Modelle haben dabei folgende Charakteristika:

Slide 6

- Beschreibung dessen, *was* das System tut (im Gegensatz zum Design, wo das *wie* im Vordergrund steht).
- Anwendungsorientierung, keine Berücksichtigung der späteren Implementierung
- Möglichkeit der Diskussion mit Endbenutzern durch die Anwendungsorientierung
- Unabhängigkeit bei Änderungen der Implementierungsumgebung

Das Anforderungsmodell

Das Anforderungsmodell soll die Funktionalität des Systems definieren. Dabei wird vom Standpunkt des Benutzers ausgegangen. Das Modell besteht aus

- dem Use-Case-Modell (Beschreibung der Funktionalität),

Slide 7

- Interface-Beschreibungen (Beschreibung der Interaktion mit dem System) und
- einem Modell des Problembereichs (Basis für alle Beteiligten)

Aktoren

Um die Anwendungsfälle zu beschreiben, müssen zuerst die Benutzer des Systems identifiziert werden. Dies geschieht mit Hilfe von *Aktoren*, welche bestimmte Rollen bei der Benutzung des Systems definieren. Aktoren können

- Endbenutzer oder

Slide 8

- andere Systembestandteile (Hard-/Software) sein.

Ein Aktor beschreibt aber nicht einen bestimmten Benutzer, sondern dient als Klassendefinition für Rollen.

Aktoren

Aktoren werden in zwei Gruppen unterteilt:

primäre Aktoren: bestimmen die Funktionalität des Systems und benutzen es als Hauptzielgruppe

Slide 9

sekundäre Aktoren: hängen von primären Aktoren ab, unterstützen diese in der Systembenutzung

Aktoren befinden sich immer außerhalb der Grenzen des zu erstellenden Systems

Use Cases

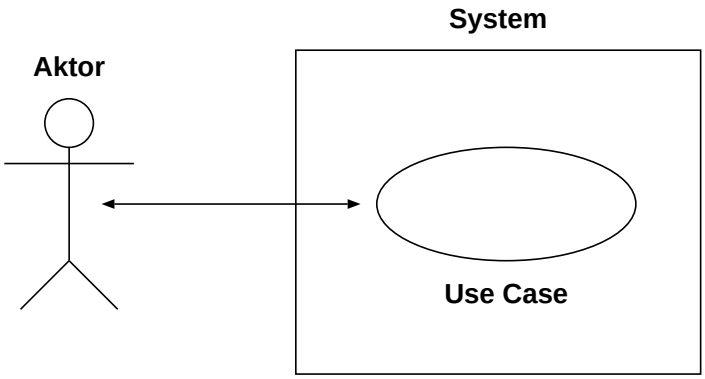
Ein Use Case beschreibt eine Interaktion eines Aktors mit dem System in natürlicher Sprache. Er beinhaltet alle Einzelereignisse, die bei der Interaktion durchgeführt werden. Diese Einzelaktivitäten lösen jeweils einen Zustandswechsel des Gesamtsystems aus. Die Summe aller Use Cases beschreibt die Gesamtfunktionalität des Systems.

Slide 10

Notation

Neben der natürlichsprachlichen Beschreibung existiert eine einfache graphische Notation, welche einen Überblick über das Gesamtsystem liefern soll:

Slide 11



Identifikation von Use Cases

Use Cases werden immer ausgehend von Aktoren identifiziert:

- für jeden Aktor wird festgelegt, welche Ereignisse er erzeugen kann
- die Ereignisse, die diesen initalen Ereignissen folgen, werden zu einem Use Case zusammengefaßt und beschrieben

Slide 12

Folgende Fragen sind dabei interessant:

- Welches sind die Hauptaufgaben eines Aktors?
- Muß der Aktor auf Systeminformation zugreifen?
- Informiert der Aktor das System über die externe Änderungen?
- Wird der Aktor über unvorgesehene Ereignisse informiert?

Instanziierung von Use Cases

Ähnlich wie Aktoren werden auch Use Cases als Klassen aufgefaßt:

- Beim Auslösen eines externen Ereignisses durch einen Aktor wird ein Use Case instanziiert

Slide 13

- Bei Use Cases, die eine gleiche initiale Sequenz von Ereignissen haben, kann erst später identifiziert werden, welcher Use Case begonnen wurde

Abweichungen innerhalb von Use Cases

Nachdem die Use Cases über mehrere Analysedurchläufe hinweg stabil beschrieben sind, werden die unterschiedlichen Ausprägungen definiert:

Hauptablauf: Beschreibung des normalen oder am häufigsten vorkommenden Ablaufs

Slide 14

alternative Abläufe: Beschreibung der Ereignisse im Fehlerfall

Erweiterung von Use Cases

Analog zur Vererbung von Klasseneigenschaften können Use Cases auch erweitert werden. Ausgehend von einem kompletten Ablauf werden hierfür Erweiterungen spezifiziert, die folgenden Zwecken dienen können:

Slide 15

- Beschreibung von optionalen Abläufen
- Beschreibung komplexer, seltener Abläufe
- Beschreibung eines allgemeinen Ablaufs, auf dem mehrere spezialisierte Abläufe aufbauen

Der zu erweiternde Ablauf muß dabei unabhängig von den Erweiterungen beschrieben sein.

Interface-Beschreibungen

Neben der Beschreibung der Funktionalität wird die Interaktion mit dem System in Interface-Beschreibungen konkretisiert:

Slide 16

- Beschreibung der Benutzerschnittstelle (graphisch, Kommandozeile usw.)
- Beschreibung von Protokollen
- Erstellung von Prototypen

Objekte des Problembereichs

Als drittes Modell wird in der Anforderungsanalyse insbesondere bei ungenauen Anforderungsbeschreibungen zusätzlich versucht, ein logisches Modell des Problembereichs zu erstellen. Es hat folgende Aufgaben:

Slide 17

- Schaffung eines gemeinsamen Vokabulars
- Hilfestellung beim Verstehen des Problembereichs
- Hilfe bei der Identifikation von Use Cases durch Beschreibung der Dinge, die im System bearbeitet werden

Dieses Vorgehen entspricht teilweise den klassischen objektorientierten Analysemethoden

Verfeinerung des Anforderungsmodells

Zur Unterstützung von Wiederverwendung und zur Vorbereitung der Umwandlung in das Analysemodell wird das Anforderungsmodell weiter verfeinert:

Slide 18

- gemeinsame Teile von Use Cases werden als eigenständige, *abstrakte* Use Cases definiert
- konkrete Use Cases benutzen diese abstrakten Use Cases (vgl. dazu die Erweiterung von Use Cases)
- gemeinsame Teile von Aktoren werden ebenso als *abstrake* Aktoren beschrieben

Das Analysemodell

Das Anforderungsmodell wird im zweiten Schritt der Analysephase in ein Analysemodell überführt. Es besteht aus Interface-Objekten, Entity-Objekten und Kontroll-Objekten. Bei der Transformation wird das in den Use Cases beschriebene Systemverhalten auf die Objekte dieser drei Typen aufgeteilt.

Slide 19

Inferface-Objekte

Funktionalität, welche an den Systemgrenzen erbracht wird, wird in Interface-Objekten behandelt. Um diese zu identifizieren, gibt es drei Möglichkeiten:

Slide 20

- direkte Übernahme der Objekte aus der Interface-Beschreibung
- Betrachtung der Interaktion der Aktoren mit dem System
- Betrachtung der interface-spezifischen Teile der Use Cases

Entity-Objekte

Information, die im System über längere Zeit hinweg gehalten wird, wird in Entity-Objekten abgelegt. Sie existieren über mehrere Abläufe von Use Cases hinweg und entsprechen weitgehend den Objekten, die im Modell des Problembereichs identifiziert werden konnten.

Slide 21

Kontroll-Objekte

Kontroll-Objekte verbinden Interface- und Entity-Objekte und können direkt aus den Use Cases abgeleitet werden. Ein Kontroll-Objekt sollte an möglichst wenigen Use Cases beteiligt sein. Kontroll-Objekte sollen die Modifikation von Systemverhalten erleichtern.

Slide 22

Probleme bei der Verwendung

Bei der Benutzung von Use Cases in Verbindung mit objektorientierten Techniken (eine verbreitete Kombination) sind jedoch folgende Punkte zu beachten:

Slide 23

- Use Cases stellen nur eine informelle Systembeschreibung dar
- sie stellen die Reihenfolge von Operationen in den Vordergrund, beim objektorientierten Entwurf ist die Reihenfolge jedoch nachrangig
- sie zerlegen ein System nach funktionalen Gesichtspunkten, dies widerspricht objektorientieren Vorgehensweisen

Es ist deshalb empfehlenswert, Use Cases nur als Mittel zum Erfassen und Prüfen von Anforderungen zu benutzen, nicht aber als Analyse- und Designmethode