FHD

Fachhochschule Darmstadt
Fachbereich Informatik

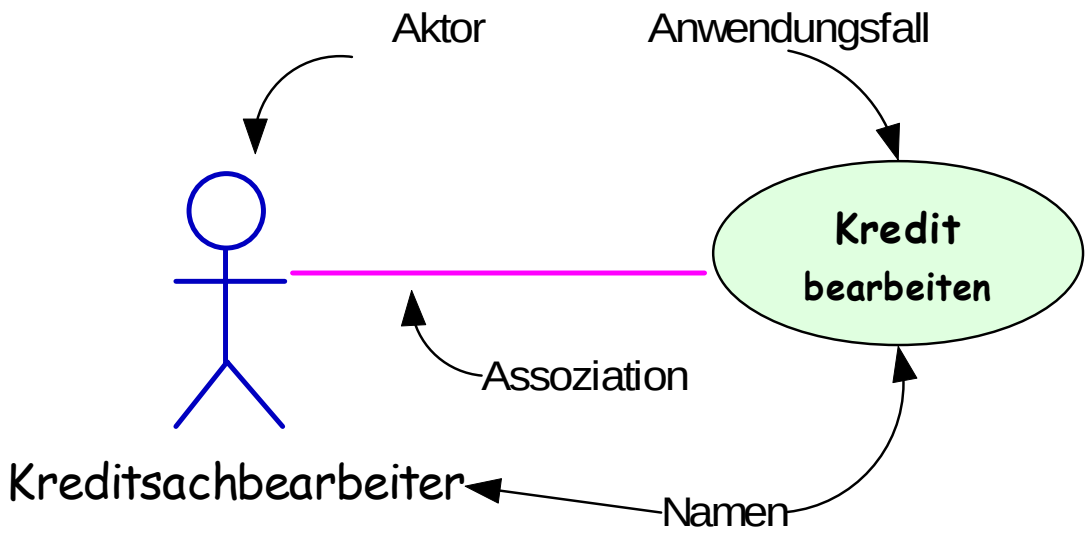
Softwaretechnik I

Kapitel 3 ff

- Use Cases & UML

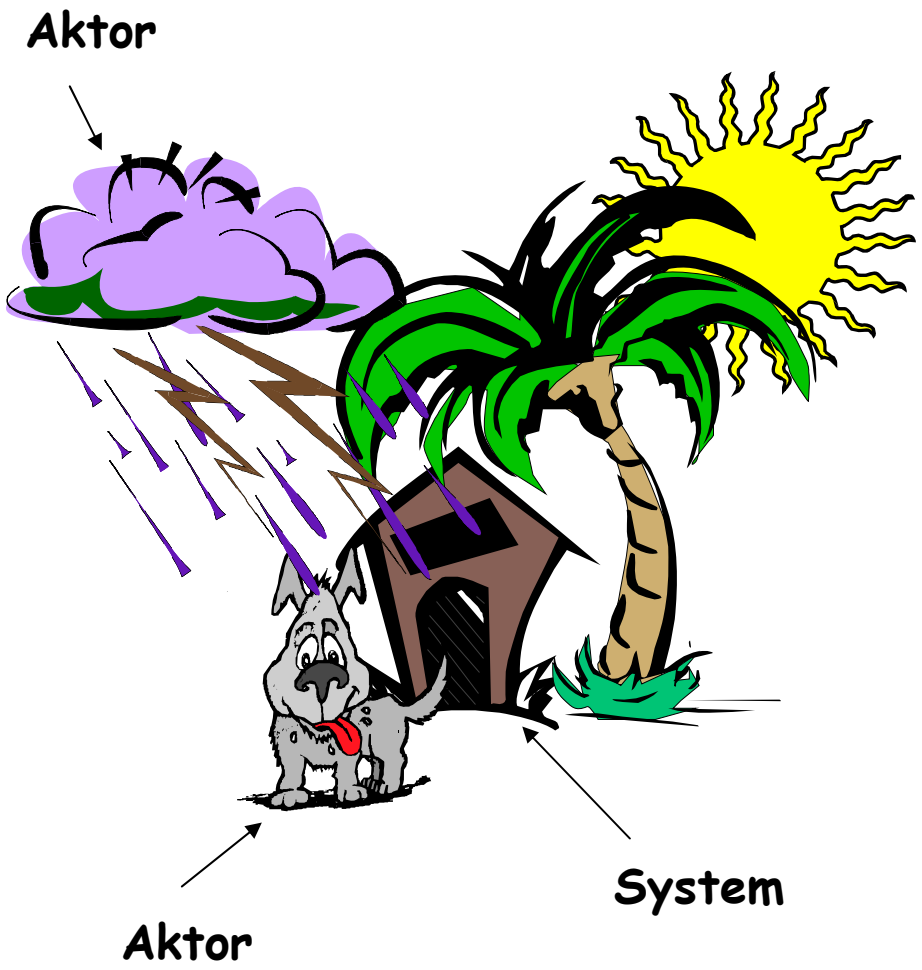
Quellenhinweis:
Einige Folien zu dieser Vorlesung entstammen Präsentationen von Prof. G. Raffius und Prof. W. Weber

Namen und Akteure



- Jeder Anwendungsfall muss einen Namen haben, der ihn von anderen Anwendungsfällen unterscheidet
- Namen sollten kurze Formulierungen mit aktiven Verben aus dem Vokabular des modellierten Systems sein

Der Kontext eines Systems



- Alle Dinge außerhalb des Systems, die mit dem System interagieren, bilden den Kontext eines Systems.
- Der Kontext definiert die Umgebung, in der das System lebt.
- Ein Produkt kann häufig in unterschiedlichen Kontexten betrieben werden.
- Innerhalb eines Kontextes kann es verschiedene Anwendungsfälle geben
- Durch den Kontext werden die nicht funktionalen Anforderungen vorgegeben

Definitionen (I)

- Aktor
  - irgend jemand oder etwas mit Verhalten. Auch das zu entwickelnde System ist ein Aktor
- Stakeholder
  - jemand oder etwas mit einem persönlichen Interesse am Verhalten des Systems unter Diskussion (SuD)
- Primary Actor
  - der Stakeholder, der die Interaktion mit dem SuD initiiert um ein Ziel zu erreichen
- Use Case
  - ein Vertrag für das Verhalten des SuD
- Scope
  - identifiziert das System über das wir diskutieren

Definitionen (II)

- Level
  - Auf welcher Ebene ist das Ziel angesiedelt. Gesamtziel, Userziel oder Teilfunktion?
- Preconditions and guarantees
  - was wahr sein muss vor und nach dem Ablauf eines Use Case
- Trigger
  - Auslöser der Use Cases
- Main Success scenario
  - Ein Use Case in dem alles funkioniert (nichts falsch wird)
- Extensions
  - was unterschiedlich imAblauf sein kann

Auslöser von Use Cases

- Der Start eines Use Cases wird durch ein Ereignis (Trigger) ausgelöst.
- Ereignisse können sowohl **asynchrone Ereignisse**, ausgelöst durch Akteure oder Systeme in der Umgebung sein, als auch **Zeitereignisse**
- Fragestellungen: **„Wer tut Was“** und **„Es ist Zeit für ...“**
- Beispiele
  - Bediener schaltet System ein
  - Es ist Zeit für ein Backup

Stakeholder und Akteure

- Ein Stakeholder ist jemand, der ein Interesse am Verhalten eines Use Cases hat.
- Ein Aktor ist jemand oder etwas, das ein eigenes Verhalten besitzt.
  - Ein Aktor (actor) repräsentiert eine Rolle, die ein Mensch ein Gerät oder ein anderes System gegenüber dem zu entwickelnden System spielt
  - Jeder Aktor ist gleichzeitig auch ein Stakeholder des Systems, aber nicht umgekehrt
  - Das System selbst (SuD, System under Design) ist ebenfalls ein Aktor
  - In einem Use Case können verschiedene Aktoren auftreten
    - Haupt Aktor (primary actor)
    - Hilfs Aktor (supporting actor)
    - interner Aktor (internal Aktor)
- Akteure können mit Anwendungsfällen durch Assoziationen verbunden werden. Dies zeigt, dass Anwendungsfall und Aktor miteinander kommunizieren und jeder Nachrichten sendet und empfängt

Dokumentation: Aktor Profil Tabelle

Name	Profil: Hintergrund und Fähigkeiten
Kunde	Person von der Straße, kann einen Touch Screen benutzen, aber es wird nicht angenommen, dass er Erfahrung mit GUIs hat. Kann Schwierigkeiten mit dem Lesen haben, kann kurzsichtig sein
Operator	Person, die ständig mit der Software arbeitet. Ist ein erfahrener Benutzer und kann den Wunsch haben, sich die Oberfläche anzupassen
Manager	Gelegentlicher Nutzer, kennt GUIs, ist aber nicht vertraut mit irgendeiner speziellen Software Funktion. Ungeduldig

Dokumentation: Muster für Use Case Spezifikation (I)

Use Case ## „Use Case Name“ <name>

Context of Use:

<a longer statement of the context of use if needed>

Scope:

<what system is being considered black-box under design>

Level:

<one of: summary, user-goal, subfunction>

Primary Actor:

< a role name for the primary actor or description>

Stakeholder and Interests:

<list of stakeholders and key interests in the use case>

Precondition:

<what we expect is already the state of the world>

Minimal Guarantees:

<how the interests are protected under all exits>

Sucess Guarantees:

<the state of the world if goal succeeds>

Trigger:

<what starts the Use Case, may be time event>

Dokumentation: Muster für Use Case Spezifikation (II)

Main Sucess Scenario:

<put here the steps of the scenario from trigger to goal delivery and any cleanup after>

<step #> <action description>

<step #> <action description>

Extensions:

<put here the extensions one at time, referring to the steps of the main scenario>

<step altered><condition>: <action or sub use case>

<step altered><condition>: <action or sub use case>

Technologie & Data Variations:

<put here the variations that will cause eventual bifurcation in the scenario>

<step or variation # > < list of variations>

<step or variation # > < list of variations>

Related Information:

<whatever your project needs for additional information>

Beispiel Geldautomat (Ablauf als Text-Template)

Use Case Benutzer validieren

- **Hauptablauf:**
  - Der Anwendungsfall beginnt, wenn das System den Kunden nach seiner PIN Nummer fragt.
  - Der Kunde kann nun eine PIN Nummer über die Tastatur eingeben.
  - Der Kunde schließt die Eingabe durch das Betätigen der Eingabetaste ab.
  - Das System überprüft dann die PIN Nummer auf ihre Gültigkeit.
  - Ist die PIN Nummer gültig, akzeptiert das System die Eingabe und der Anwendungsfall ist beendet.
- **Ausnahmeablauf:**
  - Der Kunde kann eine Transaktion jederzeit durch Betätigen der Abbruch Taste beenden und den Anwendungsfall so neu beginnen
- **Ausnahmeablauf:**
  - Der Kunde kann die PIN Nummer jederzeit löschen, bevor er die Eingabetaste betätigt und eine neue PIN Nummer eingeben
- **Ausnahmeablauf:**
  - Wenn der Kunde eine ungültige PIN Nummer eingibt, beginnt der Anwendungsfall erneut. Passiert dies dreimal in Folge, bricht das System die gesamte Transaktion ab und unterbindet für 60 Sekunden jede Interaktion mit dem Benutzer

Template in Tabellenform

USE CASE # / Name			
Context of Use			
Scope			
Level			
Primary Actor			
Stakeholder and Interests	Stakeholder	Interest	
Precondition			
Minimal Guarantee			
Sucess Guarantees			
Trigger			
Main Sucess Scenario	Step #	Action	
Extension	Step #	Condition	Action
Technologie and Data Variations	Step #	Variation	
Related Information			

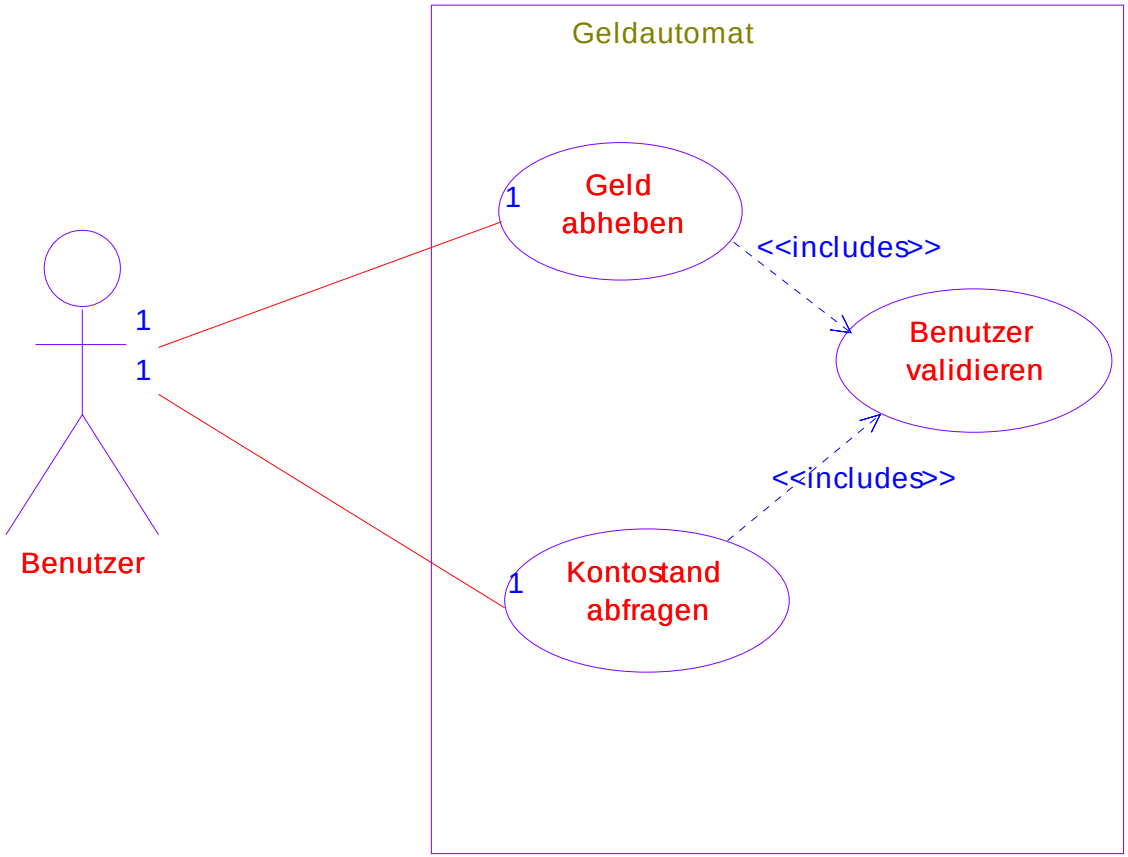
Beispiel Geldautomat (I) – als Tabelle

USE CASE # / Name	1	Benutzer validieren
Context of Use	Geldautomat einer Bank	
Scope	System under Design (Black Box)	
Level	Subfunction	
Primary Actor	System	
Stakeholder and Interests	Stakeholder	Interest
	Bank	Schutz vor unberechtigtem Zugriff
Precondition	Kunde hat Kreditkarte eingeschoben	
Minimal Guarantee	Kein unberechtigter Zugang	
Sucess Guarantees	Zugriff wird gewährt	

Beispiel Geldautomat (II) – als Tabelle

Trigger	System fragt Kunden nach PIN		
Main Sucess Scenario	step #	Action	
	1	Kunde gibt PIN ein	
	2	System validiert PIN	
	3	System gibt Zugang frei	
Extension	step #	Condition	Action
	*	Kunde bricht Vorgang ab	Beenden des Anwendungsfalls
	2a	PIN stimmt nicht und Versuche < 3	Weiter mit Schritt 1
	2b	PIN stimmt nicht und Versuche >= 3	System für 60s sperren
Technologie and Data Variations	step #	Variation	
Related Information			

Use Cases Geldautomat



Gedanken zu Use Cases (I)

- Use Cases sind geeignet um das Verhalten eines Systems zu visualisieren, zu spezifizieren, und zu dokumentieren
- Ein Use Case beschreibt eine Menge von Aktionsfolgen, einschließlich Varianten, die ein System ausführt um ein erkennbares, für einen Akteur nützliches Ergebnis zu erarbeiten.
- Ein Use Case beschreibt nicht, wie das Verhalten implementiert wird
- Use Cases geben eine Außenansicht von Systemen. Sie zeigen wie das System im Zusammenhang verwendet wird

Gedanken zu Use Cases (II)

- Use Cases geben eine Möglichkeit für den Entwickler mit den Endanwendern und den Fachleuten für den Anwendungs-bereich zu einem gemeinsamen Verständnis zu gelangen
- Sie schaffen die Basis für die Requirements die Überprüfung der Architektur und den späteren Test
- Ein Anwendungsfall stellt eine funktionale Anforderung an das System als Ganzes dar
- Der Kontext eines Anwendungsfalls liefert non-functional Requirements
- Die Sammlung der Use cases ergeben noch keine vollständige Sammlung der Requirements

Übung: Use Case "Geld abheben" (I)

USE CASE # / Name	2	Geld abheben
Context of Use		
Scope		
Level		
Primary Actor		
Stakeholder and Interests	Stakeholder	Interest
Precondition		
Minimal Guarantee		
Sucess Guarantees		

Übung: Use Case "Geld abheben" (I)

USE CASE # / Name	2	Geld abheben
Context of Use	Geldautomat einer Bank	
Scope	System under Design (Black Box)	
Level	User	
Primary Actor	User	
Stakeholder and Interests	Stakeholder	Interest
	Bank	Schutz vor unberechtigtem Zugriff
	User	Abbuchung nur nach Auszahlung
Precondition	keine	
Minimal Guarantee	Keine unberechtigte Auszahlung	
Sucess Guarantees	Geld wird ausgezahlt, Karte zurückgegeben	

Übung: Use Case "Geld abheben" (II)

Trigger			
Main Sucess Scenario	step #	Action	
	1		
	2		
	3		
	4		
Extension	step #	Condition	Action
	*		
	1a		
	2a		
Technologie and Data Variations	step #	Variation	
	2		
Related Information			

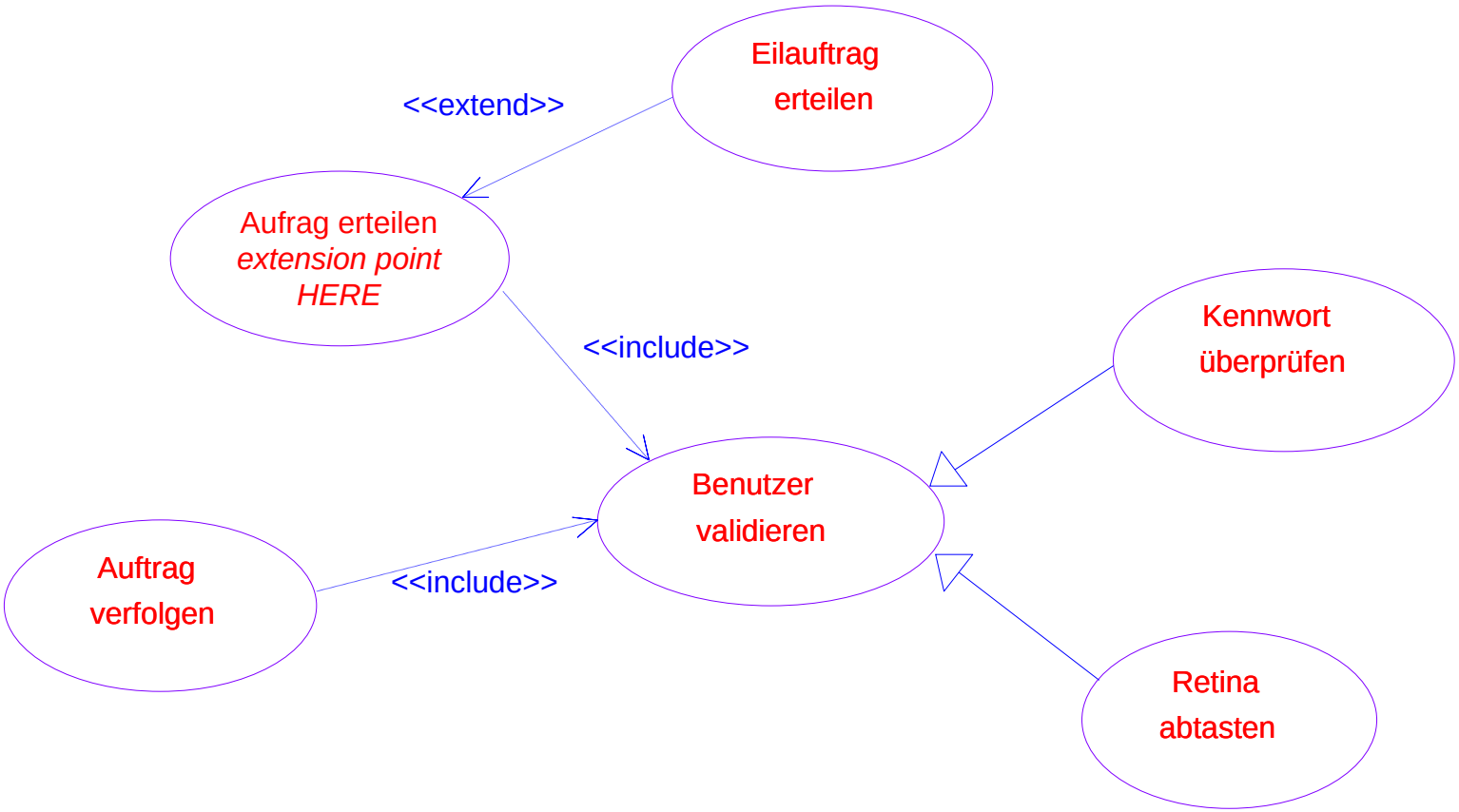
Übung: Use Case "Geld abheben" (II)

Trigger	Benutzer schiebt Karte in das System		
Main Sucess Scenario	step #	Action	
	1	Use Case: Benutzer validieren	
	2	Kunde gibt Wunschbetrag ein	
	3	System gibt Karte aus, Kunde entnimmt Karte	
	4	System gibt Geld aus, Kunde entnimmt Geld	
Extension	step #	Condition	Action
	*	Kunde bricht Vorgang ab	Beenden des Anwendungsfalls
	1a	Validierung schlägt fehl	Karte ausgeben und Beenden des Anwendungsfalls
	2a	Betrag über Limit	Weiter mit Schritt 2
Technologie and Data Variations	step #	Variation	
	2	Bei Summen über 1000€ stellt das System eine Überprüfungsfrage (im Stil „Sind Sie sicher?)	
Related Information	Die Karte wird VOR dem Geld ausgegeben. Ansonsten wird die Karte oft nicht mitgenommen.		

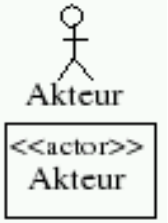
Beziehungen zwischen Use Cases

- Extend
  - Die erweitert (extend) Beziehung sagt aus, dass der Basisanwendungsfall durch den anderen Anwendungsfall erweitert wird (z.B. optionales Verhalten).
- Include
  - Der Basisanwendungsfall bezieht explizit das Verhalten des anderen Anwendugsfalls ein
- Generalisierung
  - der spezialisierte Anwendungsfall ererbt das Verhalten des allgemeineren Anwendungsfalls und kann dieses ergänzen oder überschreiben

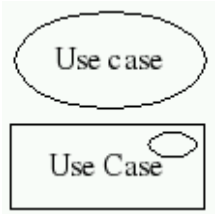
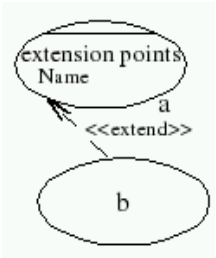
Beziehungen zwischen Use Cases am Beispiel



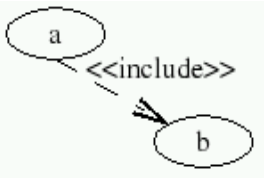
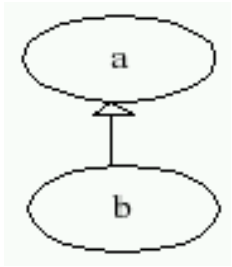
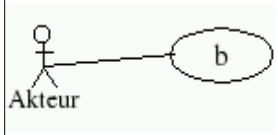
Notation UML 2.0 (I)

Symbol	Bedeutung
	**System** Das Rechteck stellt das geplante System dar. Das System erhält einen Namen. Ein Use Case Diagramm kann auch mehrere Systeme enthalten, die ineinander geschachtelt sein dürfen. Dadurch kann ein System in Teilsysteme gegliedert werden.
	**Akteur** Ein Akteur ist ein Element, das nicht zum geplanten System gehört. Er kann eine Person sein, die auf das System zugreift, oder ein anderes System, das mit dem geplanten System kommuniziert. Die UML erlaubt mehrere Symbole zur Darstellung eines Akteurs. Er kann als Strichmännchen dargestellt werden. Es ist optional erlaubt ein Klassensymbol zu verwenden, das mit dem Stereotyp <<actor>> markiert wird. Zusätzlich können eigene Symbole verwendet werden um nicht menschliche Akteure darzustellen.

Notation UML 2.0 (II)

	**Use Case** Eine Ellipse stellt einen Anwendungsfall des Systems dar. Ein Anwendungsfall ist ein in sich abgeschlossener Vorgang, der für einen oder mehrere Akteure ein beobachtbares Ergebnis liefert. Er beschreibt aus Sicht der Akteure welche Leistungen das System für den Anwender zur Verfügung stellt. Ein Use Case stellt somit einen Teil der Gesamtfunktionalität des Systems dar. In UML 2.0 kann auch ein Rechteck, das mit einer Ellipse markiert wird, als Use-Case-Symbol verwendet werden. Der Name kann innerhalb oder außerhalb des Symbols stehen.
	**Extension points** Das Verhalten eines Use Case kann durch einen weiteren Use Case erweitert werden. Dies wird durch die extend-Beziehung und den extension point angegeben. Die Beziehung wird gestrichelt dargestellt; der Pfeil zeigt auf den erweiterten Use Case und trägt den Stereotyp <<extend>>. Der extension point gibt den Ort im erweiterten Use Case an, an dem der erweiternde Use Case eingefügt wird. b erweitert das Verhalten von a. b wird an dem extension point "Name" in a eingefügt.

Notation UML 2.0 (III)

	**include-Assoziation** Die include Beziehung definiert einen Use Case, der die Funktionalität, die ein anderer Use Case zur Verfügung stellt, importiert. Der Use Case a importiert die Funktionalität des Use Case b.
	**Vererbungsbeziehung** In Use Case Diagrammen ist die Verebungsbeziehung erlaubt. Sie kann verwendet werden um Verhalten zwischen Use Cases zu vererben. Sie kann aber auch verwendet werden, um Vererbungsbeziehungen zwischen Akteuren aufzubauen.
	**Assoziation** Eine Linie stellt eine Assoziation zwischen einem Akteur und einem Use Case dar. Sie beschreibt den Zugriff des Akteurs auf die Funktionalität, die das System in diesem Use Case zur Verfügung stellt, bzw. eine Antwort des Systems an einen Akteur.

Use Cases für ein Gesamtsystem: Beispiel Hochschulverwaltung (I)

Aufgabe:

Modellieren Sie ein Use Case Diagramm für das System Hochschulverwaltung.

Anforderungen:

Das System Hochschulverwaltung soll folgende Bereiche abdecken:

Benutzertypen sind

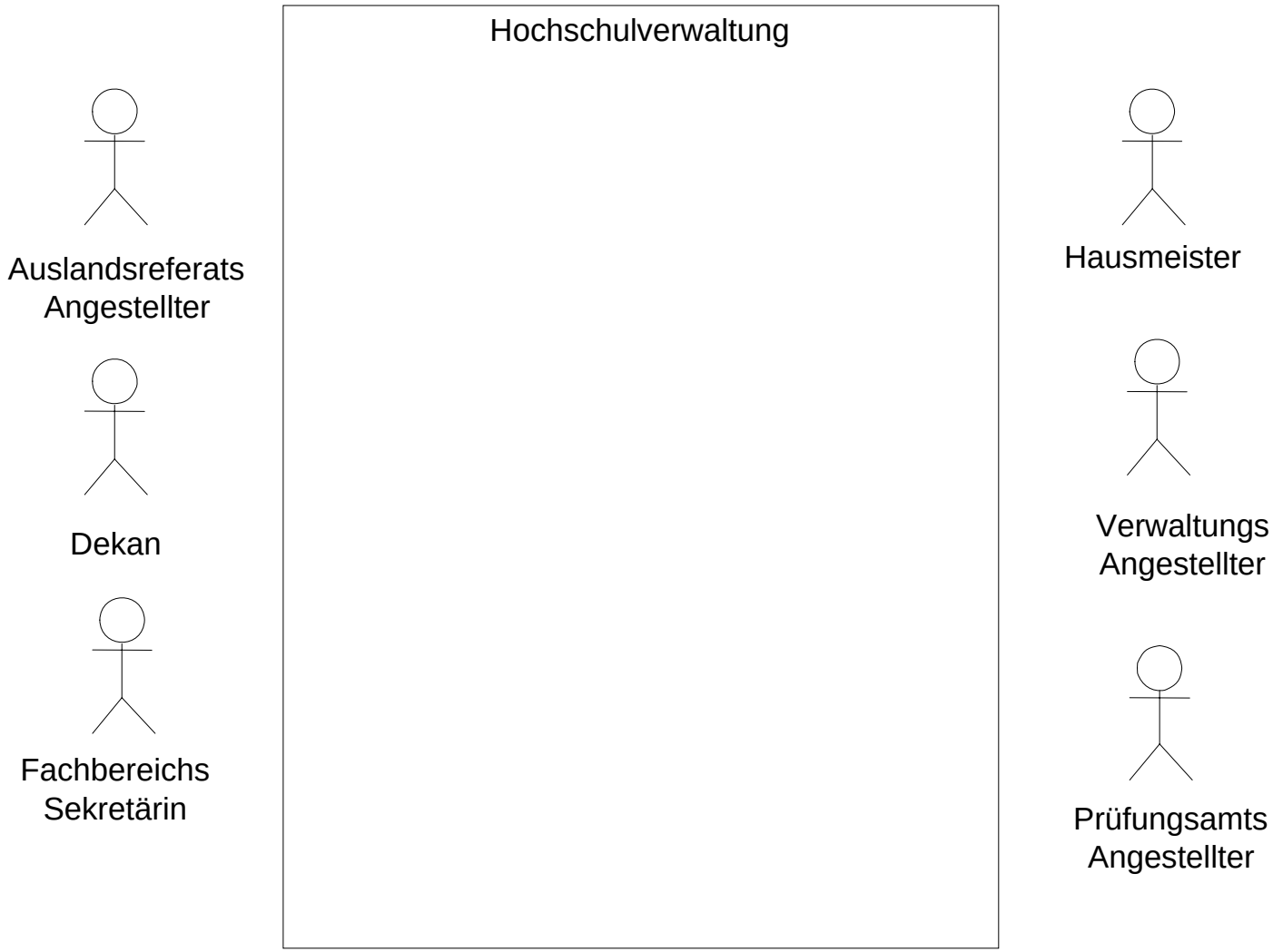
- Auslandsreferats-Angestellter
- Prüfungsamtsangestellter
- Hausmeister
- Verwaltungsangestellter
- Dekan
- Fachbereichssekretärin

Use Cases für ein Gesamtsystem: Beispiel Hochschulverwaltung (II)

- Immatrikulation von Studenten,
- Exmatrikulation von Studenten,
- Rückmeldung von Studenten
- Erfassen von Prüfungsergebnissen,
- Anmeldung zur Diplomarbeit und Registrierung der Gutachten
- Verwaltung von Studienaustausch u.Immatrikulation von ausländischen Studierenden im Rahmen eines Austauschs
- Das Erfassen und Verwalten von Lehrbeauftragten und Tutoren
- Das Verwalten von verliehenem Hochschuleigentum an Studenten, Tutoren und Lehrbeauftragte (Schlüssel, Bücher etc.)
- Das Erstellen von Lehrveranstaltungskalendern mit Zeit- und Raumbelegung

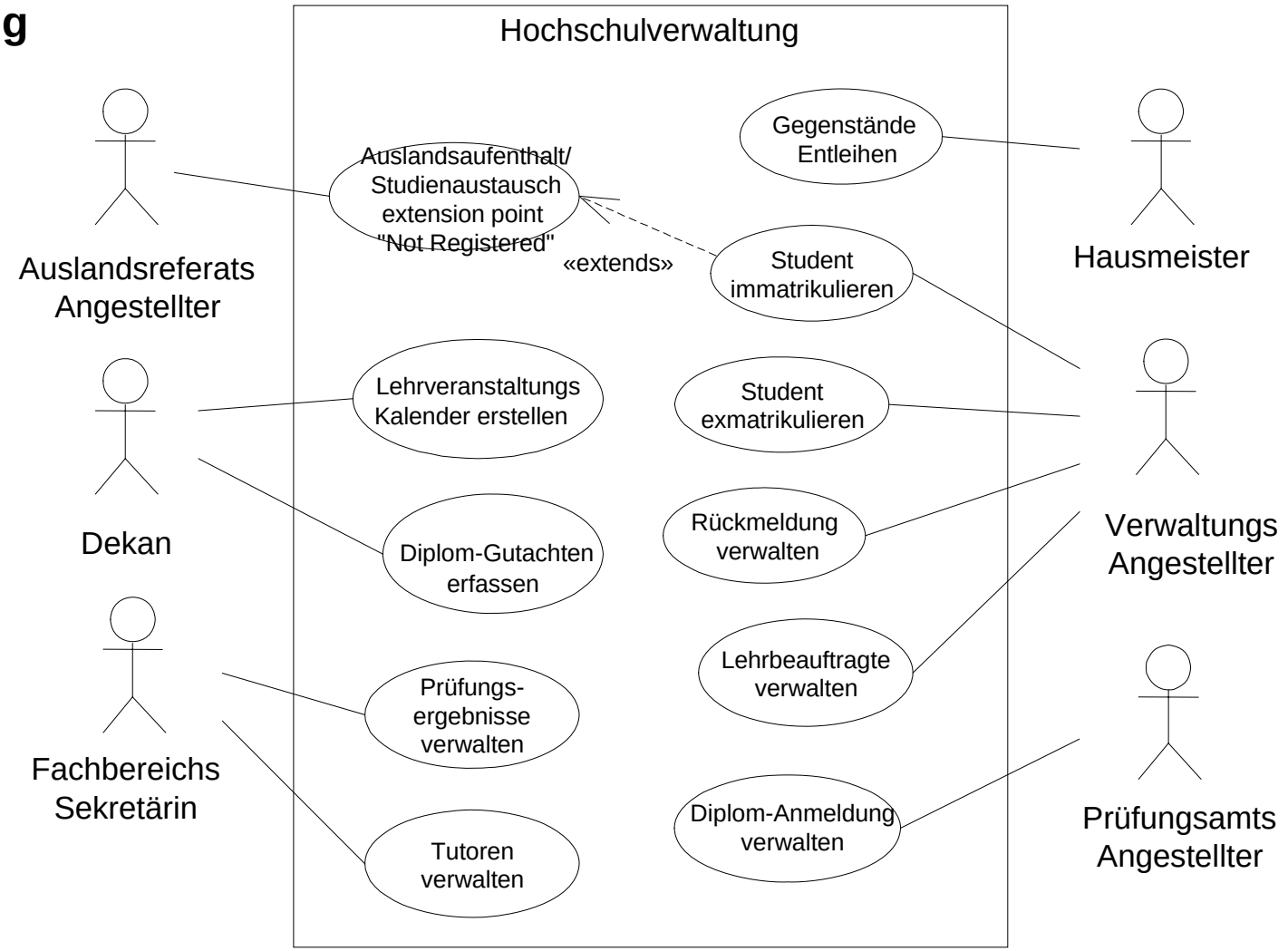
Use Cases für ein Gesamtsystem: Beispiel Hochschulverwaltung (III)

Übung



Use Cases für ein Gesamtsystem: Beispiel Hochschulverwaltung (III)

Lösung



Zwischenstand Use Cases

Bekannt:

- Darstellung eines einzelnen Use Cases
- Darstellung eines Gesamtsystems mit Use Cases
- Darstellung mit UML 2.0

Noch (weitgehend) offen:

- Wie finde ich die Aufteilung der Use Cases?
- Was lagere ich als Unter-Use Case aus?
- Welche Granularität / welchen Level brauche ich?
- Was gehört in der Beschreibung wohin?

Lösung:

- Regeln, Tipps...
- ... und Iterationen

12 Schritte zu Use Cases für ein System (I)

1. Finde die Grenzen des Systems
   (Kontext Diagramm, IN/OUT Liste)
2. Brainstorme und liste die Aktoren (Aktor Profil Tabelle)
3. Brainstorme und liste die Ziele der Aktoren in Bezug auf das System
   (Aktor-Ziele Liste)
4. Schreibe die äußersten Summary Use Cases, die das gefundene zusammenfassen
5. Überdenke und verbessere die strategischen Use Cases. Füge Ziele hinzu, entferne Ziele und fasse zusammen
6. Nehme einen Use Case und schreibe eine Geschichte, um mit dem Material vertraut zu werden
7. Füge die Stakeholder hinzu, deren Interessen, Vorbedingungen und Garantien. Überprüfe sie doppelt.

12 Schritte zu Use Cases für ein System (II)

8. Schreibe das Haupterfolgsszenario (main success scenario). Vergleiche es mit den Interessen und Garantien
9. Brainstorme und liste die möglichen Fehlerbedingungen und die alternativen Erfolgsbedingungen
10. Beschreibe wie die Aktoren und das System sich in den Erweiterungen (extensions) Verhalten sollen
11. Breche jeden Unter Use Case heraus, der seinen eigenen Raum braucht
12. Beginne vom Anfang und verbessere die Use Cases. Füge hinzu, entferne oder fasse zusammen wenn angebracht. Überprüfe Vollständigkeit, Lesbarkeit und Fehlerbedingungen.

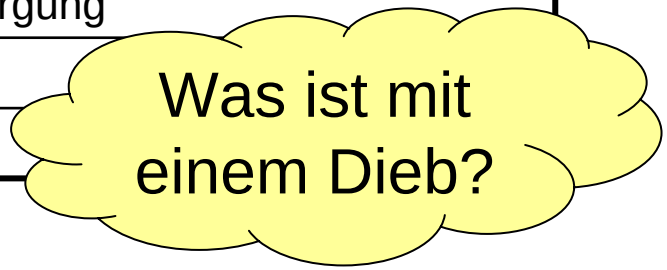
Übung: Use Cases für das System „Geldautomat“

- Spielen Sie die 12 Schritte für das Beispiel „Geldautomat“ durch
- Erstellen Sie keine Use Cases im Detail
- Bestimmen Sie nur die Gesamtstruktur der Use-Cases
- Verwenden Sie auch die Assoziationen zwischen Use Cases

Übung: Use Cases „Geldautomat“ – Ergebnis (1. Iteration, unvollständig)

1. Grenzen des Systems

IN	**OUT**
Glasscheibe	Geld
Tastatur	Unterbrechungsfreie Stromversorgung
Kartenleser	EC-Karte
Bildschirm	



2. Aktoren und 3. Ziele

Aktoren	**Ziele**	**Zu klären**
Bankkunde	Geld abheben	
	Kontostand abfragen	
Service Techniker	Fehlerprotokoll auslesen	Remote?
	Hardware-Selbsttest	Remote?
Geld-Bote	Geld einfüllen	
Bank-Verantw. GA	Remote-Abfrage Status	
?? System??	Fehler-Benachrichtigung	

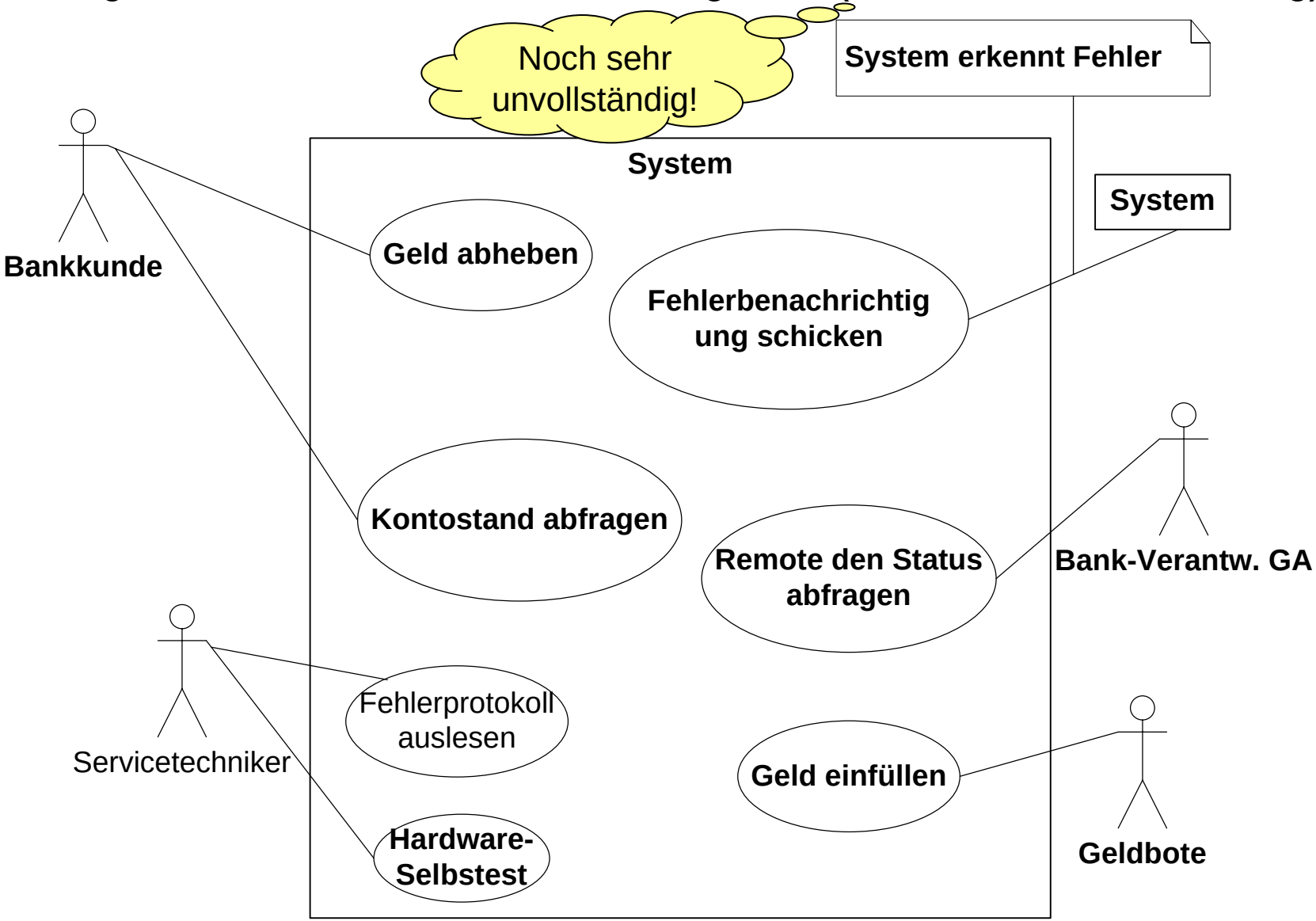
Übung: Use Cases „Geldautomat“ – Ergebnis (1. Iteration, unvollständig)

4. Summary Use Cases
   - Bank: Kostengünstigen Bargeldverkehr realisieren
   - Kunde: (und Bank): Rund-um-die-Uhr verfügbarer Bargeldverkehr

…

7. Stakeholder, Interessen, Garantien
   - Gesetzgeber: Dokumentieren der Zahlungsvorgänge
   - usw.
8. Haupterfolgsszenario:
   - Geldauszahlung: Karte rein, richtige PIN eingeben, Betrag eingeben, Karte raus, Geld raus
9. Fehler, Alternativen:
   - Karte falsch, PIN falsch, Konto überzogen, Geld-Stau, Karte nicht entnommen, Geld nicht entnommen
10. Verhalten der Aktoren / des Systems in den Extensions
    - Karte ist keine EC-Karte: System wirft Karte aus, gibt Meldung aus
    - usw.
11. Use Cases rauziehen?
12. Iterierieren? Unbedingt – und vervollständigen!

Übung: Use Cases „Geldautomat“ – Ergebnis (1. Iteration, unvollständig)



Tipps (I): Mache die Use Cases einfach lesbar (jeden einzelnen)

- Halte dich kurz und bringe es auf den Punkt
- Starte von oben und erzeuge eine stimmige Geschichte
- Oben befindet sich ein strategischer (summary) Use Case
- Die User Goal und Subfunction Use Cases verzweigen von hier aus
- wähle Namen mit kurzen Verben, die das Ziel des Use Cases ausdrücken
- starte vom Trigger und fahre fort bis das Ziel erreicht oder aufgegeben ist
- Ein Use Case hat zwischen 3 und 9 Schritten. Wenn mehr als 9 Schritten vorhanden sind, fasse die Schritte zusammen.
- Schreibe volle Sätze mit aktiven Verben die das Unterziel, das erreicht werden soll, ausdrücken
- Stelle sicher, dass der Aktor und seine Absicht in jedem Schritt sichtbar werden
- Stelle sicher, dass die Fehlerbedingungen gut erkennbar und die recovery Actions gut lesbar sind.
- Stelle sicher, dass gut erkennbar ist was als nächstes passiert, auch ohne die Nummern der Schritte
- Füge alternatives Verhalten in die Extensions ein und nicht als „if statements“ in den Hauptablauf
- Erzeuge Extension Use Cases nur unter ausgewählten Bedingungen

Tipps (II): Erinnerungen für jeden Use Case

- Halte die Benutzeroberfläche heraus
  - Stelle sicher, dass die Schritte die Du beschrieben hast die wahre Absicht des Aktors widerspiegeln und nicht die Bewegungen des Aktors beim manipulieren der Oberfläche.
  - Dies gilt nur, wenn funktionale Requirements beschrieben werden.
  - Man kann Use Cases auch benutzen um die Benutzer-Oberflächen zu dokumentieren

- Jeder Use Case hat zwei Enden: Erfolg und Misserfolg
  - Beachte, dass auch ein Sub Use Case mit Erfolg oder Misserfolg enden kann
  - Taucht der Fehler im Hauptablauf auf, so wird er in den Extensions behandelt.
  - In der Extension werden Erfolg und Fehlerbehandlung in der gleichen Extension beschrieben

Tipps (III): Ceckliste 1

- **Use Case Titel**
  - Ist es eine Formulierung mit einem aktiven Verb, die das Ziel des Primary Actors ausdrückt?
  - Kann das System das Ziel erfüllen?
- **Scope und Level**
  - Sind die Felder ausgefüllt?
- **Scope**
  - Behandelt der Use Case das betrachtete System als ein „Black Box System“?
  - Wenn das betrachtete System das zu entwickelnde System ist, haben dann die Designer alles im System zu entwickeln und nichts außerhalb?
- **Level**
  - Entspricht der Inhalt des Use Cases dem Ziel Level?
  - Ist das Ziel auf dem richtigen Level?
- **Primary Actor**
  - Hat der Primary Actor ein eigenes Verhalten?
  - Hat der Primary Actor ein Ziel gegenüber dem SuD dem ein Service Angebot des SuD entspricht?

Tipps (IV): Checkliste 2

- **Vorbedingungen**
  - Sind die Vorbedingungen obligatorisch und können sie durch das SuD erreicht werden?
  - Ist es wahr, dass sie im Use Case nicht mehr überprüft werden?
- **Stakeholder und Interessen**
  - Sind die Stakeholder und ihre Interessen alle aufgeführt und muss das System die Interessen alle erfüllen?
- **Minimale Garantien**
  - Sind die Interessen aller Stakeholder geschützt?
- **Erfolgsgarantien**
  - Sind die Interessen aller Stakeholder befriedigt?
- **Hauptablauf**
  - Hat der Hauptablauf zwischen 3 und 9 Schritten?
  - Läuft der Hauptablauf vom Trigger bis zur Erfüllung der Erfolgsgarantie?
  - Erlaubt der Ablauf die richtigen Variationen in der Reihenfolge?
- **Erweiterungsbedingungen**
  - Muss das System sie erkennen und behandeln?
  - Braucht das System die Erweiterung wirklich?

Tipps (V): Checkliste 3

- **Jeder Schritt des Szenarios**
  - Ist der Schritt als ein Ziel formuliert, das es zu erreichen gilt?
  - Läuft der Prozess deutlich weiter nach der erfolgreichen Erfüllung des Schritts?
  - Ist es klar, welcher Aktor das Ziel betreibt - wer „spielt den Ball“?
  - Ist die Absicht des Aktors klar?
  - Ist das Ziel des Schrittes niedriger als das Ziel Level des gesamten Use Cases? Ist es vorzugsweise nur ein bisschen unter dem Ziel Level?
  - Ist sicher gestellt, dass der Schritt nicht das User Interface Design des Systems beschreibt?
  - Ist klar welche Information in dem Schritt ausgetauscht wird?
  - Validiert der Schritt (im Gegensatz zu überprüft)?
- **Technologie und Daten Variationen**
  - Ist sichergestellt, dass es sich nicht um eine normale Erweiterung des Verhaltens des Hauptablaufs handelt?
- **Inhalt des Use Cases als Ganzes**
  - Frage an den Auftraggeber und den Nutzer: „Ist es das, was Ihr wollt?"
  - Frage an den Auftraggeber und den Nutzer: „Kannst Du nach der Lieferung bestätigen, dass dies geliefert wurde?“
  - Frage an den Entwickler: „Kannst Du das implementieren?"

Tipps (VI): Guidelines für den Hauptablauf

- Benutze eine einfache Grammatik
  - Subjekt ... Verb ... Objekt ... Präposition
- Zeige klar: „Wer hat den Ball“
- Schreibe aus der Vogelperspektive
- Zeige wie sich der Prozess fortbewegt
- Zeige was der Aktor will, nicht seine Bewegungen
- Benutze eine sinnvolle Menge von Aktionen
- Wähle exakte Begriffe und Wörter
  - z.B. Benutze „validiere“, und nicht „überprüfe“! Das System validiert das Passwort und überprüft es nicht
- wenn sinnvoll, erwähne die Zeit

⇨ Idiom: „User has System A kick System B“

⇨ Idiom: „Tue Schritte x-y bis Bedingung“

Tipps (VII): (Potenzielle) Stakeholder

- Mögliche Stakeholder sind
  - der Primary Actor
  - der Firmenbesitzer
  - gesetzliche Behörden
  - der Testingenieur
  - der Serviceingeneur
- Ein Use Case beschreibt nicht nur die Interaktionen zwischen dem Primary Actor und dem System.
- Ein Use Case beschreibt wie das System die Interessen aller Stakeholder schützt, mit dem Primary Actor als treibender Kraft

- Das Interesse des Primary Actors wird im Use Case Namen festgehalten. Normalerweise erhält er etwas
- Das Firmeninteresse besteht meistens darin, dass der Primary Actor nicht etwas umsonst erhält, für die Leistung bezahlen muss oder keinen Schaden anrichten kann
- Das Interesse des Gesetzgebers ist meist, dass die Firma zeigen kann, dass sie die Regeln einhält und dass die Transaktionen protokolliert werden
- Eines der typischen Interesse von Stakeholdern ist die Behandlung von Fehlern und das Recovery bei Fehlern innerhalb einer Transaktion

Tipps (VIII): Vorbedingungen

- Die Vorbedingungen eines Use Cases definieren die gültigen Randbedingungen unter denen der Use Case ablaufen kann
- Die Vorbedingungen müssen erwähnt werden, da sie im Use Case nicht mehr abgeprüft werden
- Es gibt zwei gebräuchliche Situationen für solche Vorbedingungen
  - Der User ist eingelogged und seine Autorisierung verifiziert
  - Ein zweiter Use Case verlässt sich darauf, dass in einem ersten Use Case Randbedingungen geschaffen wurden, auf die sich der zweite Use Case verlassen kann
- Immer wenn man eine Vorbedingung findet, kann man davon ausgehen, dass es einen Use Case auf höherem Level gibt, in dem die Vorbedingung gesichert wurde

Zusammenfassung Use Cases

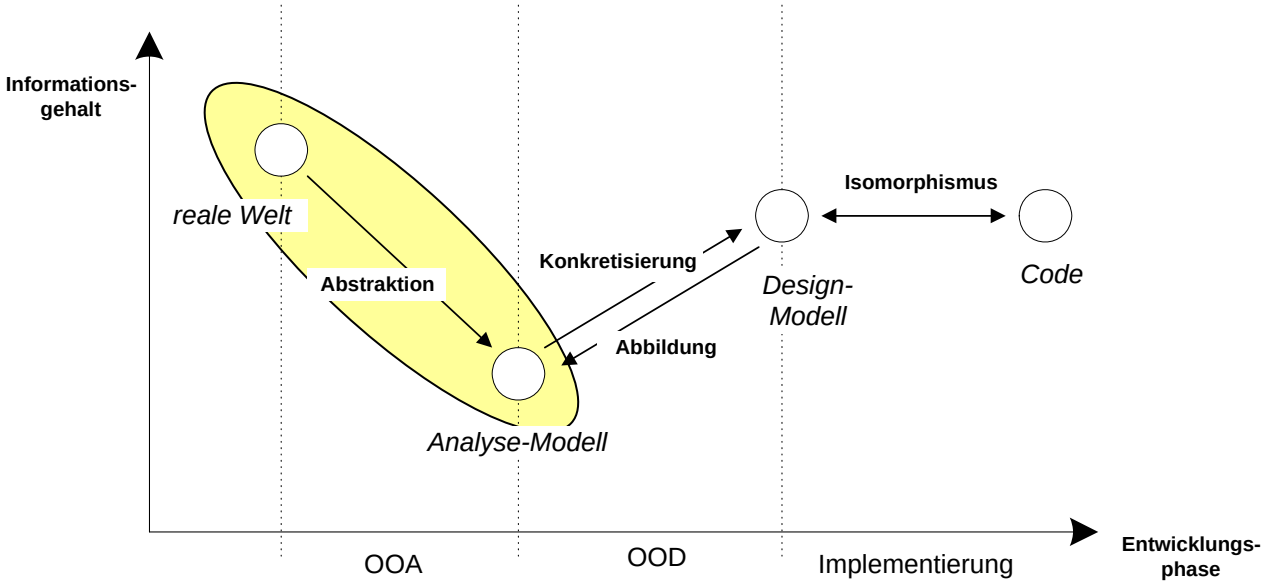
- Darstellung
  - eines einzelnen Use Cases
  - eines Gesamtsystems mit Use Cases
  - mit UML 2.0

- Regeln und Tipps
  - Aufteilung der Use Cases (12 Schritte)
  - Beschreibung einzelner Use Cases (Geschichte, Grammatik)

⇨ Bei allen Regeln und Tipps – das Finden und Schreiben von Use Cases bleibt ein iterativer Prozess

⇨ Versuchen Sie nicht, „perfekte“ Use Cases zu schreiben, bevor das Gesamtkonzept stabil erscheint

Objektorientierte Analyse



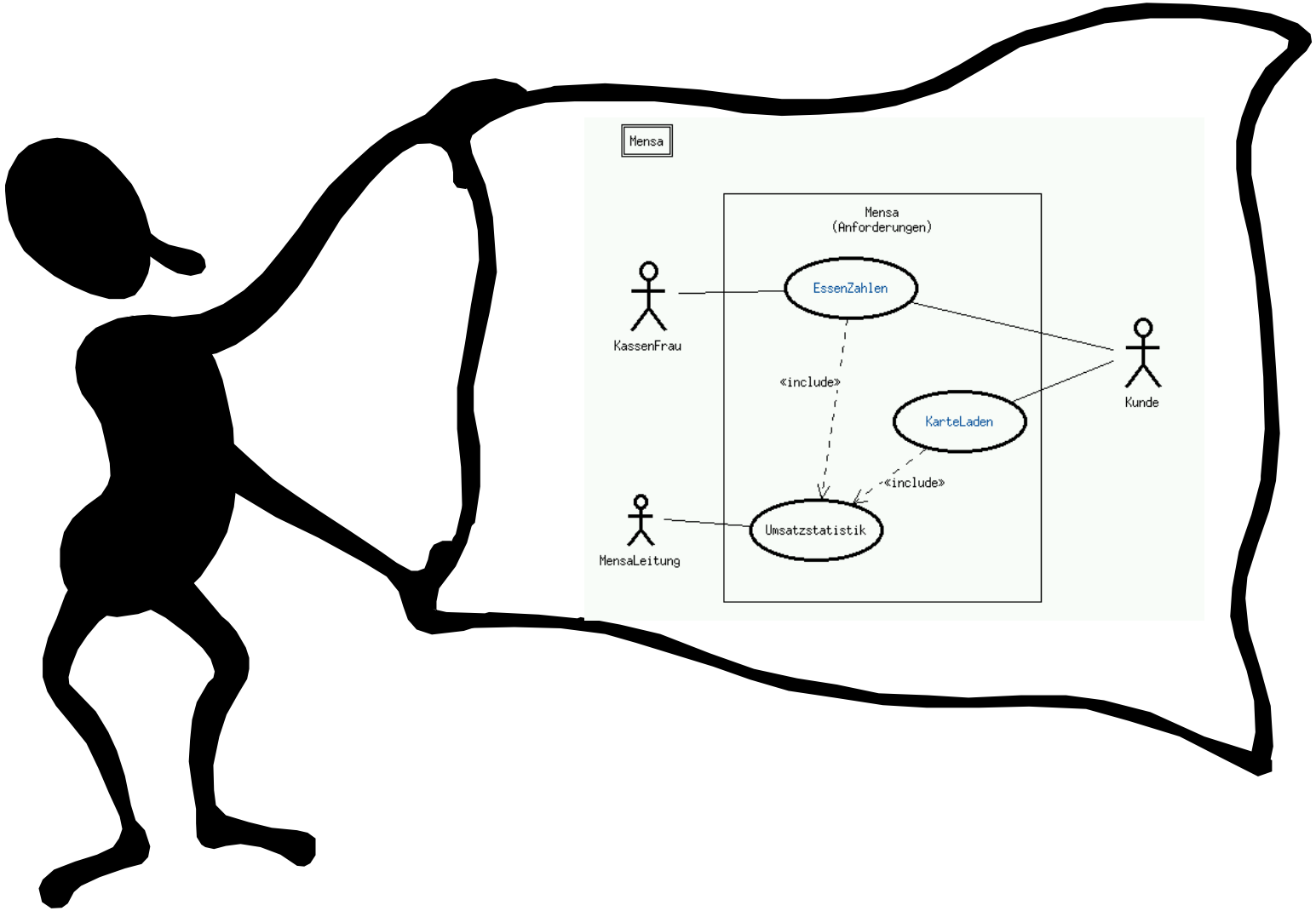
funktionale Anforderungen:
Beschreibung mit Use Cases

nicht-funktionale Anforderungen:
Beschreibung als Text

⇨Input für "Pflichtenheft"
für die Entwicklung

Basis für Verhandlung mit
Auftraggeber (gefiltert)

Fragen


Mensa
Mensa
(Anforderungen)
EssenZahlen
KassenFrau
«include»
KarteLaden
Kunde
«include»
Umsatzstatistik
MensaLeitung