Hochschule
Bonn-Rhein-Sieg

Software Engineering I

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen

Vorlesung für den Studiengang Bachelor Wirtschaftsinformatik

Wintersemester 2017 / 2018

Prof. Dr. Sascha Alda
(sascha.alda@h-brs.de)

Aufbau der Vorlesung

Kapitel	Thema	
1	Einführung ins Software Engineering	✓
2	Software-Prozessmodelle im Software Engineering	✓
3	**Modellierung und Erhebung von Anforderungen**	
4	Objektorientierte Analyse von Anforderungen	
5	System-Design (Grundlagen von Software-Architekturen)	
6	Objektorientiertes Design (Grundlagen und Entwurfsmuster)	
7	Testgetriebene Entwicklung mit JUnit	
8	Software-Wartung (Refactoring von Source Code)	
	Gastvortrag Alireza Farnoudi, FA. Congstar, Köln „Software-Entwicklung mit Scrum - ein Erfahrungsbericht“ Januar 2018.	

Anmerkung: ein Kapitel erstreckt sich über 1-2 Vorlesungen

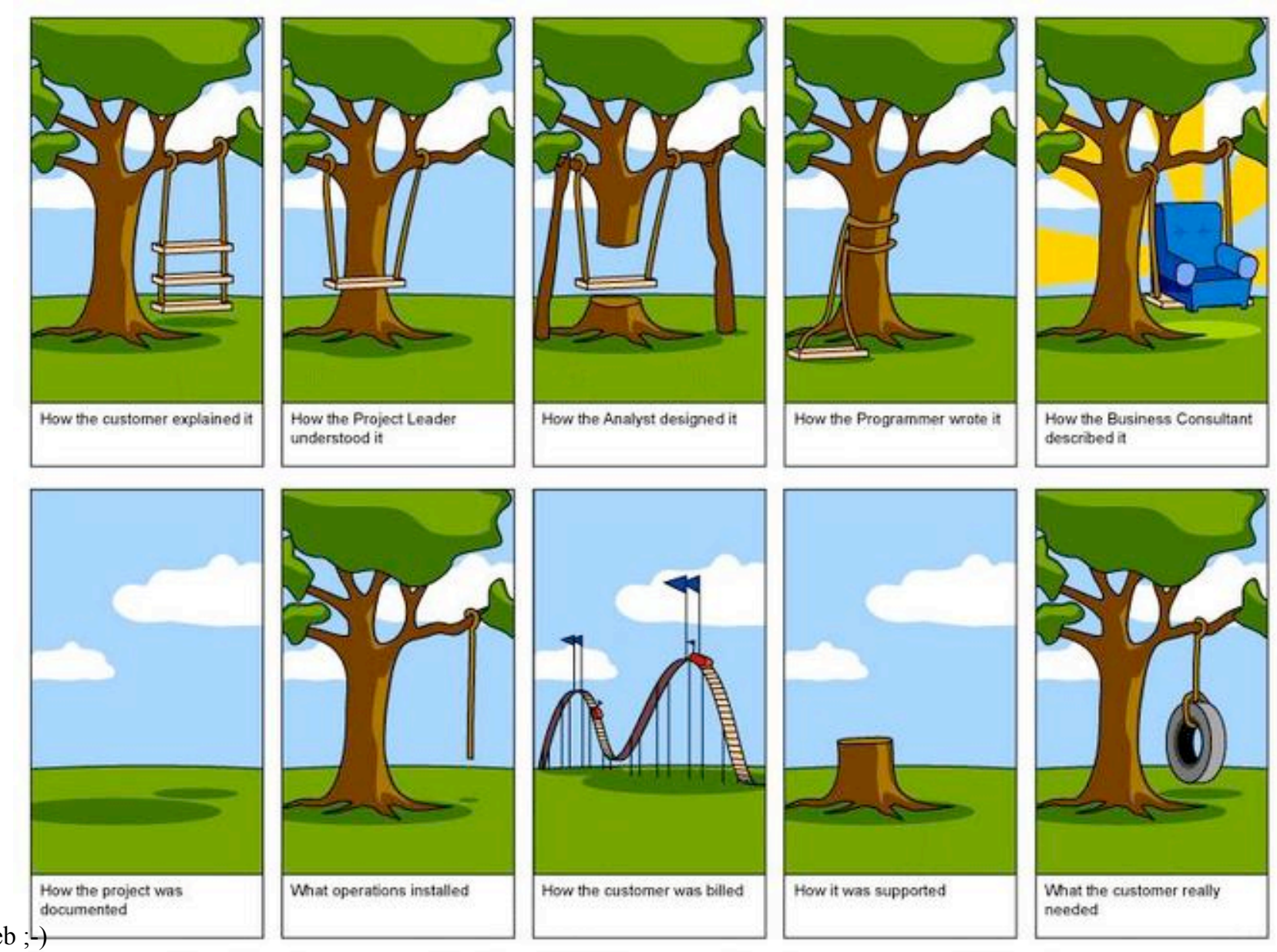
Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	**Allgemeines über Anforderungen**	
2	Agile Erhebung von Anforderungen mit User Stories	
3	Objektorientierte Erhebung von Anforderungen mit Use Cases	
4	Gegenüberstellung der Ansätze	
5	Zusammenfassung und Ausblick	

Erhebung Anforderungen

- Die Erhebung von Anforderungen stellt die erste Phase eines Software-Lebenszyklus dar
  - Evolutionäre oder agile Prozesse: Startaktivität einer jeden Iteration
- Ziel: Produktion einer Anforderungsspezifikation ausgehend von informalen Ideen eines Software-Systems (z.B. aus Lastenheft)
- Anforderungen stellen dar, was das Software-System tun soll, und nicht wie es funktionieren soll
- Das Gebiet des Anforderungsmanagement (*requirements engineering*) ist ein eigenes Teilgebiet des Software Engineering

Stakeholder, Phasen und „Leiden“ im Anforderungsmanagement



Quelle: Web ;-)

Der Begriff Anforderungen

- In der Literatur gibt es unterschiedliche Definitionen und Interpretation, was eine Anforderung ist.
- Übliche Definitionen:
  - Bedingung oder Fertigkeit, die ein Software-System zu erfüllen hat (nach (Sommerville, 2007)).
  - Eine vom Benutzer gestellte Bedingung (*condition*) oder Fertigkeit (*capability*) an eine Software, um ein Problem aus einer Anwendungs-domäne zu lösen (nach (Brügge and Dutoit, 2013)).
  - Anforderungen beschreiben, *was* ein zukünftiges System leisten muss, um ein geschäftliches oder betriebliches Ziel (Mehrwert) für den Kunden zu erreichen (Rupp, 2009).
- Es gibt verschiedene Kategorien von Anforderungen (nächste Folien)

Funktionale Anforderungen

- Funktionale Anforderungen beschreiben die eigentliche Funktionalität einer zu entwickelnden Software
  - Meist beschrieben durch die notwendigen Interaktionen zwischen dem System und seiner Umgebung (z.B. Endanwender, Drittsysteme)
  - Nicht-technische Beschreibung, also unabhängig von einer Implementierung
- Beispiele:
  - „Ein Student soll in der Lage sein, Räume über das Buchungssystem zu reservieren.“
  - „Ein Dozent soll in der Lage sein, eine Reise über das Buchungssystem zu buchen. Eine Buchungsbestätigung soll über eine Email erfolgen“

Nicht-funktionale Anforderungen

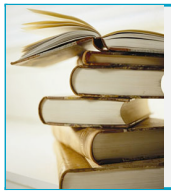
- Nicht-funktionale Anforderungen legen Qualitätsaspekte eines Ziel-Systems fest, die nicht im Zusammenhang mit der Funktionalität des Systems stehen
- Mögliche Aspekte: Performanz, Bedienbarkeit, Sicherheit
- Aktuellerer, besserer Begriff: Qualitätsanforderungen
- Beispiel:
  - „Eine Raumreservierung muss schnell abgespeichert werden“
  - „Eine Raumreservierung soll binnen 2 Sekunden ausgeführt sein“
  - „Das zu entwickelnde System soll sicher sein“
  - „Das Software-System soll einfach zu bedienen sein“

Sind alle nicht-funktionalen Anforderungen gut formuliert?

Weitere Anforderungskategorien

- Technische Anforderungen beschreiben technische Eigenschaften oder Voraussetzungen bezüglich des Entwurfs der Software
  - Beschreibung im technischen Jargon
  - Anforderungen bezüglich Architektur, Hardware, Schnittstellen usw.
  - Beispiel: „Die Abspeicherung von Buchungen soll in einer MySQL-Datenbank erfolgen“

- Rahmenbedingungen definieren zusätzliche Anforderungen bezüglich des Kontext, in der die Software entwickelt wird bzw. betrieben wird
  - Meist organisatorische oder juristische Vorgaben
  - Beispiel: Einhaltung von SO-X (Sarbanes-Oxley Act) bei börsennotierten Unternehmen
  - Einhaltung DIN/ISO Normen z.B. für die Einhaltung von Qualitätsstandards

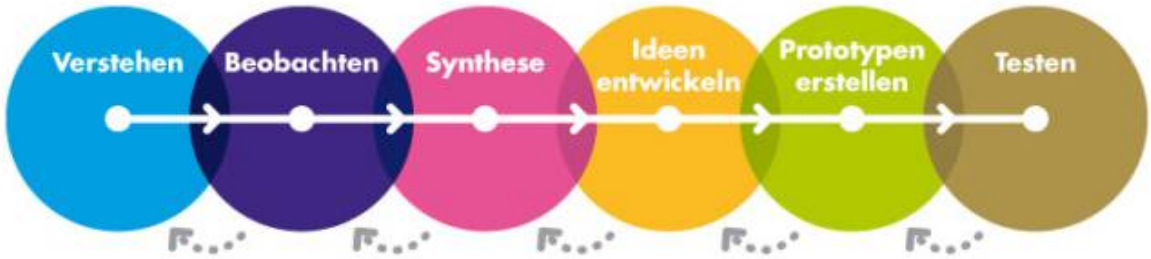


Weitere Anforderungskategorien und –Artefakte werden in der Vorlesung SE II weiter vertieft

Wie erhält man Anforderungen?

- Kreativitätstechniken wie in den Übungen erläutert
  - Mind-Map, Brainstorming, 635-Methode, Morphologische Methoden, u.v.a.
- Interviews mit relevanten Stakeholdern
- Beobachtungen von Endanwendern bei der Verrichtung ihrer Arbeit
  - Ableiten von Anforderungen aus der IST-Situation
- Workshops mit allen Beteiligten („Think Tank“)
- Aktueller Ansatz, der verschiedene Kreativitätstechniken und weitere Methoden als interdisziplinären Prozess versteht: Design Thinking (Grots and Paschke, 2009)

Methoden zur Erhebung von Anforderungen werden in SE II vertieft!



Fokus in dieser Vorlesung: Funktionale Anforderungen

- Der Fokus in dieser Vorlesung liegt in der Modellierung von funktionalen Anforderungen

- Für die Darstellung von funktionalen Anforderungen werden hier zwei Varianten präsentiert:
  - Modellierung durch User Stories (bekannt aus Scrum, XP)
  - Modellierung durch Use Cases (objektorientierter Ansatz nach (Brügge and Dutoit, 2013))

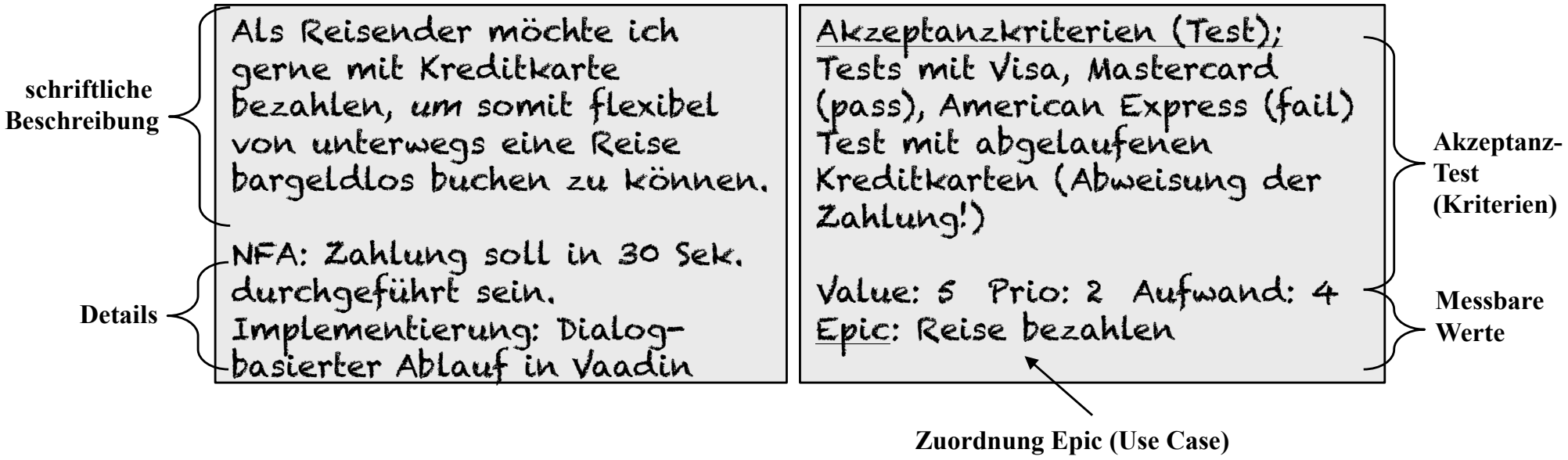
- Es wird später gezeigt, wie sich beide Ansätze ergänzen können

Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	**Agile Erhebung von Anforderungen mit User Stories**	
	2.1 – Definition und Einordnung	
	2.2 – Die INVEST Merkmale	
	2.3 – Priorisierung von User Stories	
	2.4 – Ableitung von Tasks	
3	Objektorientierte Erhebung von Anforderungen mit Use Cases	
4	Gegenüberstellung der Ansätze	
5	Zusammenfassung und Ausblick	

User Story – eine Definition

- Eine User Story beschreibt eine Funktionalität, die einen Mehrwert für einen Endanwender oder den Käufer (Kunden) einer Software bringt (Cohn, 2004)
- Eine User Story besteht aus drei Teilen:
  - Einer schriftlichen Beschreibung der Story, aus der der Mehrwert klar wird
  - Weitere (wenige) Details über eine Story
  - Einen Akzeptanztest (Akzeptanzkriterien) für die Story
- Meist noch ergänzt um messbare Werte sowie Zuordnung zu einem Epic (Use Case)



Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	**Agile Erhebung von Anforderungen mit User Stories**	
	2.1 – Definition und Einordnung	✓
	2.2 – Die INVEST Merkmale	
	2.3 – Priorisierung von User Stories	
	2.4 – Ableitung von Tasks	
3	Objektorientierte Erhebung von Anforderungen mit Use Cases	
4	Gegenüberstellung der Ansätze	
5	Zusammenfassung und Ausblick	

Entwicklung von User Stories – INVEST Merkmale

- Eine gute User Story sollte die INVEST Merkmale erfüllen:
  - **I** ndependent (unabhängig)
  - **N** egotiable (verhandelbar)
  - **V** aluable to users or customers (bringt einen Mehrwert)
  - **E** stimatable (schätzbar)
  - **S** mall (klein)
  - **T** estable (testbar)
  - = I N V E S T (Cohn, 2004)

- Die einzelnen Merkmale werden auf den folgenden Folien einzeln dargestellt

Entwicklung von User Stories – *I* N V E S T Merkmale

- User Stories sollten keine Abhängigkeiten untereinander aufweisen
- Folgerung: User Stories können unabhängig voneinander entwickelt werden
- Vorteil: bessere Planung, Priorisierung und Tracking (Controlling) ist möglich

- Beispiele:

Als Student möchte ich im Buchungssystem eine Reise mittels der Kreditkarte Visa bezahlen.

Als Student möchte ich im Buchungssystem eine Reise mittels der Kreditkarte Mastercard bezahlen.

<<depends on >>

- In Praxis sind Abhängigkeiten aber durchaus üblich:

Als Usability-Experte möchte ich einen komponentenbasierten sowie einen dialogbasierten Bezahlprozess bereitstellen, um neue Bezahlmethoden flexibel zu integrieren

Entwicklung von User Stories – I *N* V E S T Merkmale

- Die Inhalte von User Stories sollten zwischen Kunde, Endanwender und Entwickler verhandelbar (engl.: *negotiable*) sein.
- Eine User Story ist also keine schriftlich fixierte Spezifikation
- Details müssen (!) im Laufe der Entwicklung (z.B. innerhalb einer Iteration oder in den Planning Meetings) ergänzt werden (User Story = Reminder)
- Voraussetzung : Kunde ist stets erreichbar für das Entwicklerteam
- Beispiele:

> Als Student möchte ich im Buchungssystem eine Reise mittels der Kreditkarte Visa bezahlen.

Detailfragen wie:
- Wie ist der genaue Ablauf während der Bezahlung?
- Wie sieht die Benutzeroberfläche aus?
- Wie werden Fehlerfälle behandelt?

können / müssen später erläutert werden!

Entwicklung von User Stories – I N V E S T Merkmale

- Eine User Story muss dem Kunden oder dem Endanwender einen Mehrwert (Nutzen, Vorteil, Gewinn, engl.: *value*) bringen
- Praktischer Mehrwert (Endanwender), finanzieller Mehrwert (Kunde)
- Der Mehrwert sollte auf der User Story schriftlich ausgedrückt werden.
- Typische Struktur zur schriftlichen Fixierung (Cohn, 2004):

Als Anwender mit der Rolle

benötige ich eine Funktionalität

um einen **Mehrwert** zu bekommen

Struktur zur Beschreibung einer User Story (Synonyme für Begriffe üblich)

Als Viel-Reisender möchte ich gerne eine Liste aller meiner gebuchten Reisen stets abrufen können, *damit* ich einen Überblick über zukünftige Reisen habe. Diese Funktionalität würde meine Produktivität erhöhen und meine Arbeitsplanung verbessern.

Als Kunde mit der Rolle des Abteilungsleiters „Debitoren“ möchte ich den Bezahldienst PayPal anbinden, um eine kostengünstige Alternative zur Kreditkartenbezahlung anzubieten.

Beispiel einer User Story mit Darstellung des Mehrwerts

Entwicklung von User Stories – I N V E S T Merkmale

- Der Mehrwert (*value*) einer User Story sollte für eine spätere Bewertung (z.B. Priorisierung) quantifiziert werden.

- Typisches Schema nach (Wiegers, 1999): Skala mit *relativen* Werten 1-5
  - 1 – User Story bringt den geringsten Mehrwert
  - 5 – User Story bringt den meisten Mehrwert (häufig auch bis 9..)

- In der Industrie ist eine Skala von 1 – 5 Standard.

Als Besucher des Portals möchte ich regelmäßig über neue Reiseziele Informationen bekommen, damit ich zukünftige Urlaube frühzeitig buchen kann. Value: 2	Als Reisender möchte ich die Möglichkeit haben, meine Reise mittels einer MasterCard zu bezahlen, was meine Liquidität vorübergehend sichert. Value: 5

Beispiele einer User Story mit Darstellung des Mehrwerts (Skala 1-5)

Entwicklung von User Stories – I N V E S T Merkmale

- Den Aufwand zur Umsetzung einer User Story sollte durch das Entwickler-Team geschätzt (engl.: *estimate*) werden können
  - Basis für die Planung und Priorisierung
  - Intuitive Schätzung rein auf fachliche Beschreibung der User Story!

- Typische Probleme beim Schätzen von User Stories:
  - Fehlendes Fachwissen oder technisches Know-How

- Taktiken um diese Probleme zu umgehen:
  - Fehlendes Fachwissen: Diskussion in Meetings oder innerhalb der Iteration
  - Fehlendes Technisches Wissen: Aneignung des Wissens durch „Spike“ Story für eine bessere Schätzung in dem nächsten Sprint (Bergsmann, 2014) (siehe auch noch mal SE-2)

> Als leitender Software-Architekt möchte ich zur Umsetzung der Kreditkartenbezahlung mögliche technische Anbindungen an Kreditinstitute analysieren, um in den folgenden Springs das Know-How verwerten zu können.

Beispiel einer Spike Story als Ergänzung zur normalen User Story „Kreditkartenbuchung“

Entwicklung von User Stories – I N V E S T Merkmale

- Der Aufwand einer User Story wird in Story Points ausgedrückt
- Ein Story Point hat keinen Default-Referenzwert und keine vergleichbare Einheit, muss von einem Team individuell bestimmt werden.
- Empfehlungen für Referenzwert: Ein Story Point ist…
  - … ein *idealer* Arbeitstag (Gloger, 2008 u.a.)
  - … eine Aufgabe mit einer bestimmten Komplexität (Capgemini, (Rau, 2016))
  - Einordnung nach Fibonacci-Reihe (Capgemini, (Gloger 2008), u.a.)

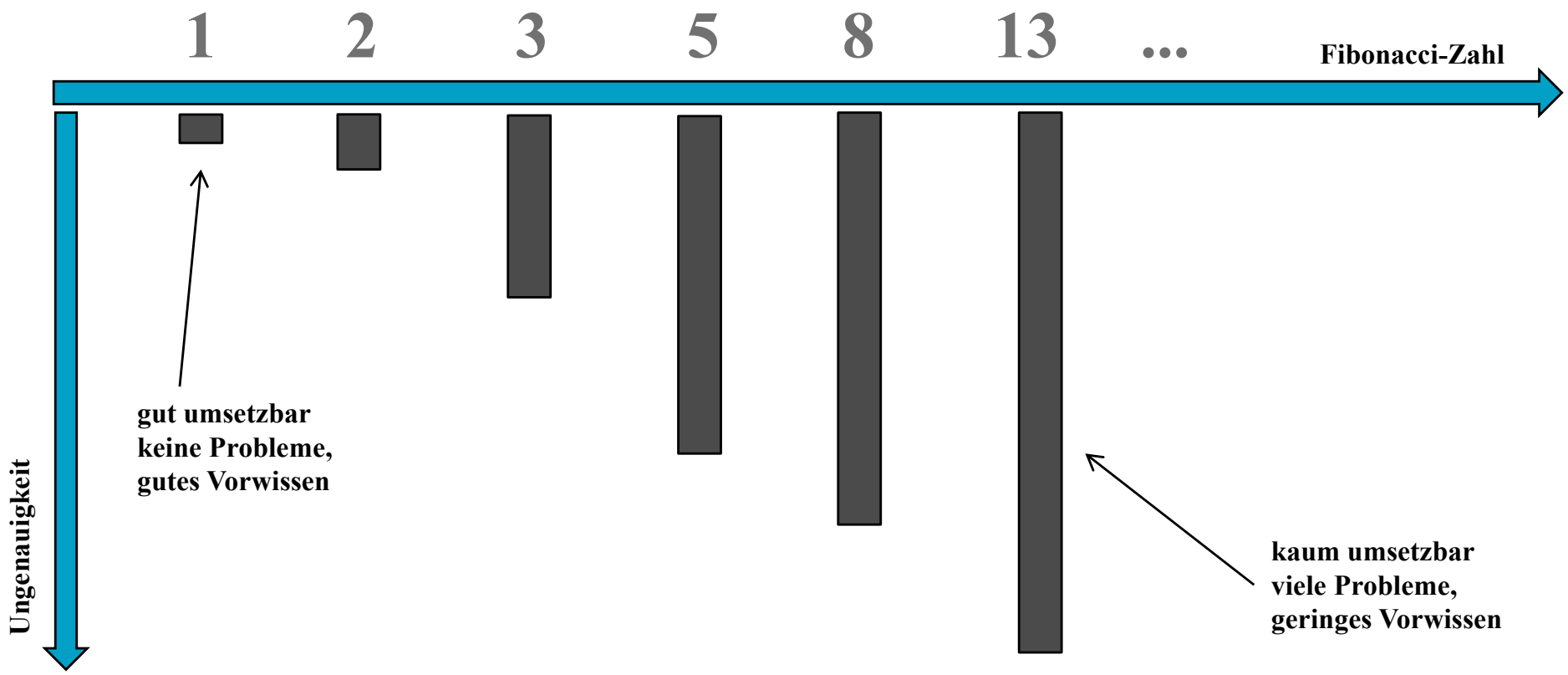
Als Besucher des Portals möchte ich regelmäßig über neue Reiseziele Informationen bekommen, damit ich zukünftige Urlaube zügig vorbereiten kann. Value: 2 Aufwand (SP): 2	Als Reisender möchte ich die Möglichkeit haben, meine Reise mittels einer MasterCard zu bezahlen, was meine Liquidität vorübergehend sichert. Value: 5 Aufwand (SP): 5

Beispiele einer User Story mit Darstellung der Aufwände der Story Points

Story Points nach der Fibonacci-Reihe

- Die Fibonacci-Reihe definiert (Fibonacci-)Zahlen als Summe der zwei vorangegangen Zahlen: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

- Die Schätzung einer User Story mit Story Points gemäß Fibonacci drückt eine Unschärfe / Ungenauigkeit in der Schätzung aus:



Schätzung von User Stories mit Planning Poker (Gloger, 2008)

- Ansatz: in der strategischen Planungsphase (ggf. auch in Planning Meetings) werden alle relevanten User Stories geschätzt. Typischer Prozess:

- Der Product Owner stellt die User Story vor.
- Das Team diskutiert die User Story mit dem Product Owner
- Alle Team-Mitglieder schätzen simultan den Aufwand der User Story:
- Varianten:
  - Mit Fingern. Skala: 1 (niedrig) bis 5 (hoch)
  - Mit Karten (Fibonacci-Reihe)
- Die Bewertungen mit dem niedrigsten und dem höchsten Wert werden vom jeweiligen Teammitglied erläutert.
- Finale Festlegung nach Diskussion

- Vorteil: intuitive Schätzung ohne Gruppendruck

Quelle Foto:
http://www.crisp.se/bocker-och-produkter/planning-poker

Entwicklung von User Stories – I N V E S T Merkmale

- Ansatz bei Scrum: das Entwicklerteam bestimmt selbst, wie *schnell* es arbeiten kann und somit wie seine Kapazitäten für eine Iteration sind

- Die Kapazität eines Teams wird als Velocity bezeichnet (dt.: *Geschwindigkeit*)

- Die für einen Sprint angenommene Velocity kann auf Basis ...
  - von vergangener Iterationen (Erfahrungswerten, tatsächliche Velocity)
  - den Komplexitäten der anstehenden Arbeiten
  - der aktuellen Team-Zusammenstellung
  - Rahmenbedingungen (z.B. Termine, Urlaub, *Karneval (!)*...)

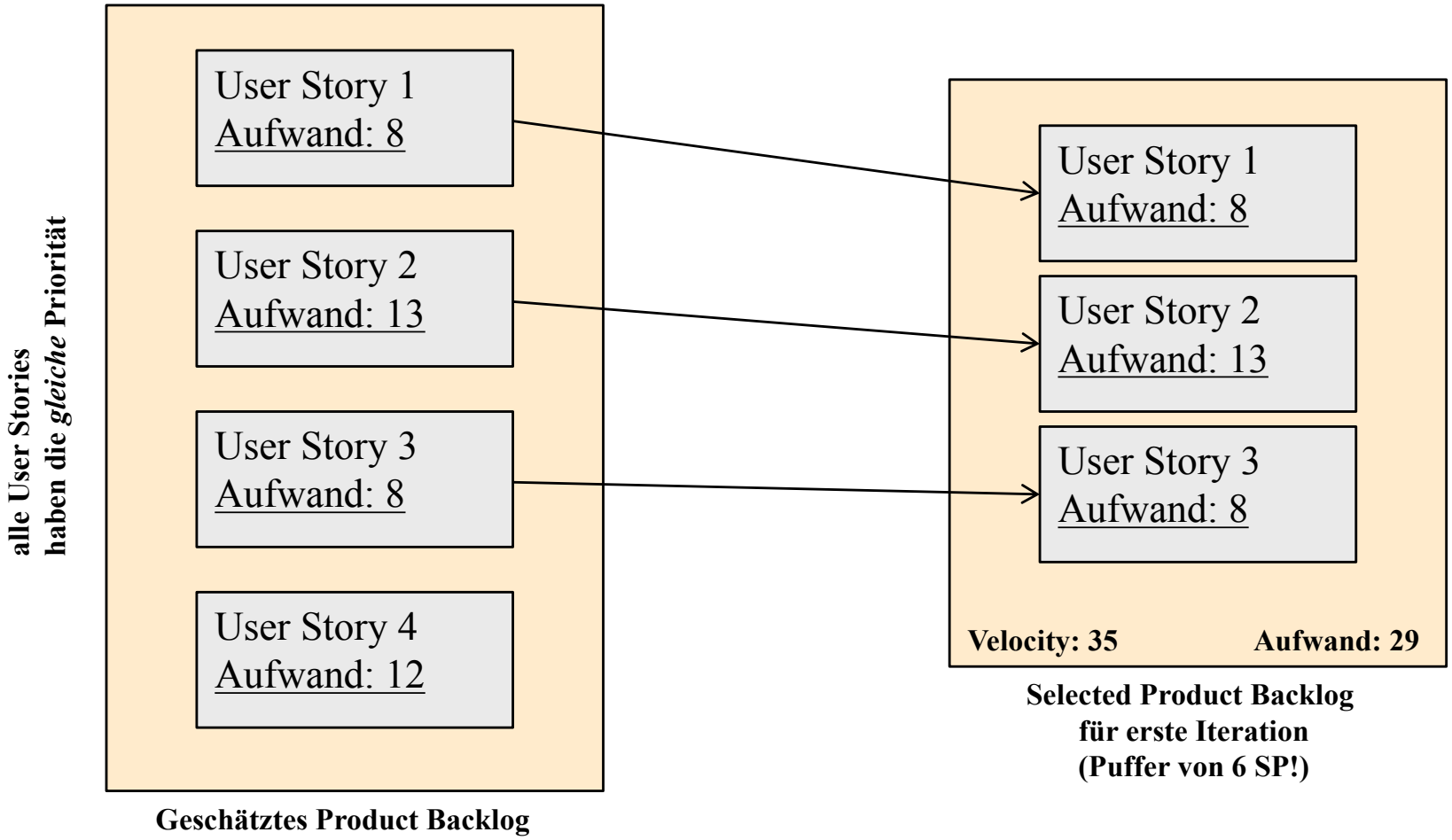
  ... geschätzt werden.

- Gerade in den ersten Sprints sollte die Velocity eher niedrig eingeschätzt werden (Scrum-Team sollte nicht überfordert werden)

- Anhand der Velocity erfolgt eine Selektion von User Stories für einen Sprint.
- Best Practice: zu Beginn eher weniger User Stories selektieren, Puffer lassen

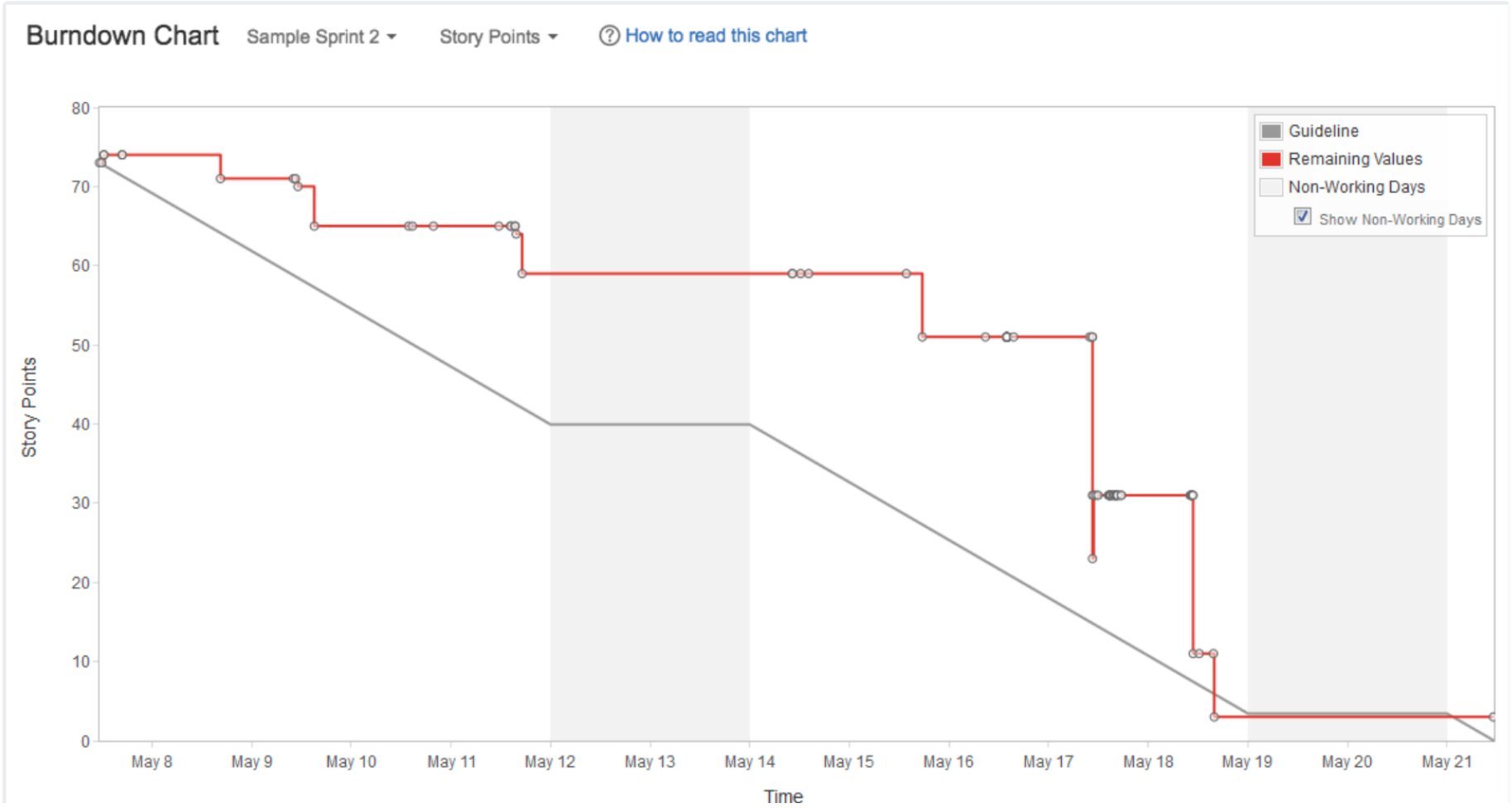
Entwicklung von User Stories – I N V E S T Merkmale

- Beispiel: Ein Entwickler-Team legt auf Basis von Erfahrungswerten eine Velocity von 35 SP fest. Eine Zuteilung von User Stories aus dem geschätzten Produkt Backlog für den Sprint kann unmittelbar erfolgen:



Burn Down Chart

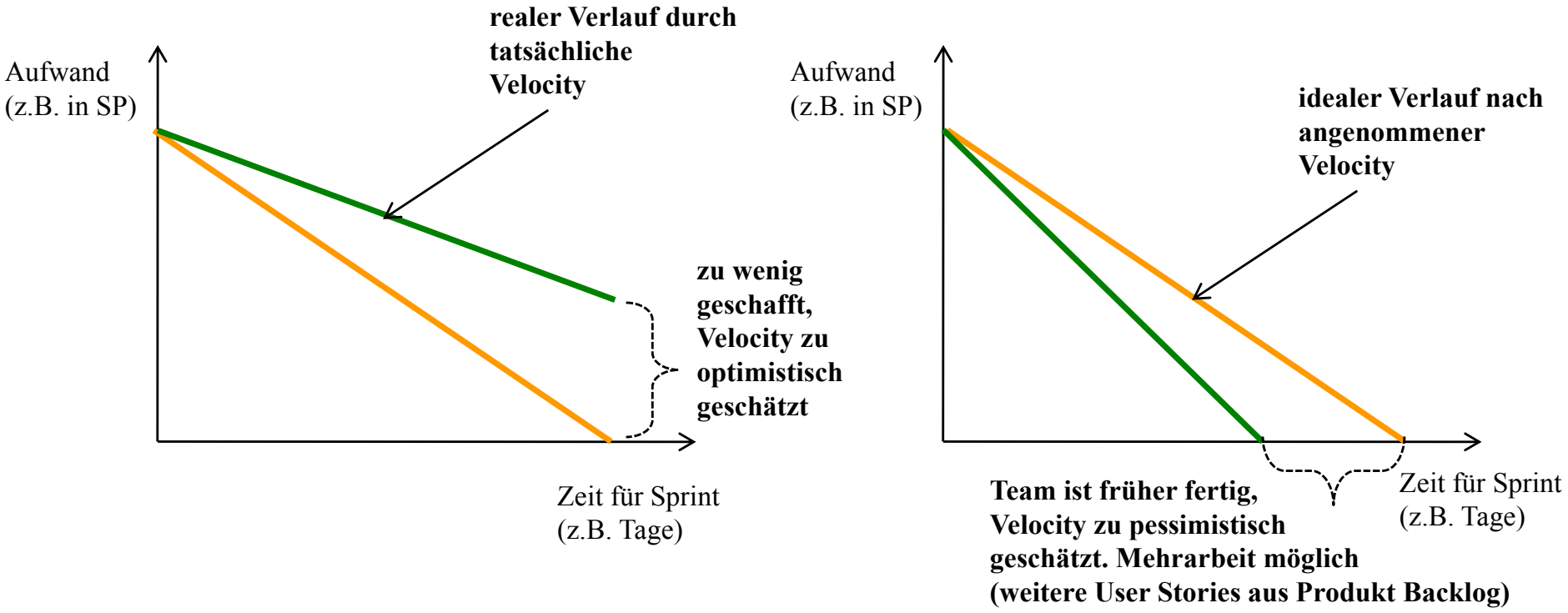
- Ein Burn Down Chart ist eine graphische Repräsentation der Arbeit, die es noch zu erledigen gilt vs. der Zeit, die noch zu Verfügung steht. Snapshot aus JIRA:



Quelle: https://confluence.atlassian.com/agile/jira-agile-user-s-guide/using-a-board/using-reports/viewing-the-burndown-chart

Analyse der Velocity anhand der Burn Down Chart

- Die Anzahl der abgeschlossenen Tasks bzw. User Stories beeinflusst die *tatsächliche* Velocity für einen Sprint
  - Regel: nur wenn eine User Story komplett abgeschlossen ist, wird sie in die Berechnung der tatsächlichen Velocity berücksichtigt.
  - Falls nicht: Übernahme in den nächsten Sprint
- Beobachtung: tatsächliche Velocity zu Beginn eher noch vage, später konstant. Mögliche Fälle:



Entwicklung von User Stories – I N V E S T Merkmale

- Die User Stories sollten kurz und prägnant (engl.: *small*) beschrieben sein.
- .... aber auch nicht zu klein:

> Die Hintergrundfarbe der Webpage soll grün sein.

- Daumenregel: Eine zusammenhängende, abgeschlossene Funktionalität an das System, die in einer Iteration umgesetzt werden kann.
- Umfangreiche User Stories sollten zerlegt werden:

> Ein Reisender soll in der Lage sein, ein Hotel umzubuchen. Des Weiteren soll ein Reisender in der Lage sein, ein Hotel zu stornieren. Ein Manager des Unternehmens kann sich eine Liste aller aktuellen Stornierungen anschauen, um sich einen aktuellen Überblick zu verschaffen.

Wie kann man diese zerlegen?

- Sehr umfangreiche User Stories (*Epics*) können als High-Level-Anforderung zur besseren Orientierung behalten werden (siehe Beispiel im Abschnitt 4)

User Stories – Heuristiken zum Schreiben von User Stories

- Aktive Sprache verwenden, aus Sichtweise der betroffenen Person oder des Systems formulieren
- Verben verwenden um Aktivitäten anzudeuten
- Ganze Sätze mit Begriffen aus der Fachdomäne (kein technisches Jargon)
- Keine Bandwurmsätze (und, und...)

aktiv formuliert aus Sicht des Studenten

Gutes Jargon aus der Domäne (Prüfungsphase, Semester)

Als Student möchte ich mit Hilfe des Studenten-Systems eLookBook Räume anhand eines digitalen Raumplans für die alle Prüfungsphasen des Semesters reservieren können, damit ich meinen Lernprozess besser steuern kann

Zudem soll er Buchungen einsehen, stornieren, delegieren und korrigieren können

Aktivität mit Verb „reservieren“ angedeutet

Zu viele Anforderungen verbunden Trennen in separate User Stories!

X

Entwicklung von User Stories – I N V E S T Merkmale

- Eine User Story sollten so geschrieben sein, dass sie testbar ist.
- Beweis dafür, dass eine User Story erfolgreich umgesetzt wurde
- Zusätzliche Hilfestellungen (Details) für den Entwickler
- Angabe eines Akzeptanztests muss bei der Story durch den Kunden erfolgen
- Testfälle sind nicht immer trivial zu bestimmen. Beispiel:

Der Studierende sollte in der Lage sein, ein anonymes Feedback zu einer Webseite abzugeben.

Wie kann man auf Anonymität testen?

Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	**Agile Erhebung von Anforderungen mit User Stories**	
	2.1 – Definition und Einordnung	✓
	2.2 – Die INVEST Merkmale	✓
	2.3 – Priorisierung von User Stories	
	2.4 – Ableitung von Tasks	
3	Objektorientierte Erhebung von Anforderungen mit Use Cases	
4	Gegenüberstellung der Ansätze	
5	Zusammenfassung und Ausblick	

Priorisierung von User Stories - MusCoW

- Bereits kennengelernt: direkte Zuordnung von User Stories auf Iterationen nur auf Basis der Aufwandsschätzung *ohne* Priorisierung

- Für eine Ersteinschätzung zur Priorisierung der User Stories eines Backlog Item kann man die „MusCoW“ Skala anwenden:

Parameter	Beschreibung
Must	In die Kategorie fallen die User Stories, ohne die die Software auf keinen Fall ausgeliefert oder vorgestellt werden kann Beispiel: sehr viele eingehende Abhängigkeiten!
Could	Alle User Stories, die man in der Software haben möchte, aber nicht zwingend nötig sind. Wenn sie fehlen, wird die Funktionalität des Systems eingeschränkt, was aber in Kauf genommen werden kann
Wish	„Nice-To-Have“ Eigenschaften, welche bei einem Fehlen, die Funktionalität der Software nicht einschränken würde.

- Priorisierung dann anhand einer Liste: je höher der Eintrag, desto wichtiger das Item!

Priorisierung von User Stories – Relatives Gewicht

- Für eine genauere Priorisierung müssen weitere Parameter in Betracht genommen werden (Gloger, 2008):

Parameter	Beschreibung	Skala
Relativer Mehrwert	Maß dafür, wie hoch der Vorteil, Nutzen oder Gewinn wäre	1 (niedrig)-5 (hoch)
Relative Strafe	Maß dafür, wie hoch die Strafe (Verlust) wäre, wenn eine User Story nicht umgesetzt würde	1 (niedrig)-5 (hoch)
Relatives Risiko	Maß dafür, ob die Umsetzung einer User Story mit einem Risiko behaftet ist (z.B. technische oder organisatorische Probleme absehbar)	1 (niedrig)-5 (hoch)
(Relativer) Aufwand	Maß dafür, wie hoch der Aufwand für die Umsetzung einer User Story ist	beliebig oder Skala

Formel zur Priorisierung von User Stories

- Für jede User Story *m* lässt sich die Priorisierung Prio_m wie folgt festlegen (Quelle, (Gloger, 2008), empfohlen auch von (Kruchten, 2011)):

$$\textbf{Prio}_m = \frac{\text{Mehrwert}_m + \text{Strafe}_m}{\text{Aufwand}_m + \text{Risiko}_m}$$

- Die User Story *m* mit dem höchsten Prio-Wert hat die höchste Priorität

Beispiel einer Priorisierung mit Relativen Gewichten

- Auf Basis der Schätzungen der Grund-Parameter Aufwand, Risiko, Mehrwert und Strafe kann eine exakte Priorisierung vorgenommen werden:

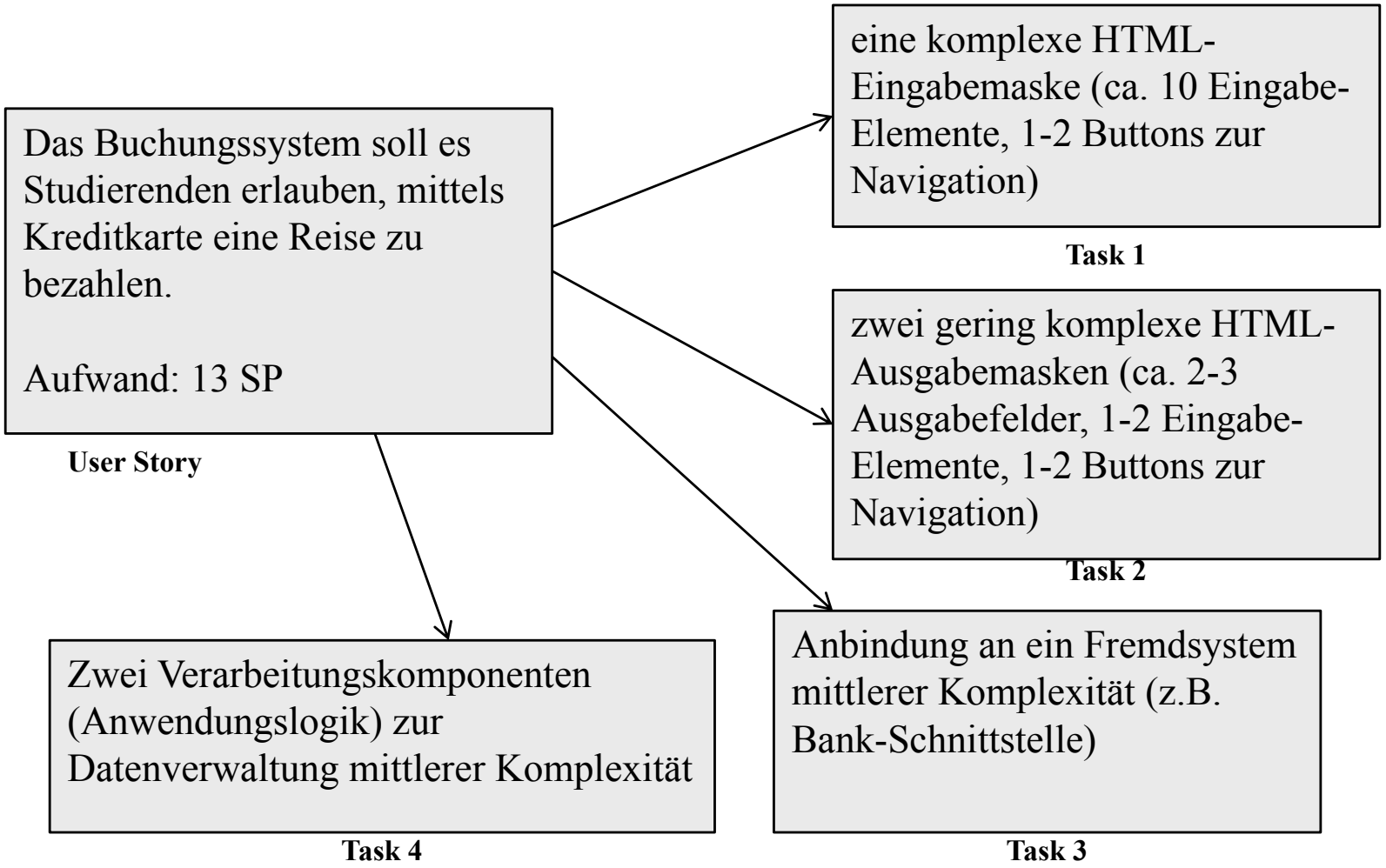
User Story	Aufwand	Mehrwert	Strafe	Risiko	$Prio_m$
User Story 1 (Kreditkarten-Bezahlung)	13	5	4	3	**0,56**
User Story 2 (Bearbeitung Benutzerprofil)	8	3	2	2	**0,5**
User Story 3 (Benachrichtigung neue Reisen)	5	2	1	2	**0,43**
User Story 4 (Termine verwalten)	5	2	2	1	**0,67**
Total	**31**	**12**	**9**	**8**	

Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	**Agile Erhebung von Anforderungen mit User Stories**	
	2.1 – Definition und Einordnung	✓
	2.2 – Die INVEST Merkmale	✓
	2.3 – Priorisierung von User Stories	✓
	2.4 – Ableitung von Tasks	
3	Objektorientierte Erhebung von Anforderungen mit Use Cases	
4	Gegenüberstellung der Ansätze	
5	Zusammenfassung und Ausblick	

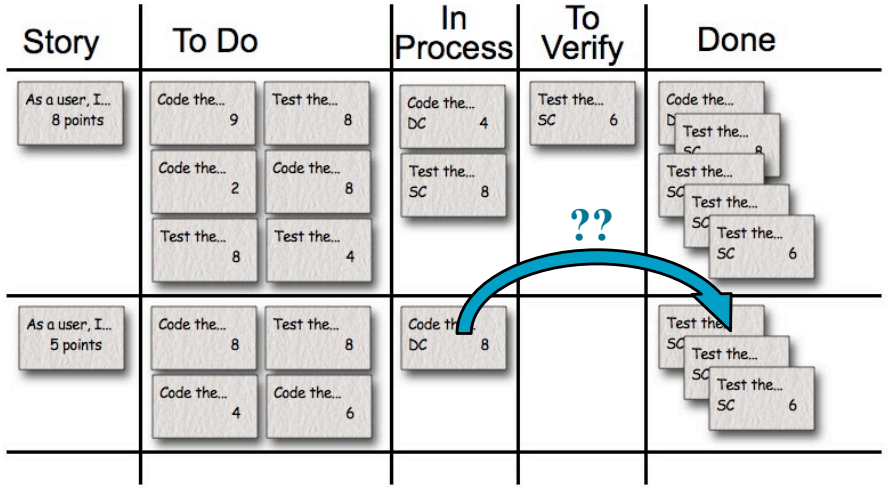
Unterteilung User Stories zu Tasks

- Beispiel: Bei der Umsetzung der User Story „Kredikartenbezahlung“ steht nach dem Planning Meeting 2 folgende Aufgaben fest:



Unterteilung User Stories zu Tasks - DoD

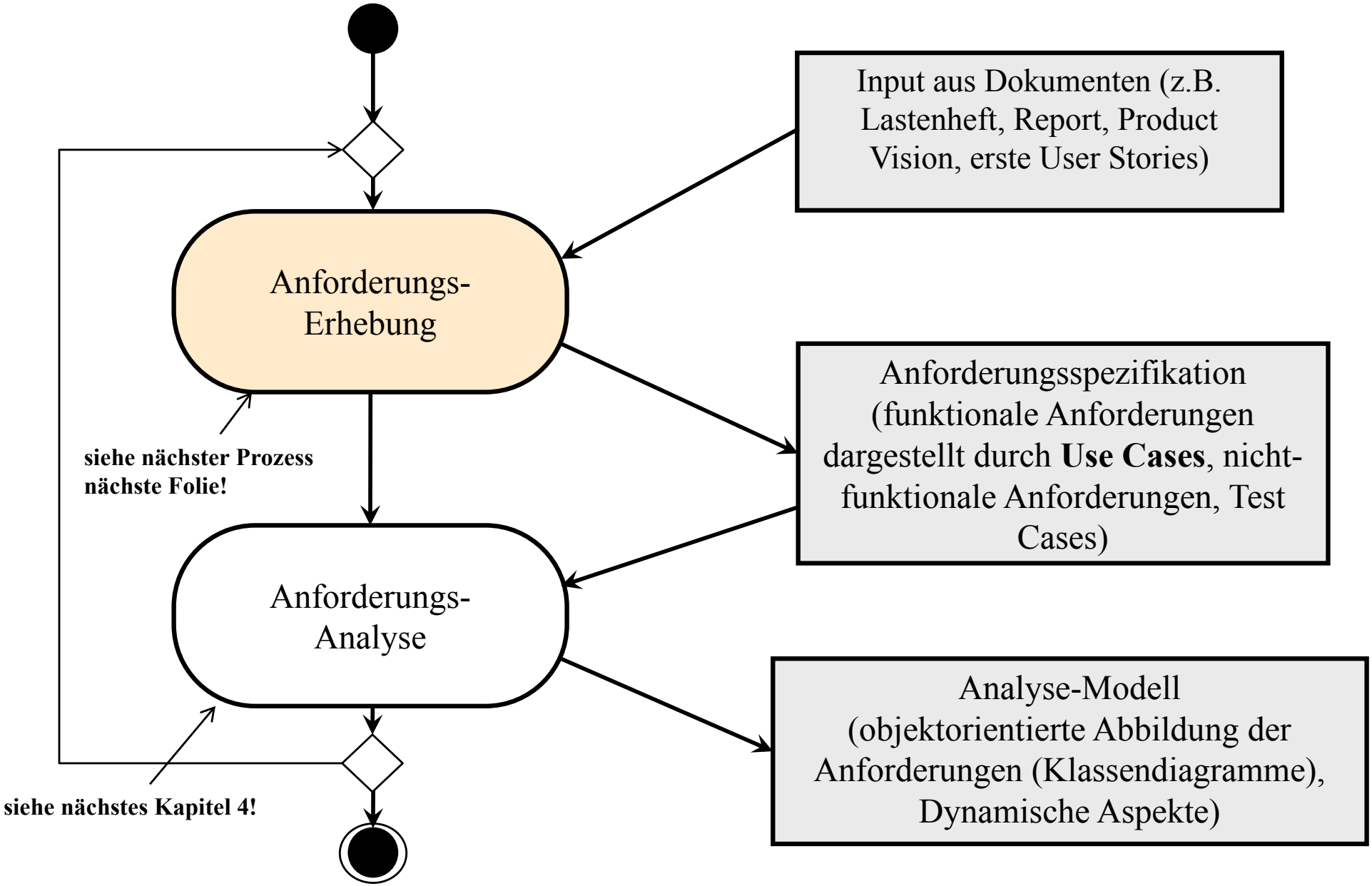
- Es gibt keine Regel oder Heuristik wie viele Tasks aus einer User Story abgeleitet werden sollen (weitere Infos dazu auch in SE-2)
- Verwaltung der User Stories und Tasks auf dem Scrum Board (Kapitel 2)
- Fragestellung: wann ist eine User Story oder ein Task erledigt?
- Wichtig: Bestimmung eines Definition of Done (DoD)! (→ Kapitel 2)
  - Beispiel: Alle Tests ok, Peer Review ok, alle Software-Metriken ok, u.a.



Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	Agile Erhebung von Anforderungen mit User Stories	✓
3	**Objektorientierte Erhebung von Anforderungen mit Use Cases**	
	3.1 – UML Use Cases	
	3.2 – Beziehungen zwischen UML Use Cases	
	3.3 – Ableitung von Test Cases	
4	Gegenüberstellung der Ansätze	
5	Zusammenfassung und Ausblick	

Objektorientierte Vorgehensweise zur Modellierung von Anforderungen (Bruegge und Dutoit, 2013)



Use Case Diagramme

- Zur graphischen Modellierung von Use Cases eines Software Systems stellt die UML Use Case Diagramme (Jacobsen, 1992) zur Verfügung.

- Use Case Diagramme stellen die funktionalen Anforderungen einer Software aus Sicht der Anwender dar.

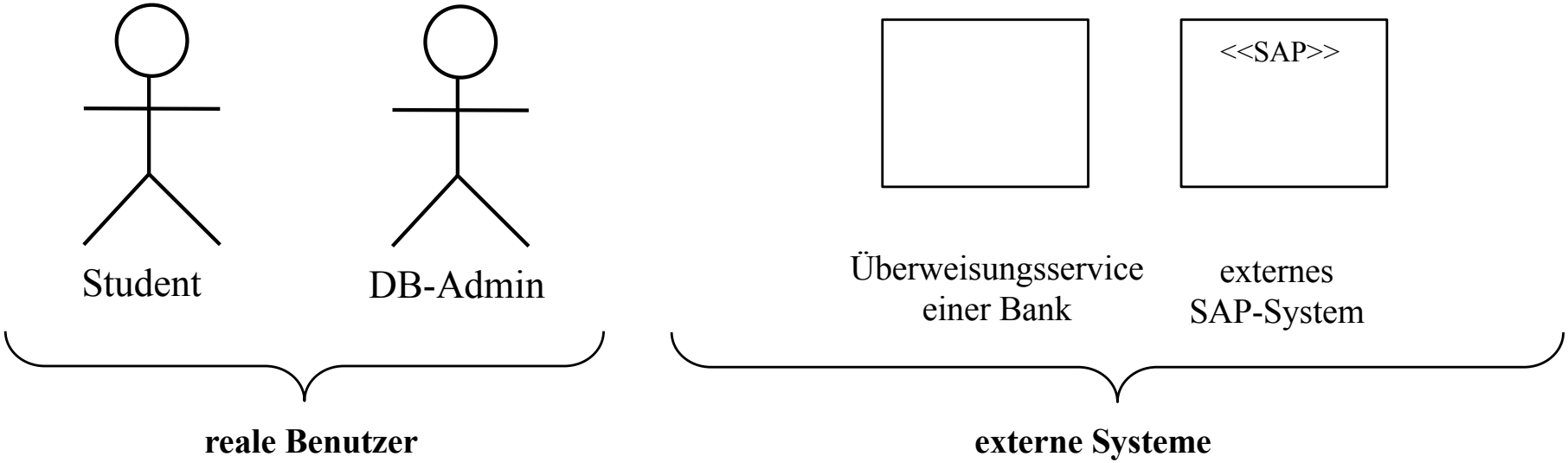
- Abstraktion von möglichen konkreten Szenarien von konkreten Benutzern (Bruegge und Dutoit, 2013)

- Fokus auf:
  - Interaktion: wie interagiert der Benutzer mit dem Software-System, um die Funktionalität des System zu verwenden
  - Fachliche Modellierung: keine technischen Details (z.B. abspeichern auf DB)

- Modellierung der Interaktionen des Software-Systems auch mit anderen externen Systemen, die außerhalb des Systems liegen

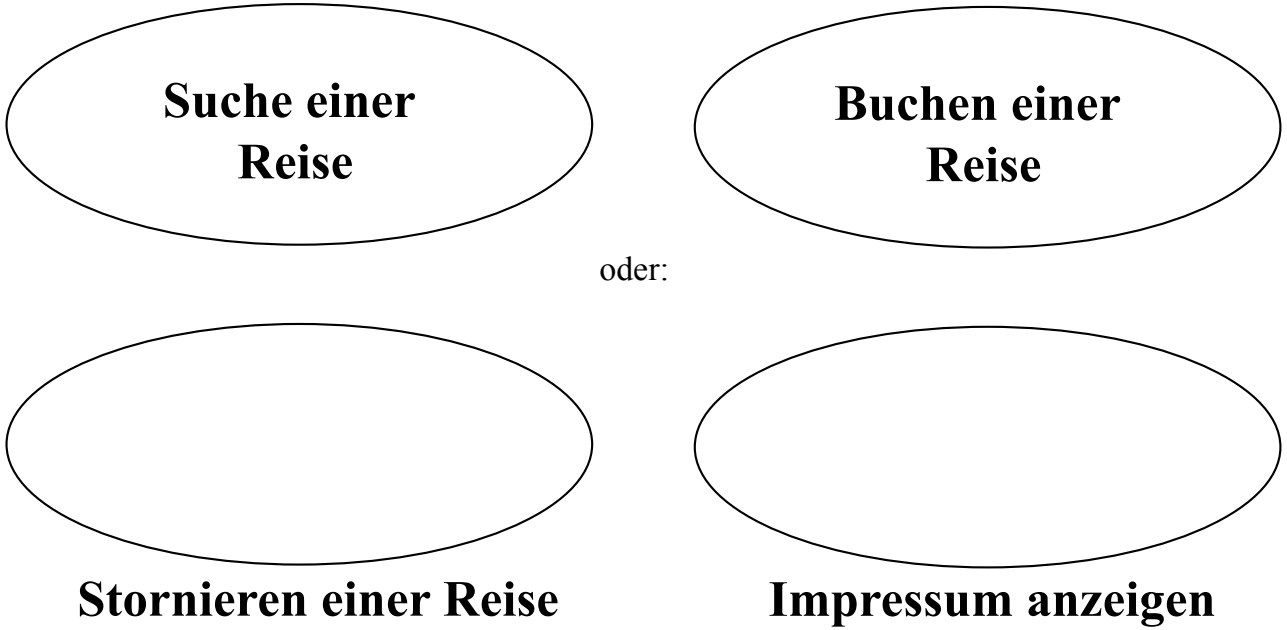
Use Case Diagramme – Akteur (Actor)

- Ein Akteur (engl.: *actor*) repräsentiert eine externe Entität, die mit dem System interagiert
- Ein Akteur kann folgende Formen annehmen:
  - ein realer Benutzer des Systems (vorzugsweise)
  - die Schnittstelle zu externen Systemen
- Ein Akteur hat eine konkrete, feste Bezeichnung, die eine Rolle darstellt.
- Modellierung in UML:



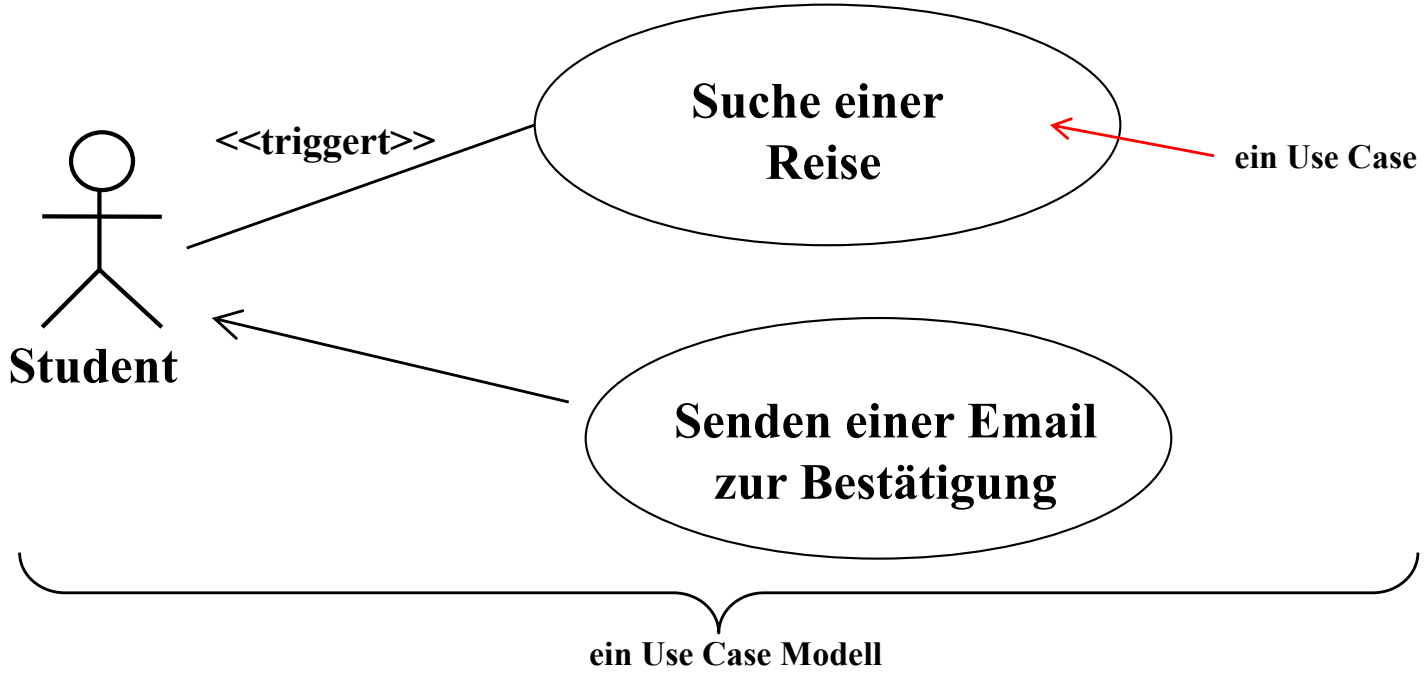
Use Case Diagramme – (einzelne) Use Case

- Ein Use Case (Anwendungsfall) beschreibt eine (Teil-)Funktionalität eines Systems, die aus Sicht eines Akteurs in Anspruch genommen werden kann.
- Menge von Ereignisse (Aktionen), die, schrittweise ausgeführt, ein spezielles Verhalten formen
- Heuristik: Verwendung eines aktiven Verbs zur Verdeutlichung der Aktivität
- In der UML durch ein Oval modelliert. Beispiel:



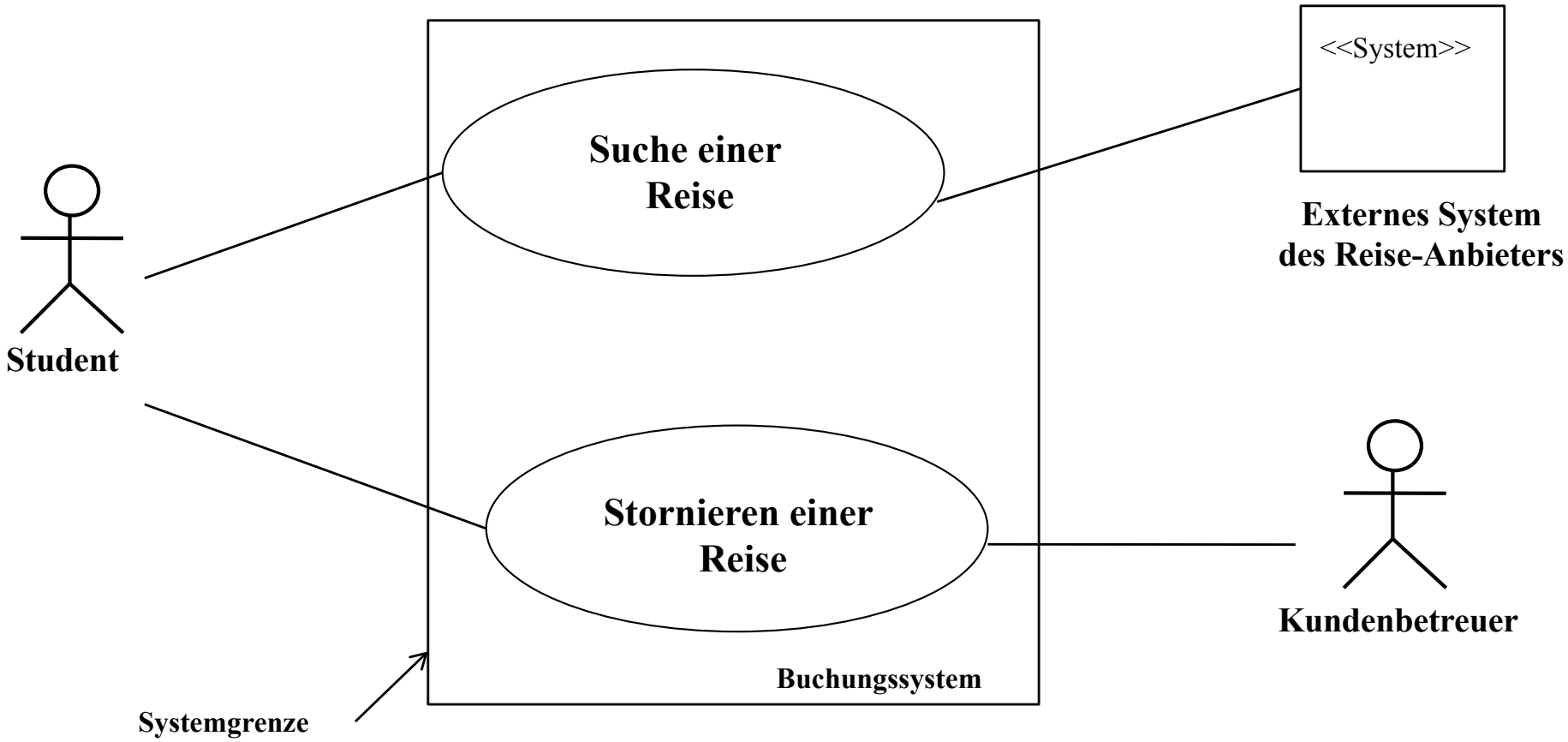
Use Case Diagramm – Interaktion zwischen Akteur und System

- Die Gesamtheit aller Use Cases und die involvierten Akteure bezeichnet man als Use Case Modell des Systems.
- Akteur kann aktiv oder passiv an der Ausführung beteiligt sein
- Die Interaktion zwischen Akteur und Use Case wird mit einer annotierten Linie (Assoziation) modelliert. Optional:
  - Beschreibung der Interaktion mit Stereotyp <<....>> (selten)
  - Gerichtete Assoziation um einen *einseitigen* Datenfluss zu modellieren



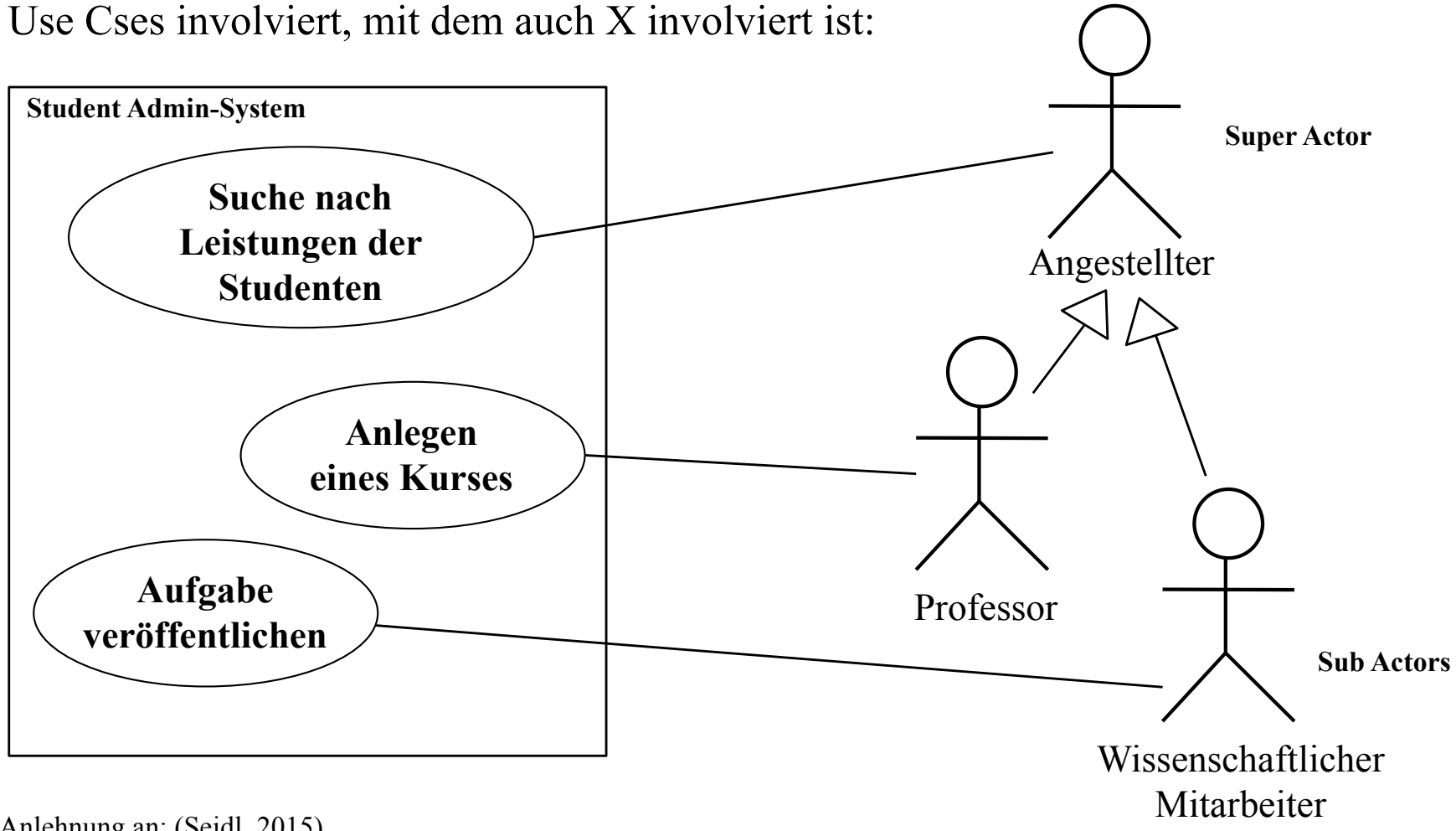
Use Case Diagramme – Interaktion zwischen Akteur und System

- Ein Use Case kann mit weiteren, externen Akteuren interagieren, um die Funktionalität auszuführen
- Mit einem Kasten werden die Grenzen des Systems modelliert (*interne* Teilsysteme können modelliert werden)



Beziehungen zwischen Akteuren

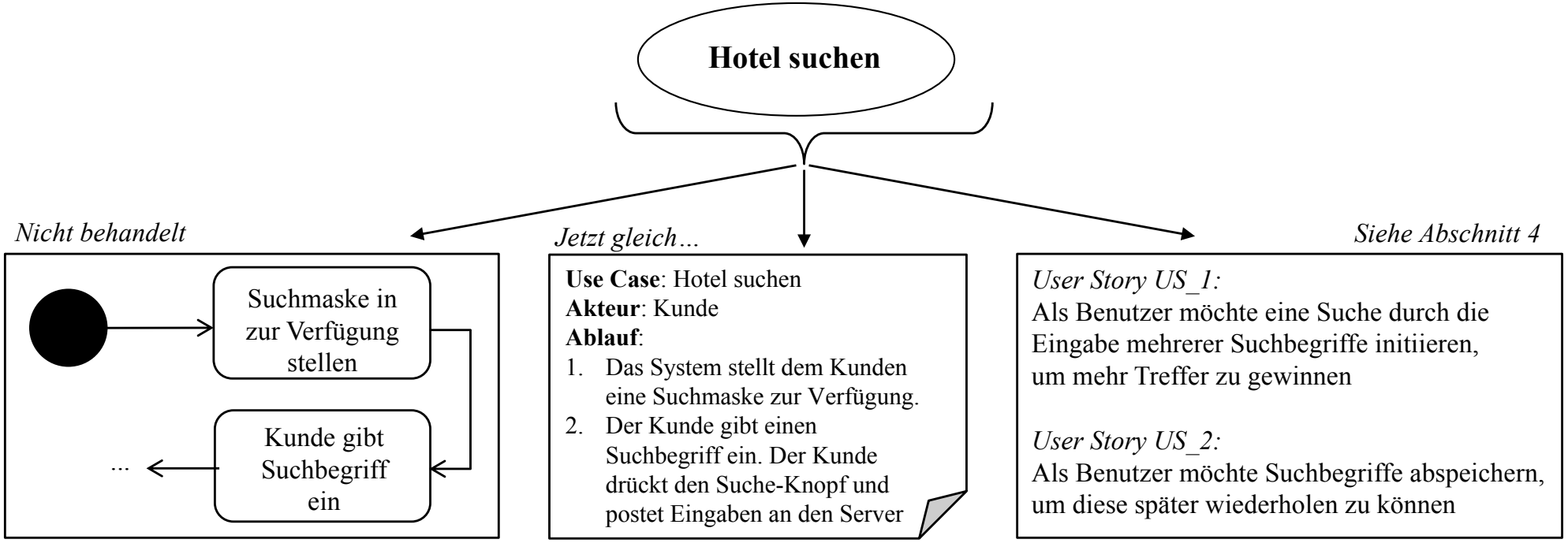
- Hierarchische Beziehungen zwischen Akteure können durch eine Vererbungs-beziehung (is-a Relationship) modelliert werden
- Wenn ein Actor Y (Sub Actor) von einem Actor X (Super Actor) erbt, dann ist Y mit allen Use Cses involviert, mit dem auch X involviert ist:



Beispiel: in Anlehnung an: (Seidl, 2015)

Verhaltensbeschreibung eines Use Cases

- Die graphische Modellierung von Use Cases reicht nicht aus, um die vollständigen Interaktionen zwischen dem System und Akteuren abzubilden.
- Abhilfe: Jeder einzelne Use Case muss durch eine zusätzliche Verhaltensbeschreibung ergänzt werden. Drei Varianten:



Modellierung des Ereignisflusses durch UML Aktivitätsdiagramm (Rupp, 2012)

Textuelle Beschreibung des Ereignisflusses (Bruegge und Dutoit, 2013)

Detaillierung der Funktionen durch User Stories (Rau, 2016)

Template für textuelle Use Cases

Name des Systems (optional)	
Name des Use Case	
Teilnehmende Akteure	
Ereignisfluss	1. Ereignis A 2. Ereignis B .. m. Ereignis Z
Vorbedingung	
Nachbedingung	
Qualitätsanforderungen	

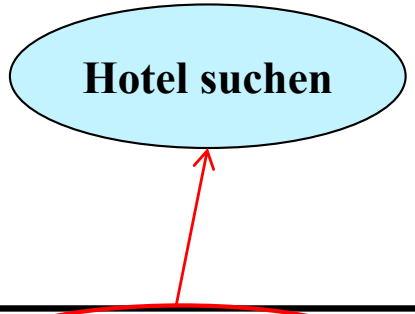
- Weitere Elemente (z.B. Alternativer Ereignisfluss, Risiken) möglich, siehe (Oestereich, 2012), (Rupp, 2009), (Bergmann, 2014)
- Vorteil:
  - Einfachere, intuitivere Verhaltensmodellierung auch für Fachseite möglich
  - Bessere Ableitung von Test Cases (siehe Abschnitt 3.3 und Übung)

Quelle: (Brügge und Dutoit, 2013)

Beispiel eines textuellen Use Case

Name des Use Cases	Hotel suchen
Teilnehmende Akteure	Anwender, Reise-Anbieter
Ereignisfluss **Einrückung verdeutlicht ein Ereignis, das vom System initiiert wird.** **Achtung: System ist kein Akteur!**	1. Anwender gibt Suchbegriff ins Eingabefeld der Dialogseite „Suchemaske“ ein und leitet ihn weiter. 2. SYSTEM empfängt den Suchauftrag und leitet diese an alle externen Reiseanbieter weiter. 3. SYSTEM empfängt die Ergebnisse, bereitet sie auf und schickt sie an den Anwender 4. Anwender navigiert durch das Suchergebnis
Vorbedingung	Anwender ist ein registrierter Kunde, ist eingeloggt Anwender hat die „Suchemaske“ geöffnet
Nachbedingung	Anwender hat die Ergebnisse auf seiner Seite
Qualitätsanforderungen	Die Ausgabe soll binnen 5 Sek. erfolgen

Abbildung eines Text Use Case auf UML Use Case



Name des Use Cases	**Hotel suchen**
Teilnehmende Akteure	Anwender, Reise-Anbieter
Ereignisfluss	1. Anwender gibt Suchbegriff ins Eingabefeld der Dialogseite „Suchemaske“ ein und leitet ihn weiter. 2. SYSTEM empfängt den Suchauftrag und leitet diese an alle externen Reiseanbieter weiter. 3. SYSTEM empfängt die Ergebnisse, bereitet sie auf und schickt sie an den Anwender 4. Anwender navigiert durch das Suchergebnis
Vorbedingung	Anwender ist ein registrierter Kunde, ist eingeloggt Anwender hat die „Suchemaske“ geöffnet
Nachbedingung	Anwender hat die Ergebnisse auf seiner Seite
Qualitätsanforderungen	Die Ausgabe soll binnen 5 Sek. erfolgen

Abbildung eines Text Use Case auf UML Use Case

Name des Use Cases	Hotel suchen
Teilnehmende Akteure	**Anwender, Reise-Anbieter**
Ereignisfluss	1. Anwender gibt Suchbegriff ins Eingabefeld der Dialogseite „Suchemaske“ ein und leitet ihn weiter. 2. SYSTEM empfängt den Suchauftrag und leitet diese an alle externen Reiseanbieter weiter. 3. SYSTEM empfängt die Ergebnisse, bereitet sie auf und schickt sie an den Anwender 4. Anwender navigiert durch das Suchergebnis
Vorbedingung	Anwender ist ein registrierter Kunde, ist eingeloggt Anwender hat die „Suchemaske“ geöffnet
Nachbedingung	Anwender hat die Ergebnisse auf seiner Seite
Qualitätsanforderungen	Die Ausgabe soll binnen 5 Sek. erfolgen

Textuelle Beschreibung von Use Cases mit Beziehungen

- Einfache Fallunterscheidungen im Ereignisfluss können pro Ereignis integriert werden (Verwendung von Wörtern wie „Falls“, „wenn“ oder „if“)

Ereignisfluss

(Basis Use Case)

...

x. Kunde bestätigt die Buchung seiner Reise und übermittelt damit seine eingegeben Kundendaten

y. Falls die Kundendaten korrekt sind, bereitet das SYSTEM die Buchung vor und legt einen neuen Datensatz „Buchung“ für den Kunden an.

z. Falls die Kundendaten *nicht* korrekt sind, fordert das SYSTEM den Kunden auf, die Kundendaten neu einzugeben

---

- Auf komplexe Kontrollstrukturen wie „while“ sollte in dieser Phase der Entwicklung verzichtet werden!
- Generell: nicht zu formal spezifizieren (es gibt keinen „Use Case Compiler“)

Textuelle Beschreibung von Use Cases mit Beziehungen

- Komplexe Fallunterscheidungen im Ereignisfluss können durch “Alternative Ereignisflüsse“ modelliert werden (Template nach (Bergsmann, 2014))

Normaler Ereignisfluss

(Basis Use Case)

…

x. Kunde bestätigt die Buchung seiner Reise und übermittelt damit seine eingegeben Kundendaten

   y. Das SYSTEM bereitet die Buchung vor und legt einen neuen Datensatz „Buchung“ für den Kunden an.

Alternativer Ereignisfluss

(Basis Use Case)

Variante “Falsche Eingabe“ falls in Ereignis y. die Daten nicht korrekt sind:

1. Das SYSTEM stellt die Kundendaten in einem neuen Dialog dar und markiert dabei die falsche Eingaben
2. Das SYSTEM fordert den Kunden auf, die Kundendaten neu einzugeben.
3. Kunde gibt die Daten neu ein und bestätigt diese erneut
4. Gehe zu Ereignis y.

Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	Agile Erhebung von Anforderungen mit User Stories	✓
3	**Objektorientierte Erhebung von Anforderungen mit Use Cases**	
	3.1 – UML Use Cases	✓
	3.2 – Beziehungen zwischen UML Use Cases	
	3.3 – Ableitung von Test Cases	
4	Gegenüberstellung der Ansätze	
5	Zusammenfassung und Ausblick	

Beziehungen zwischen UML Use Cases

- Die UML sieht drei verschiedene Beziehungstypen zwischen Use Cases vor:
  - Einbeziehung (<<include>>)
  - Erweiterung (<<extend>>)
  - Generalisierung (Vererbung wie bei Klassendiagrammen)

- Sind teilweise umstritten, insbesondere Erweiterung (<<extend>>):
  - Häufig missverstanden oder in Quellen unterschiedlich erklärt und verwendet
  - „Sollte nicht verwendet werden!“ (Oestereich, 2012)

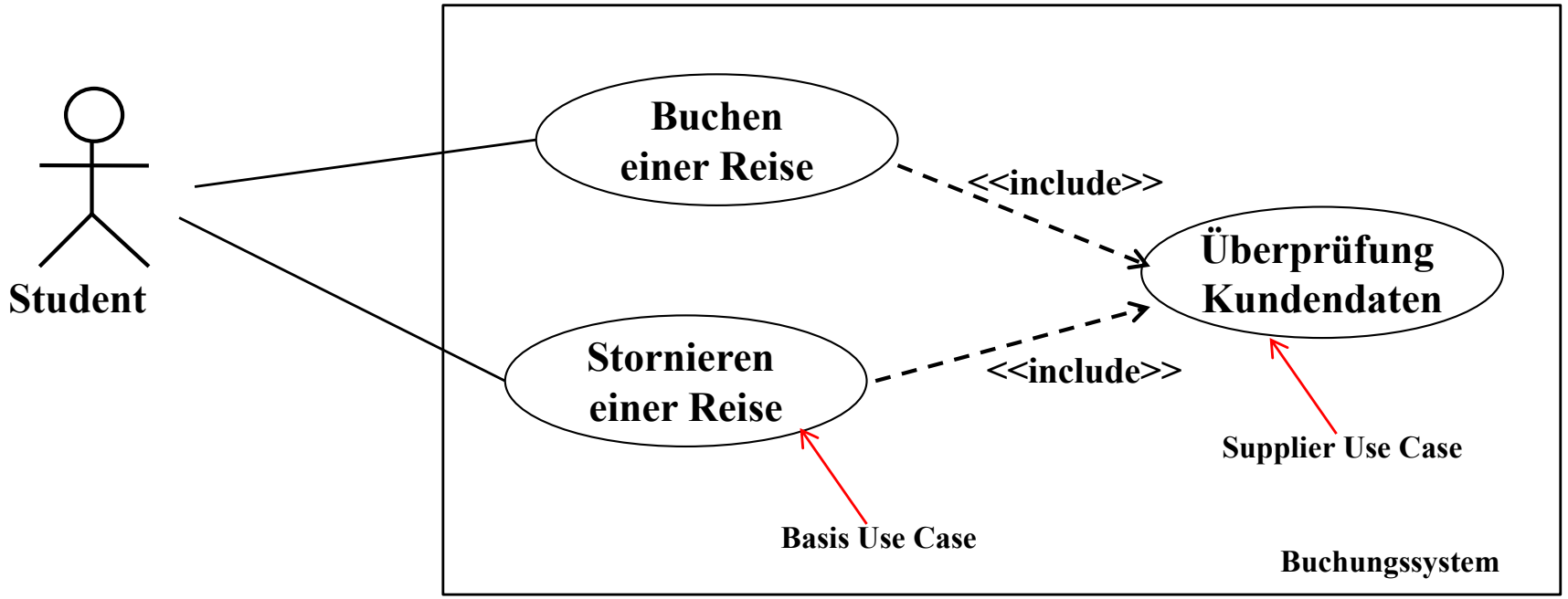
- Generell: Spärliche, bedarfsgerechte Verwendung von Beziehungstypen ratsam!

- (Persönliche) Feststellung und Empfehlung:
  - Verwendung Beziehung <<include>> ist in Praxis oft sinnvoll
  - In Vorlesung SE-1: theoretische Einführung aller Beziehungen (klausurrelevant!)
  - In Vorlesung SE-2: pragmatische Verwendung der Beziehungen (Verwendung insbesondere von <<extend>> *nicht mehr empfohlen*)

Use Case Diagramme – Beziehungen zwischen Use Cases

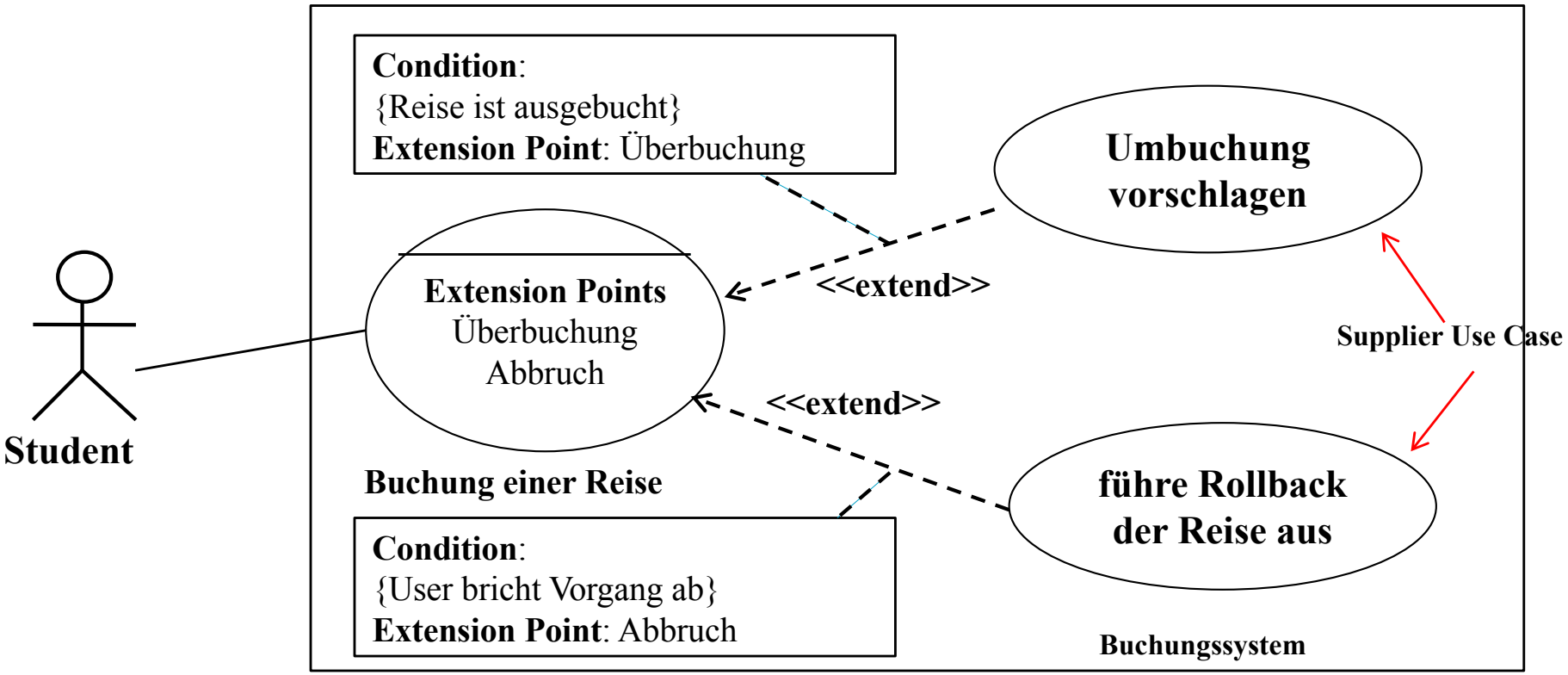
- Ein Use Case kann die Funktionen eines anderen Use Case einbeziehen (<<include>> Beziehung) und das Ergebnis weiternutzen

- Aufgabe: Ausgliederung von Funktionen, die von vielen Use Cases benutzt (wiederverwendet) werden können

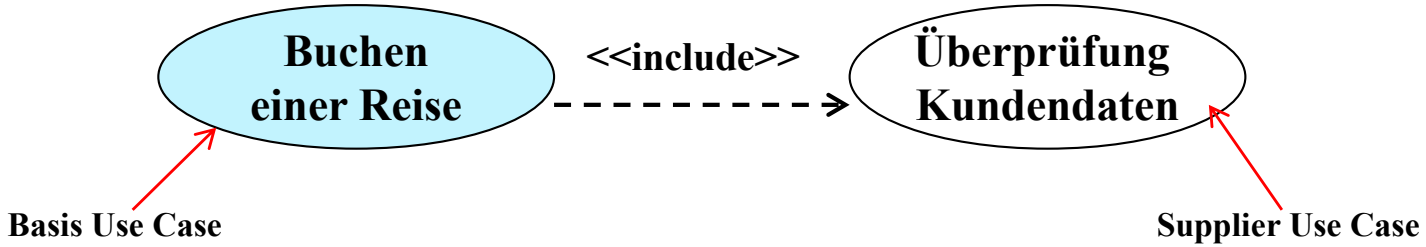


Use Case Diagramme – Beziehungen zwischen Use Cases

- Die <<extend>> Beziehung wird verwendet, um einen Use Case mit Ausnahmefällen oder optionalen Funktionen zu erweitern.
- Die Erweiterung erfolgt an Extension Points (EP) im Basis Use Case
- Condition wird spezifiziert, wann der Basis Use Case erweitert wird



Textuelle Beschreibung von Use Cases mit Beziehungen



Ereignisfluss

(Basis Use Case)

...

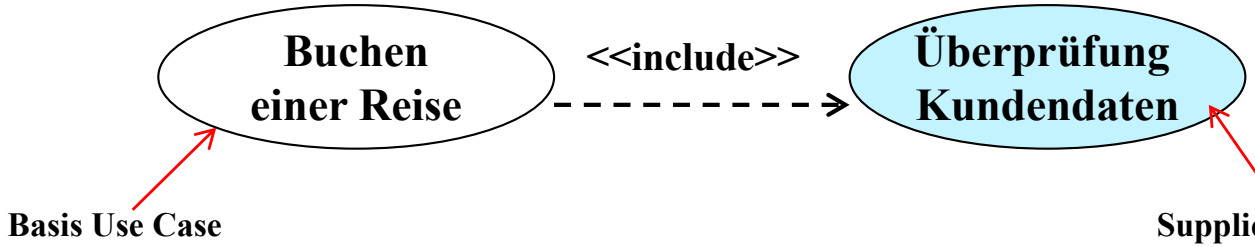
x. Kunde bestätigt die Buchung seiner Reise und übermittelt damit seine eingegeben Kundendaten

y. SYSTEM empfängt die Kundendaten und überprüft diese (durch die Einbindung des Use Case „Überprüfung Kundendaten“)

z. Falls die Kundendaten korrekt sind, bereitet das SYSTEM die Buchung vor und legt einen neuen Datensatz „Buchung“ für den Kunden an.

- Andeutung der Einbindung durch Phrase wie „durch die Einbindung“ oder „durch Verwendung.

Textuelle Beschreibung von Use Cases mit Beziehungen

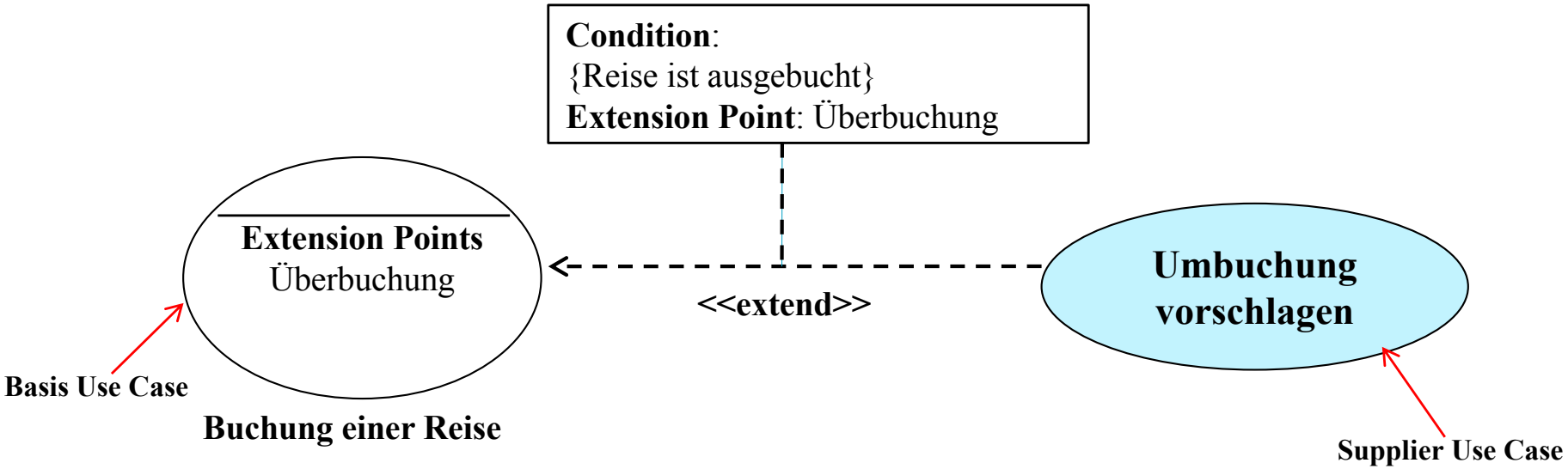


Ereignisfluss	x.	SYSTEM kontrolliert anhand der internen Kunden-Datenbank, ob die Kundendaten gültig sind
(Supplier Use Case)	y.	SYSTEM ermittelt das Ergebnis und sendet diese dem aufrufendem Teil-System zurück.

Vorbedingung und andere Elemente ausgelassen!

- Rückgabewert an den Basis Use Case kann angedeutet werden (z.B. „liefert Bestätigung zurück“, „sendet Ergebnis zurück“)

Textuelle Beschreibung von Use Cases mit Beziehungen



Ereignisfluss

(Supplier Use Case)

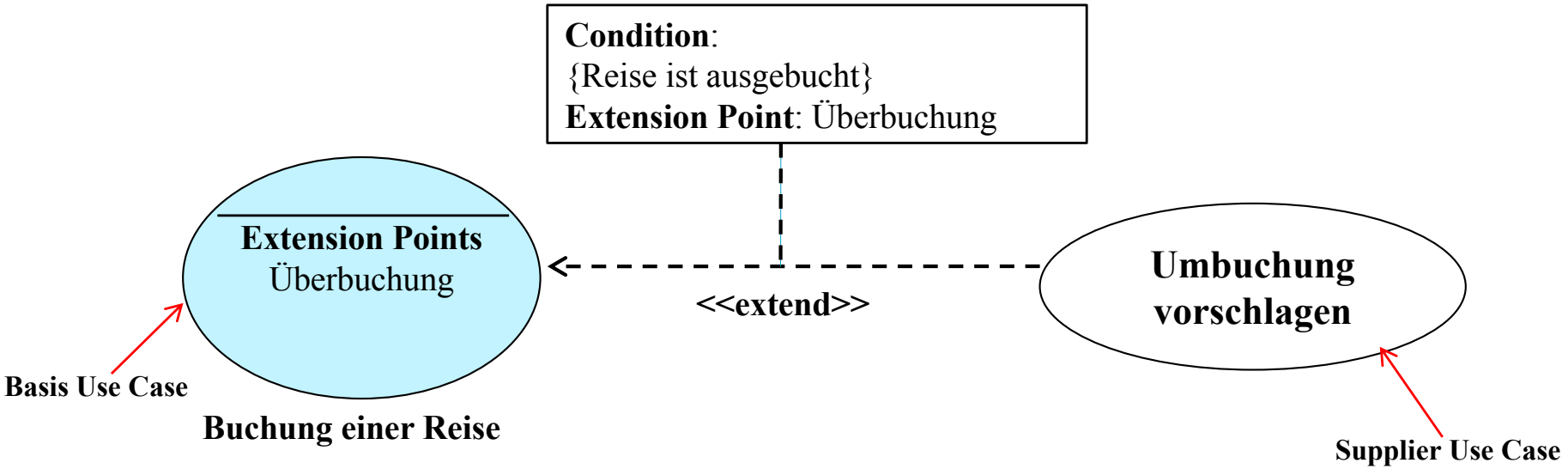
Der Use erweitert alle Use Cases mit dem Extension Point „Überbuchung“

Falls Condition (Reise ist ausgebucht) = wahr:

1. Das SYSTEM ermittelt ein alternatives Datum
2. Das SYSTEM übersendet dem Benutzer den Vorschlag.

Vorbedingung und andere Elemente ausgelassen!

Textuelle Beschreibung von Use Cases mit Beziehungen



Ereignisfluss

(Basis Use Case)

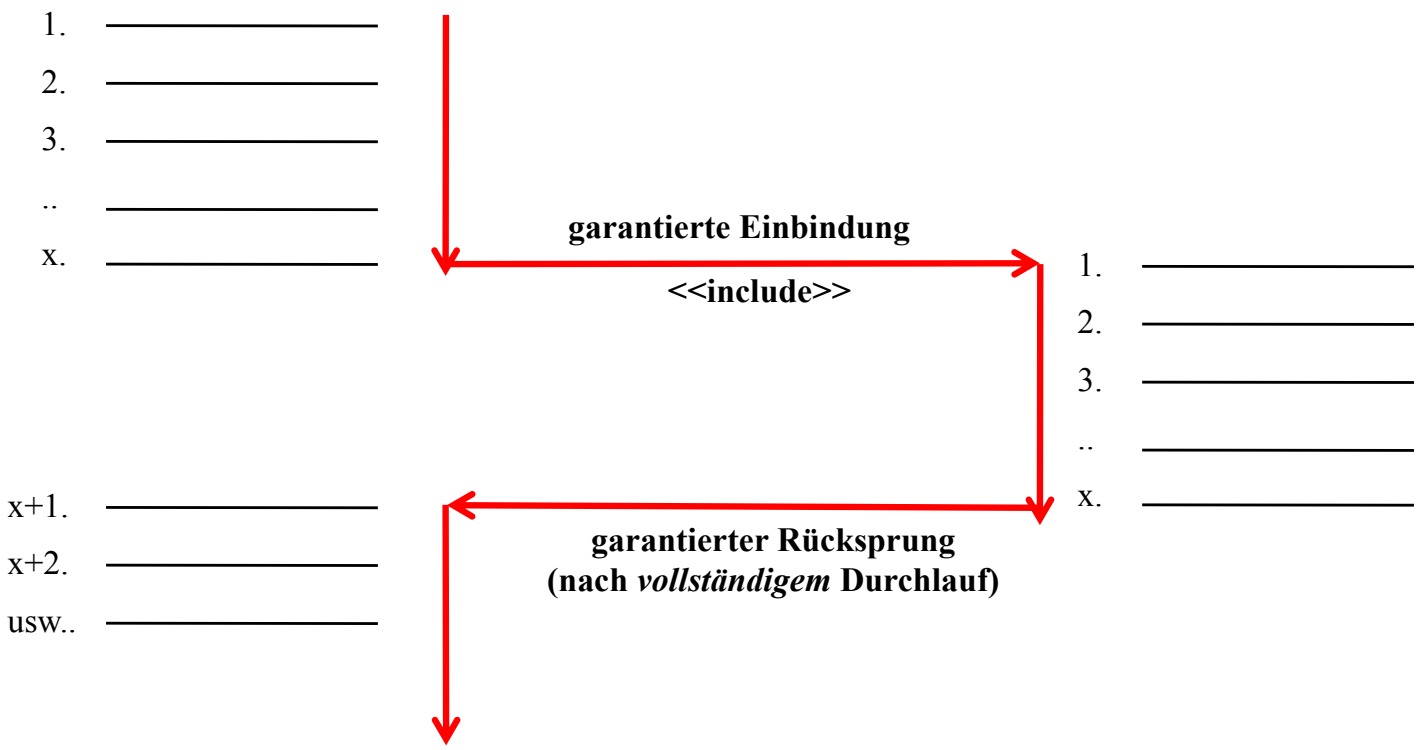
1. Das SYSTEM führt die Buchung durch und ermittelt einen Buchungsstatus

[Überbuchung]

2. Das SYSTEM sendet dem Anwender eine Bestätigung inklusive Buchungsstatus

Unterschied bei den Beziehungen: <<include>>

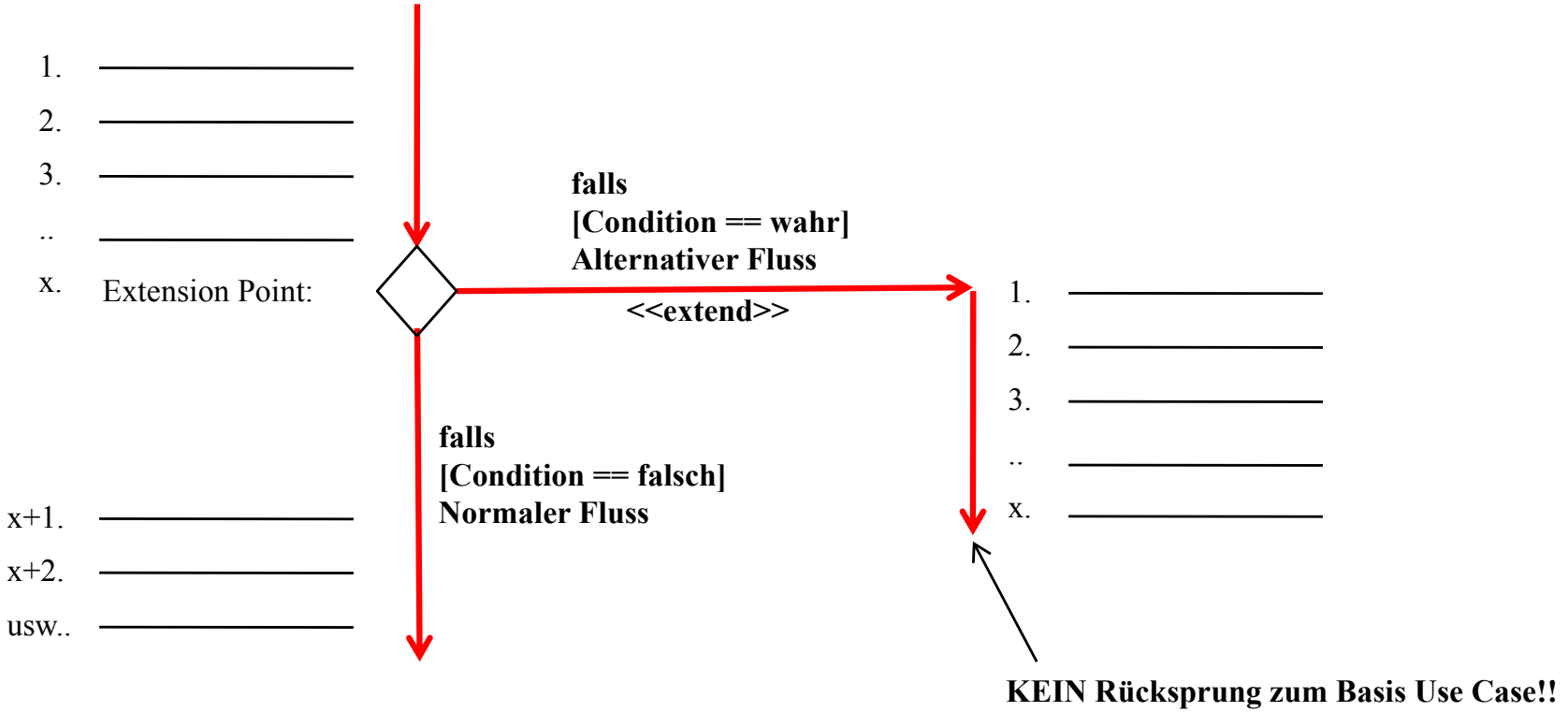
- Bei der Einbindung eines Use Case mit <<include>> wird der Ereignisfluss des Basis Use Case nicht unterbrochen:



Quelle: (Rupp, 2009)

Unterschied bei den Beziehungen: <<extend>>

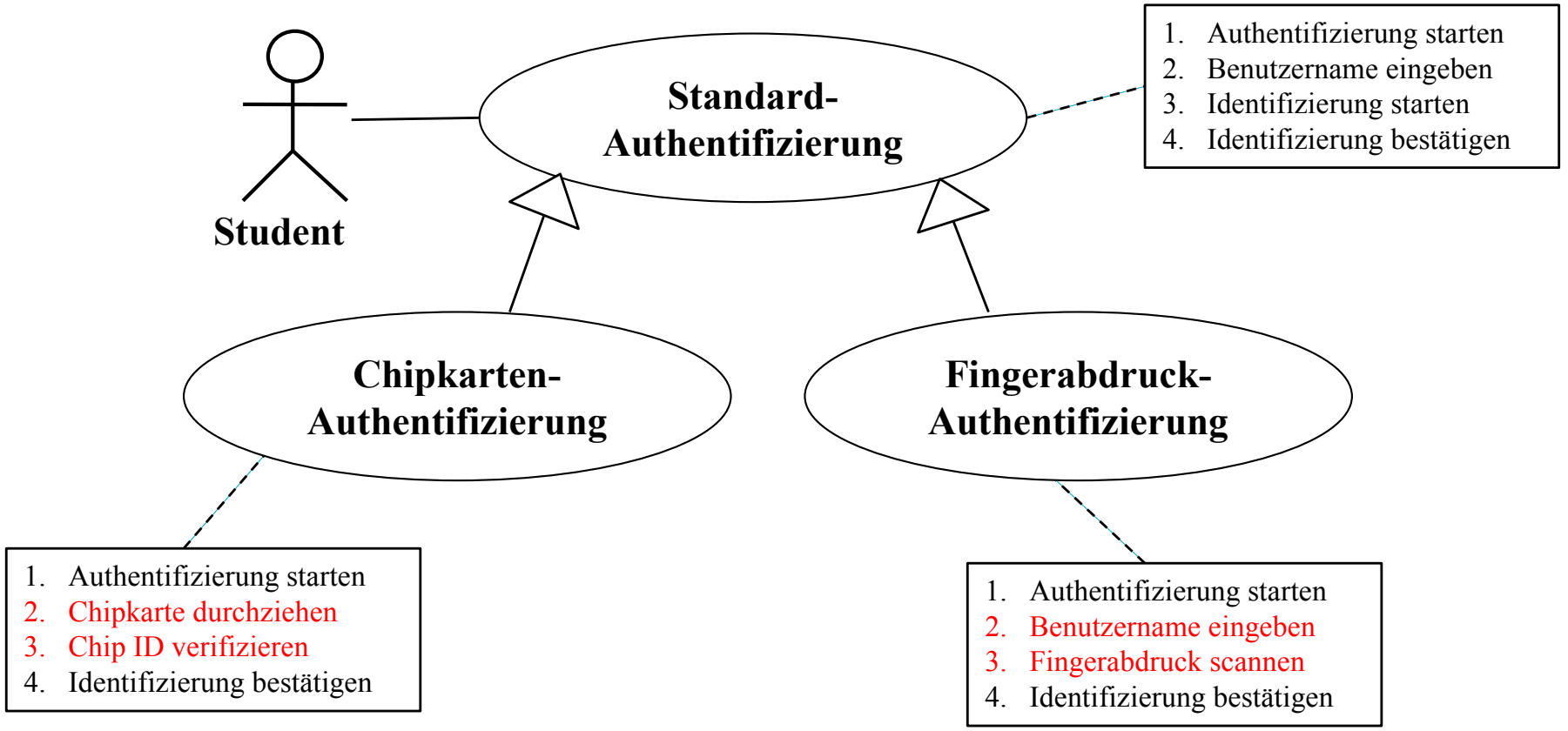
- Bei der Einbindung eines Use Case mit <<extend>> wird der Ereignisfluss des Basis Use Case unterbrochen, falls die Condition erfüllt ist:



Quelle: (Rupp, 2009)

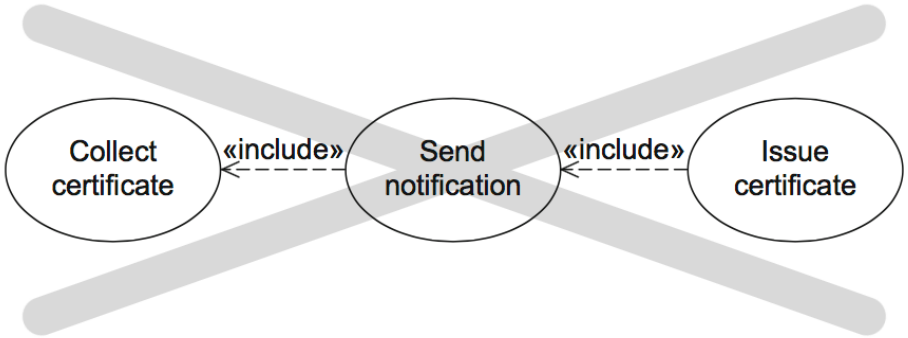
Use Cases – Generalisierung und Spezialisierung

- Ein genereller Basis Use Case kann durch Use Cases spezialisiert werden.
- Vererbung der Ereignisse durch die spezialisierten Use Cases
- Einzelne Ereignisse können von einen spezialisierten Use Case überschrieben werden

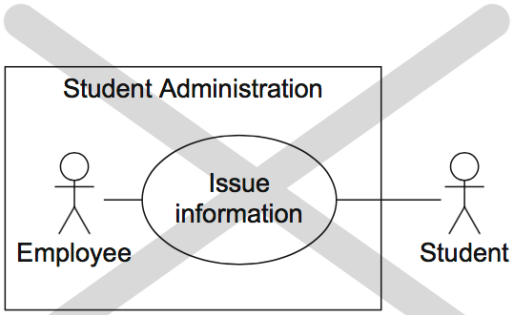


Quelle: (Rupp, 2009)

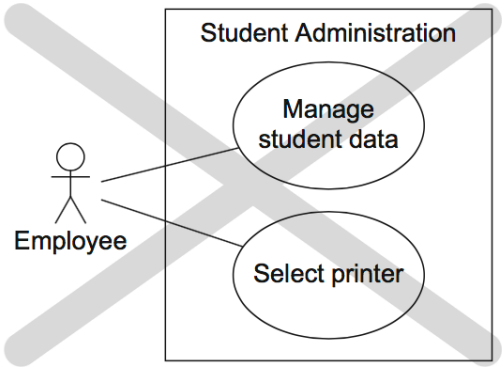
Typische Fehler bei der Verwendung von Use Cases
(Quelle: Seidl et al., 2015)



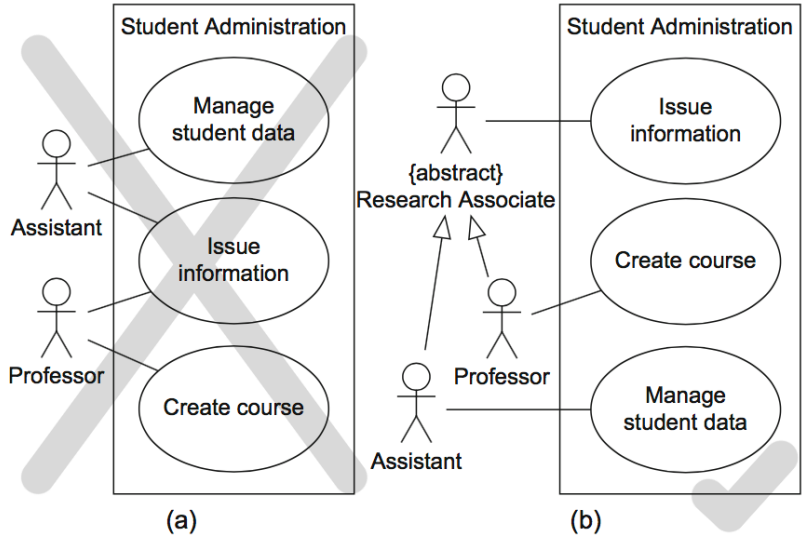
1. Keine Prozessmodellierung mit Use Cases!!



2. Akteure immer außerhalb der Systemgrenze!



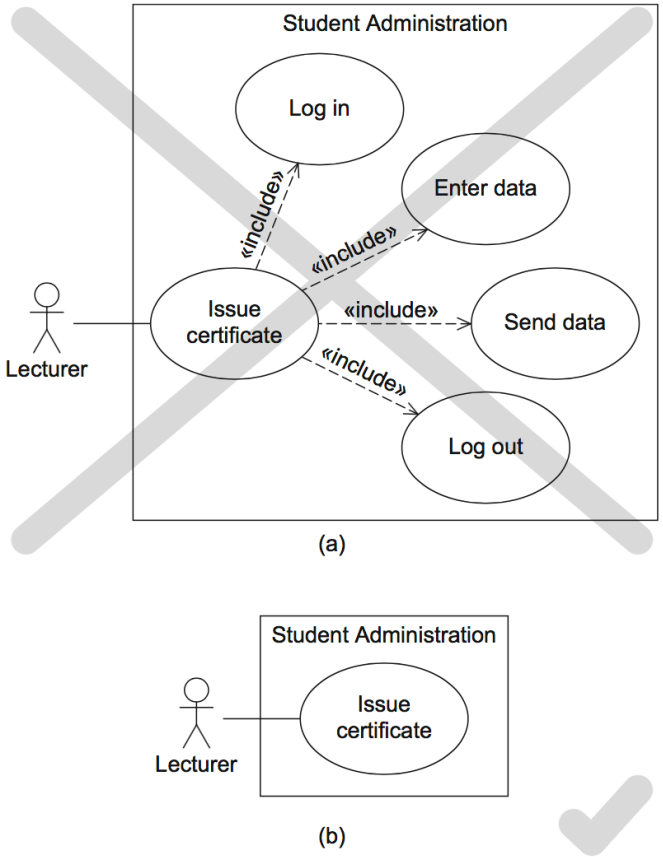
3. Einheitlicher Abstraktionslevel (Fokus auf fachliche Modellierung, *keine* zu technische Perspektive)



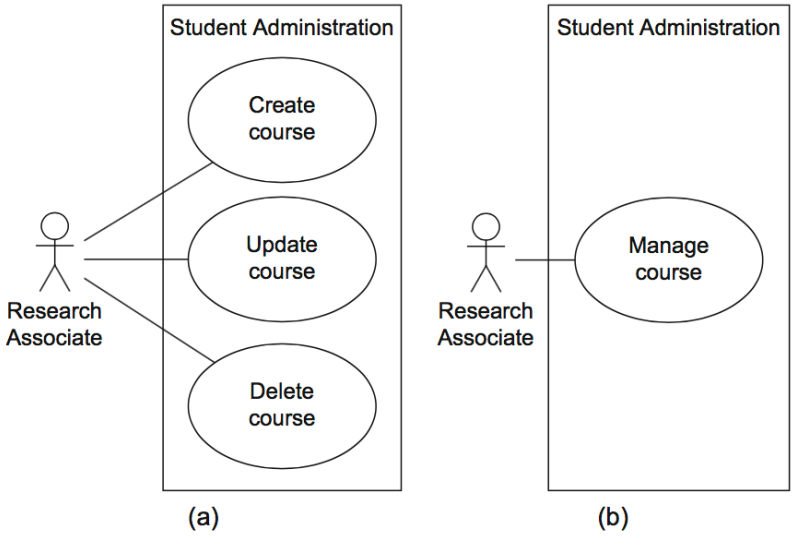
4. Korrekte Assoziationsbildung zwischen Akteuren und Use Cases (Use Case assoziiert mit zwei Akteuren → beide sind im Ablauf involviert!)

Typische Fehler bei der Verwendung von Use Cases
(Quelle: Seidl et al., 2015)

Generelle Regel bei der Modellierung eines Use Case Modells: „Weniger ist manchmal mehr!!“



5. Keine funktionale Decomposition mit einem Front-End Use Case (auch mit <<extend>> häufig gesehen…)



6. Zusammenlegung von zu detaillierten Use Cases in einen übergreifenden Use Case (Abstraktion!)

Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	Agile Erhebung von Anforderungen mit User Stories	✓
3	**Objektorientierte Erhebung von Anforderungen mit Use Cases**	
	3.1 – UML Use Cases	✓
	3.2 – Beziehungen zwischen UML Use Cases	✓
	3.3 – Ableitung von Test Cases	
4	Gegenüberstellung der Ansätze	
5	Zusammenfassung und Ausblick	

Test Cases aus Use Cases ableiten (Brügge und Dutoit, 2013)

- Für einen Testfall (engl.: Test Case) kann aus dem Use Case eine Testprozedur abgeleitet werden
  - Ziel: Funktionsweise für einen Use Case muss getestet werden (Akzeptanztest)
  - Test Prozedur muss für einen externen Tester verständlich sein
- Vorgehen bei der Erstellung einer Testprozedur:
  - definiere *pro* Ereignis konkrete Testdaten (Input)
  - definiere *pro* Ereignis die erwartenden Ausgabedaten (Output, Zwischenergebnis)
  - füge konkrete Zusatzinfos hinzu (z.B. Login, Password, usw.)
- Die Vorbedienung und (vor allem!) die Nachbedingung sollten ebenfalls im in einem Test Case überprüft werden
  - Vorbedingung kann ausgelagert werden, falls hier ein Use Case verwendet wird
- Vorteil: Test Case kann aus textueller Beschreibung einfacher abgeleitet werden

Template für eine Test Case Beschreibung

Test Case für den Akzeptanztest des Use Case „Hotel Suchen“		
Teilschritt der Testprozedur (inkl. Input)	**Das zu erwartende Ergebnis (der Output)**	**Status** (vom Tester auszufüllen)
1. Aufruf der Sucheseite ↖ **Vorbedingung testen!**	Sucheseite der Anwendung erscheint	**OK**
2. Der Kunde (normale Person, nicht eingeloggt) gibt einen Suchbegriff („München“) ein und drückt den Suche-Knopf	Auf der Seite erscheint eine Ausgabe, dass der Suchvorgang im Gange ist und wenige Sekunden nun dauern kann	**OK**
3. Das System schickt eine Liste der Hotels gemäß Suchbegriffs	Eine gültige Liste von Hotels in München wird angezeigt (50 Treffer maximal, Preis absteigend sortiert)	**OK**
4. ---	Der Kunde kann die Ergebnisse auf seinem Screen ablesen ↖ **Nachbedingung testen!**	**OK**
	Gesamtergebnis:	**bestanden**

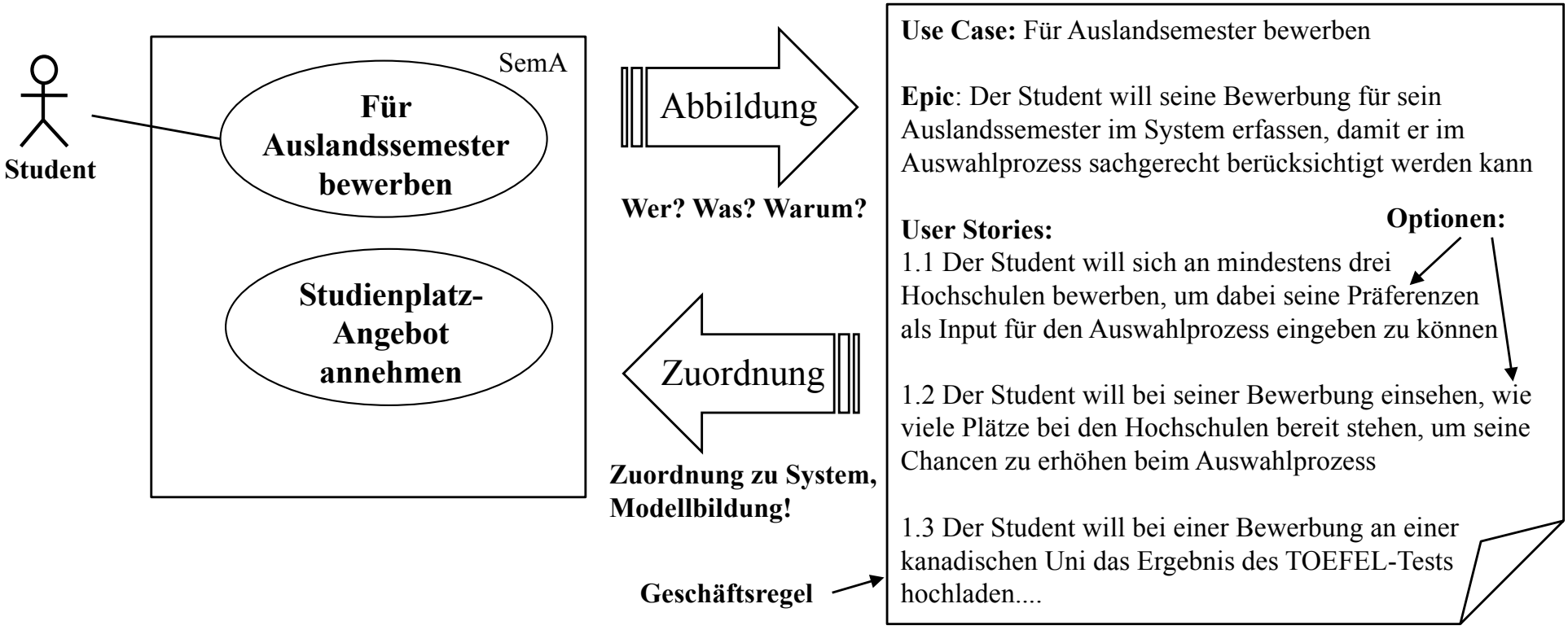
Quelle: eigene Darstellung

Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	Agile Erhebung von Anforderungen mit User Stories	✓
3	Objektorientierte Erhebung von Anforderungen mit Use Cases	✓
4	**Gegenüberstellung der Ansätze Use Case vs. User Story**	
5	Zusammenfassung und Ausblick	

Zusammenhang Use Case vs. User Story

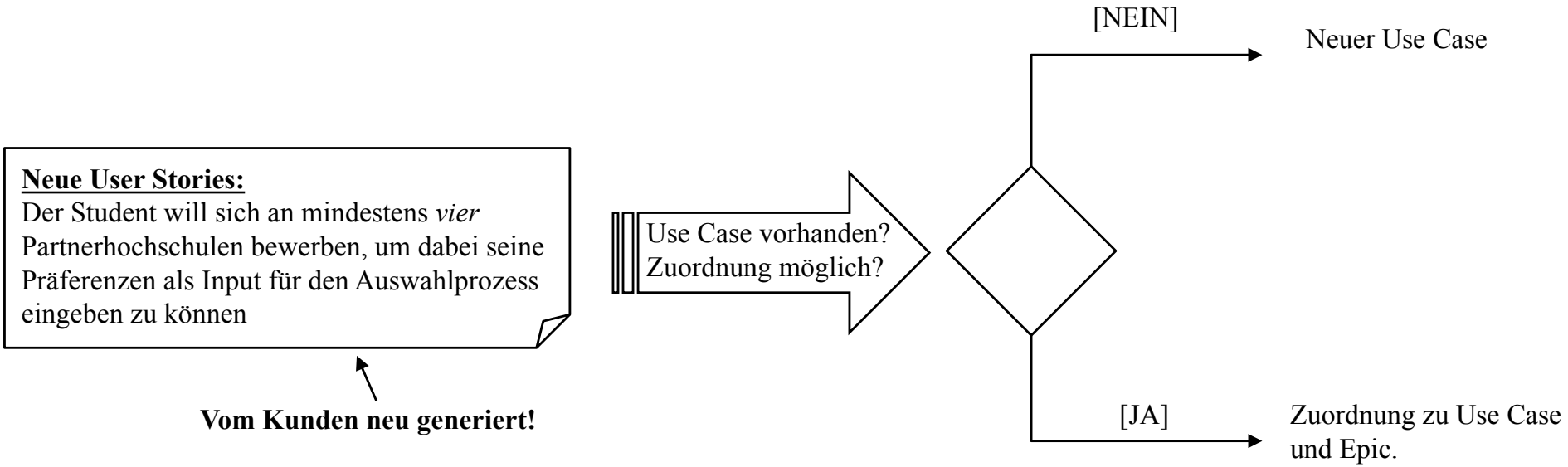
- Die Anforderungsartefakte Use Cases und User Story lassen sich im Lebenszyklus aufeinander abbilden und ergänzen voneinander (Rau, 2016)
- Abbildung von Use Cases auf Epics (1:1) sowie Zerlegung der Use Cases (der Epics) in kleinere User Stories nebst Mehrwert (1:n):
  - Methode zur Zerlegung: Interner Ablauf des Use Case, Optionen, Geschäftsregeln



Beispiel gekürzt übernommen aus (Rau, 2016), Kapitel 2.3

Zusammenhang Geschäftsprozess vs. Use Case vs. User Story

- Die Anforderungsartefakte Use Cases und User Story lassen sich im Lebenszyklus aufeinander abbilden und ergänzen voneinander (Rau, 2016)
- Kommen im Verlauf des Projekt neue User Stories in das Backlog, so muss geprüft werden, ob dazu ein passender Use Case und ggf. eine Control-Klasse vorhanden ist
- Schematische Darstellung (in Anlehnung an UML):



Darstellung Prozess: eigene Quelle, Interpretation nach (Rau, 2016), Kapiteln 2.3 und 3.3

Beziehung User Story und Use Case

- Kritik an User Stories: zu knappe Darstellung der funktionalen Anforderungen (Coplien, 2010)

- Feststellung bzw. Empfehlung: wenn User Stories *komplexe* funktionale Anforderungen repräsentieren, sollten diese durch ergänzende Infos ergänzt werden:
  - Konkretisierung einer User Story durch eine textuelle Ablaufbeschreibung (z.B. in Details)
  - Ablaufbeschreibung *kann* ähnlicher Aufbau wie ein textueller Use Case besitzen
  - Vorteil: bessere Testbarkeit der User Story (Ableitung Test Case aus Ablaufbeschreibung)

Use Case: Für Auslandsemester bewerben

Epic: Der Student will seine Bewerbung für sein Auslandssemester im System erfassen, damit er im Auswahlprozess sachgerecht berücksichtigt werden kann

User Stories:
1.1 Der Student will sich an mindestens drei Hochschulen bewerben, um dabei seine Präferenzen
als Input für den Auswahlprozess eingeben zu können
...

Detail: Ablaufbeschreibung für User Story US 1.1

1. Student öffnet die Unterseite zur Eingabe seiner persönlichen Angaben
2. Student gibt seine persönlichen Angaben ein
3. Das SYSTEM gibt ihm eine Übersicht der eingegebenen Daten, die der bestätigen kann, falls diese korrekt sind
4. Der Student wechselt zur Unterseite zur Auswahl der präferierten Hochschule

Abbildungen von Beziehungen in User Stories

- Beziehungen zu anderen Use Case sollten aus den User Stories *sehr pragmatisch* angewandt werden!

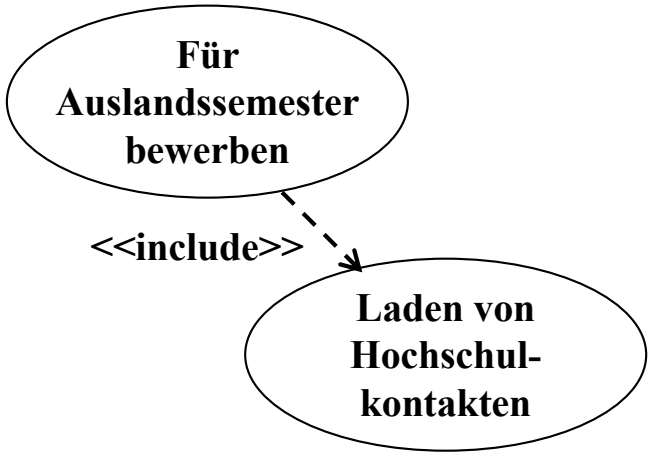
- Keine soliden Methoden aus Literatur vorhanden. Auch nicht im Sinne der agilen Softwareentwicklung (wenig Dokumentation, Modellierung nur bedarfsgerecht...)

- Allenfalls Andeutung in Details:

Detail: Ablaufbeschreibung für User Story US 1.1

1. Student öffnet die Unterseite zur Eingabe seiner persönlichen Angaben
2. Student gibt seine persönlichen Angaben ein
3. Das SYSTEM gibt ihm eine Übersicht der eingegebenen Daten, die der bestätigen kann, falls diese korrekt sind
4. Der Student wechselt zur Unterseite zur Auswahl der präferierten Hochschule
5. SYSTEM bezieht die aktuelle Liste der Hochschulen (durch Verwendung einer externen Funktionalität)

Teil des Use Case „Für Auslandssemester bewerben"



Gliederung dieser Veranstaltung

Kapitel 3: Modellierung und Erhebung von Softwareanforderungen		
1	Allgemeines über Anforderungen	✓
2	Agile Erhebung von Anforderungen mit User Stories	✓
3	Objektorientierte Erhebung von Anforderungen mit Use Cases	✓
4	Gegenüberstellung der Ansätze Use Case vs. User Story	✓
5	**Zusammenfassung und Ausblick**	

Zusammenfassung und Ausblick

Zusammenfassung

- Sie haben zwei wichtige Ansätze kennengelernt, mit denen man Anforderungen modellieren kann: Use Cases und User Stories
- Use Cases modellieren funktionale Anforderungen und stellen dabei die Interaktionen zwischen dem Software-System und den Akteuren dar
- User Stories repräsentieren knappe in sich abgeschlossene Anforderungen, die mit Hilfe von agilen Entwicklungsmethoden umgesetzt werden kann
- User Stories und Use Cases ergänzen sich: Use Cases, die funktionale Anforderungen beschreiben, können durch User Stories weiter beschrieben werden

Ausblick

- Use Cases und User Stories müssen auf eine objektorientierte Struktur abgebildet werden, damit sie im weiteren Verlauf von einer objektorientierten Programmiersprache umgesetzt werden können.

Quellen

- Oestereich, Bernd: Analyse und Design mit der UML 2.5, Oldenbourg Verlag, 2012
- Jacobson, I., M. Christerson, P. Jonsson: *Object-Oriented Software Engineering - A Use Case Driven Approach*, Addison-Wesley, 1992.
- Gloger, Boris: Scrum, 2. Aufl.. Hanser Verlag. 2009
- Cohn, Mike: User Stories Applied. Addison-Wesley, 2004.
- Rupp, C.: Requirements Engineering und Management. Hanser, 2009
- Wiegers, K.: „First Things First: Prioritizing Requirements“, in: *Software Development*, September 1999.
  Reprint: http://www.processimpact.com/articles/prioritizing.html.
- Coplien, J.: Lean Architecture: for Agile Software Development. Wiley, 2010
- Grots, A. und Pratschke, M.: Design Thinking – Kreativität als Methode. Marketing Review St. Gallen. 2-2009.

Quellen

- Rupp, Chris et. al.: Requirements-Engineering und –Management. Hanser, 5. Auflage. 2009. (oder 6. Auflage 2014)
- Bergsmann, J.: Requirements Engineering für die agile Softwareentwicklung. Dpunkt.verlag, 2014.
- Rau, Karl-Heinz: Agile objekt-orientierte Software-Entwicklung. Springer Vieweg. 2016.
- Seidl, Martina et al.: UML @ Classroom – An Introduction to Object-Oriented Modelling. Springer, 2015

Anhang