BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Agiles Requirements Engineering

Szenarien, Use Cases, User Stories

Master Technische Informatik –Embedded Systems–

Prof. Dr.-Ing. Hartmut Schirmacher

Inhalt

- Szenarien (Wdh.)
- Use Cases
- User Stories, Epics, Story Maps
- Use Case 2.0
- Use Cases vs. User Stories – was tun?

Szenarien

Szenario

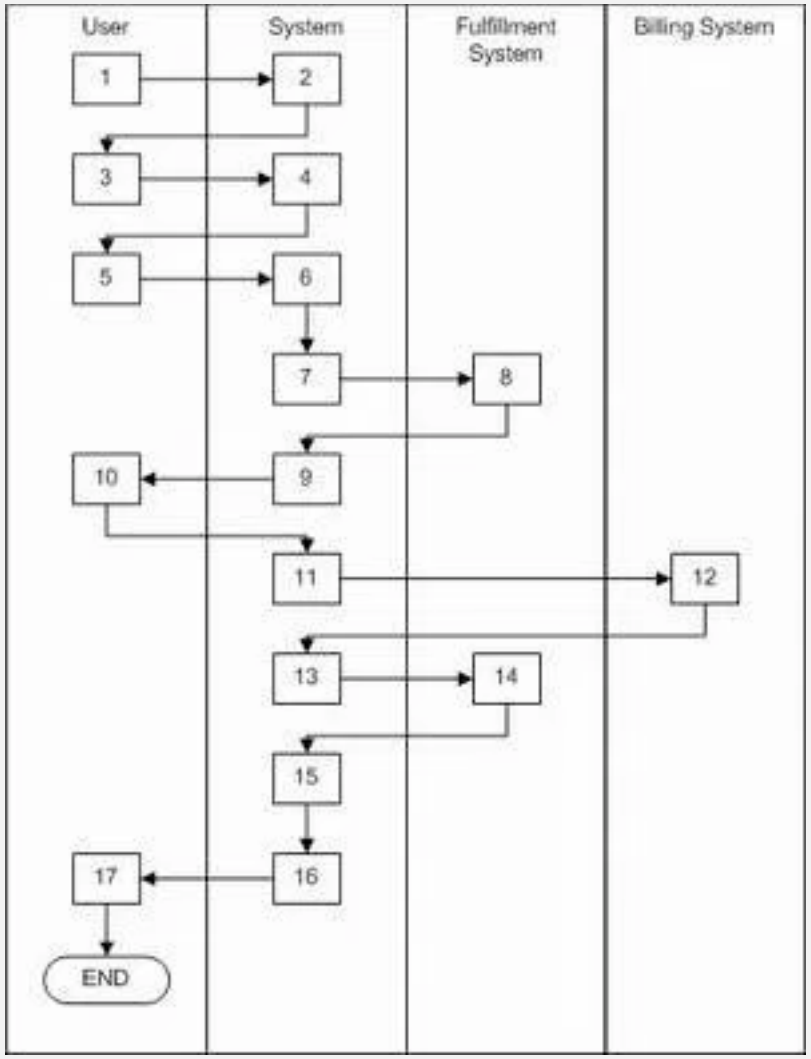
Szenario:

- Eine geordnete Menge von Interaktionen zwischen Partnern, in der Regel zwischen einem System und einer Menge systemexterner Akteure.
- Kann eine konkrete Interaktionsfolge (Beispielszenario) oder eine Menge möglicher Interaktionen (abstraktes Szenario oder Typszenario) sein.
- Szenarien variieren u.a. nach *:
  - Abstraktionsgrad vs. Realitätsnähe
  - Rolle/Zweck im Entwurfsprozess

- Bekannteste Form: Jacobson/UML:
  Use Case Scenario = Teil einer Use Case Beschreibung

*) A. Sutcliffe, Scenario-based Requirements Engineering, RE 2003 Mini-Tutorial

Scenario / Flow



- Ein Szenario oder Flow ist <u>eine</u> Möglichkeit, wie die Interaktion zwischen User und System(en) in Bezug auf ein bestimmtes Ziel ablaufen kann
- Man kann ein Szenario als Text beschreiben, oder z.B. durch Sequenzdiagramme
- Links: Beispiel mit vier „Swim Lanes“, welche vier Akteuren entsprechen

http://tynerblain.com/blog/2007/04/10/what-are-use-case-scenarios/

Use Cases

Use Case / Anwendungsfall

- Ivar Jacobson
  - Ursprünglich bei Ericsson (Telekom-Industrie)
  - 1987 *Object Oriented Development in an Industrial Environment*. OOPSLA '87 Proceedings. ACM
  - 1995 Firma von *Rational Software* übernommen, Mitarbeit an UML und Rational Unified Process

Ivar Jacobson *

- Alistair Cockburn [sprich: Coh-Börn]
  - *Einer der Mitbegründer des agilen Manifests* ***
  - *Structuring Use Cases with Goals*, J. OO Programming 1997
  - *Writing Effective Use Cases*, Addison-Wesley 2001

A. Cockburn**

- Jede Menge andere Literatur, jede Menge Variationen

*) Wikipedia / Mikemacd, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6377157

) http://alistair.cockburn.us/ *) http://agilemanifesto.org/

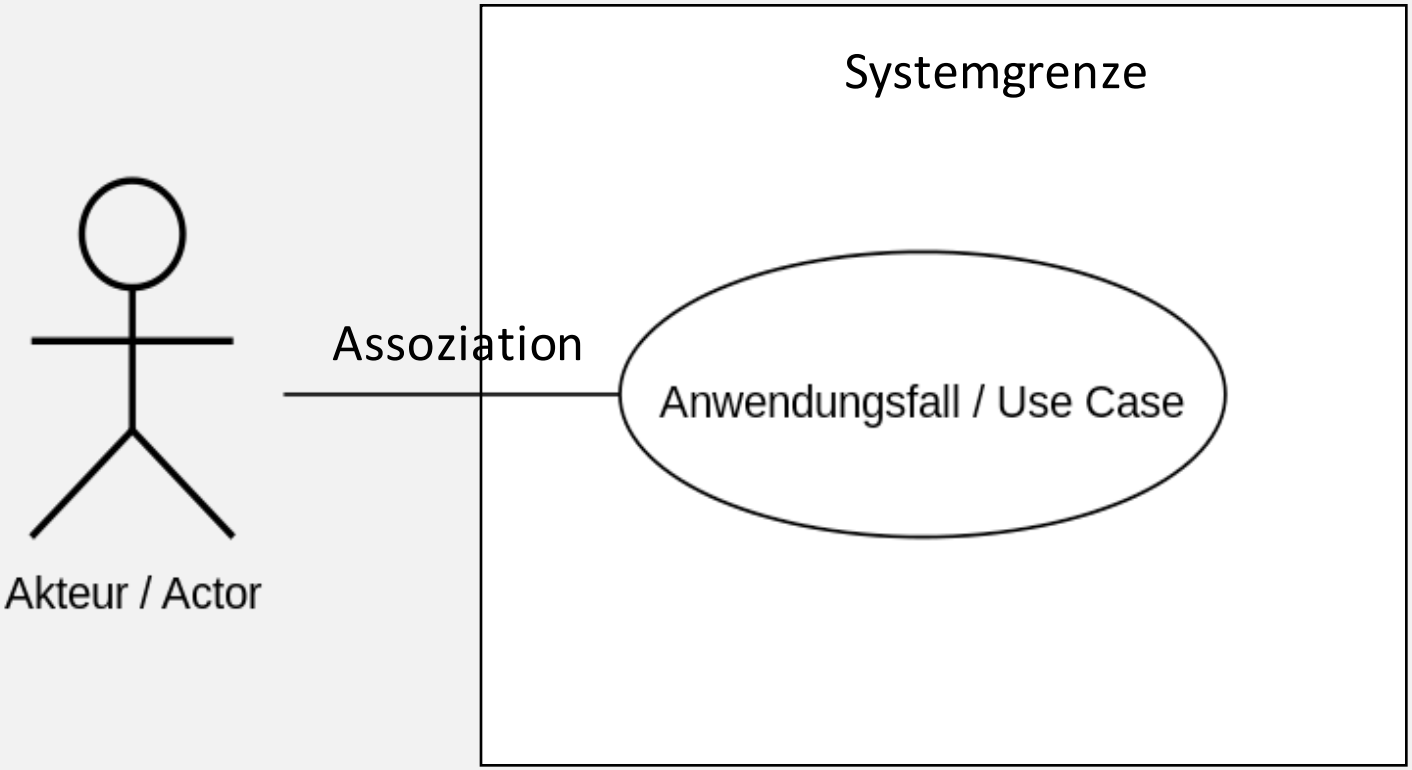
Use Case Diagramm in UML2

- Verhaltensdiagramm (früher: Strukturdiagramm)
- Für den Überblick über grundlegende Funktionalität des Systems

- Spezifikation der Abläufe/Szenarien nur durch ergänzende Dokumente oder Modelle
  - Freier Text
  - Aktivitäts-, Sequenz-, Kollaborations-, Kommunikationsdiagramm

https://de.wikipedia.org/wiki/Anwendungsfalldiagramm

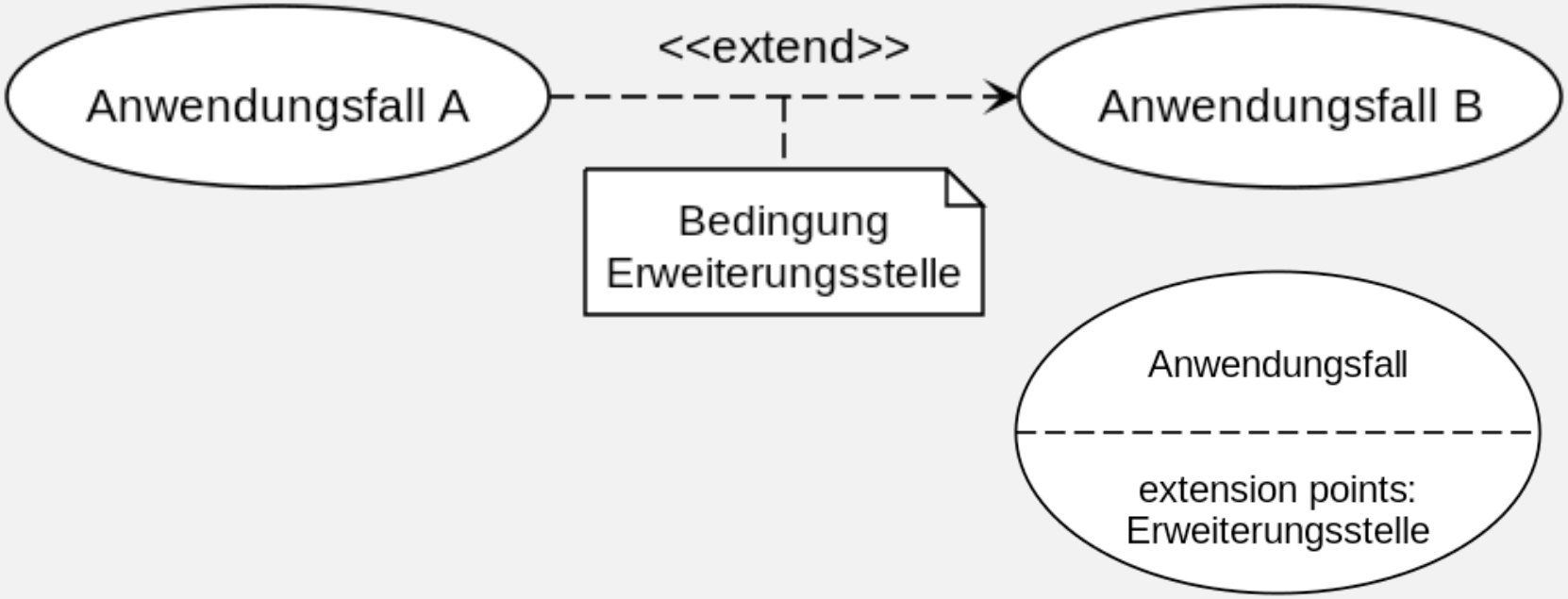
Kontext, Akteur, Anwendungsfall (UML 2)



- Von außen beobachtbares Verhalten an einer Schnittstelle des Systems
- Akteur = Benutzer oder externe Komponente

https://de.wikipedia.org/wiki/Anwendungsfalldiagramm

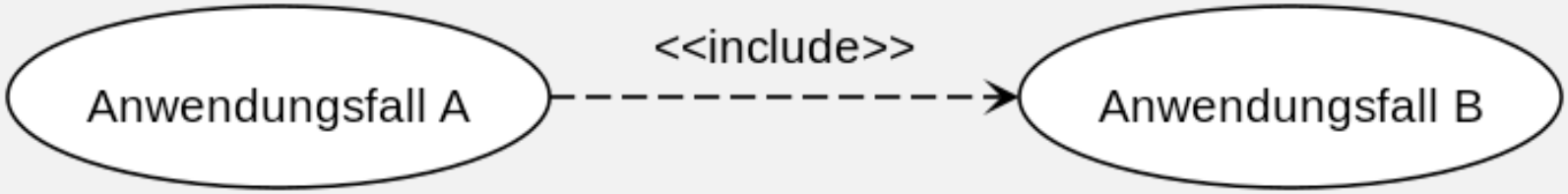
Extend, ggf. mit Bedingung (UML 2)



- Eine Erweiterung ist ein zusätzlicher Ablauf, der unter bestimmten Bedingungen im Rahmen eines Use Cases auftritt
  (davor, danach, mitten drin, parallel dazu ?)

https://de.wikipedia.org/wiki/Anwendungsfalldiagramm

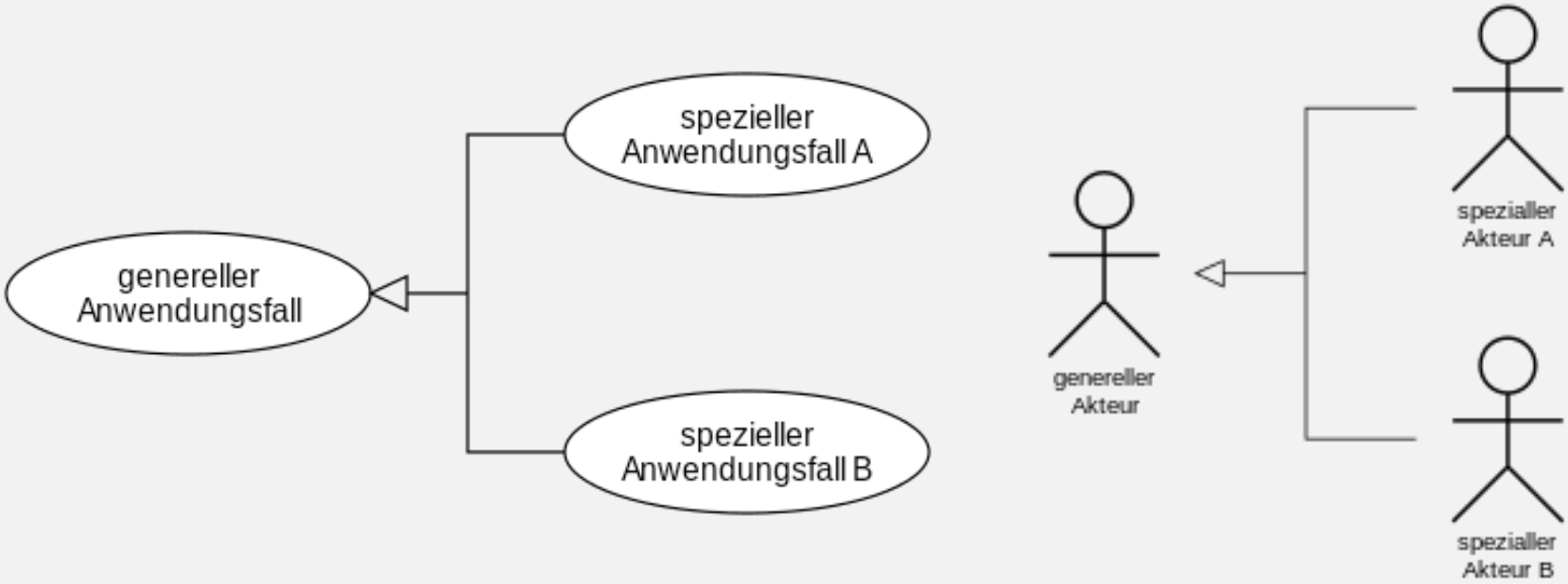
Include (UML 2)



- Use Cases können in verschiedenen Detail-Stufen ausgeführt werden
- Ein Use Case kann ein „Unterpunkt“ vieler anderer Use Cases sein

https://de.wikipedia.org/wiki/Anwendungsfalldiagramm

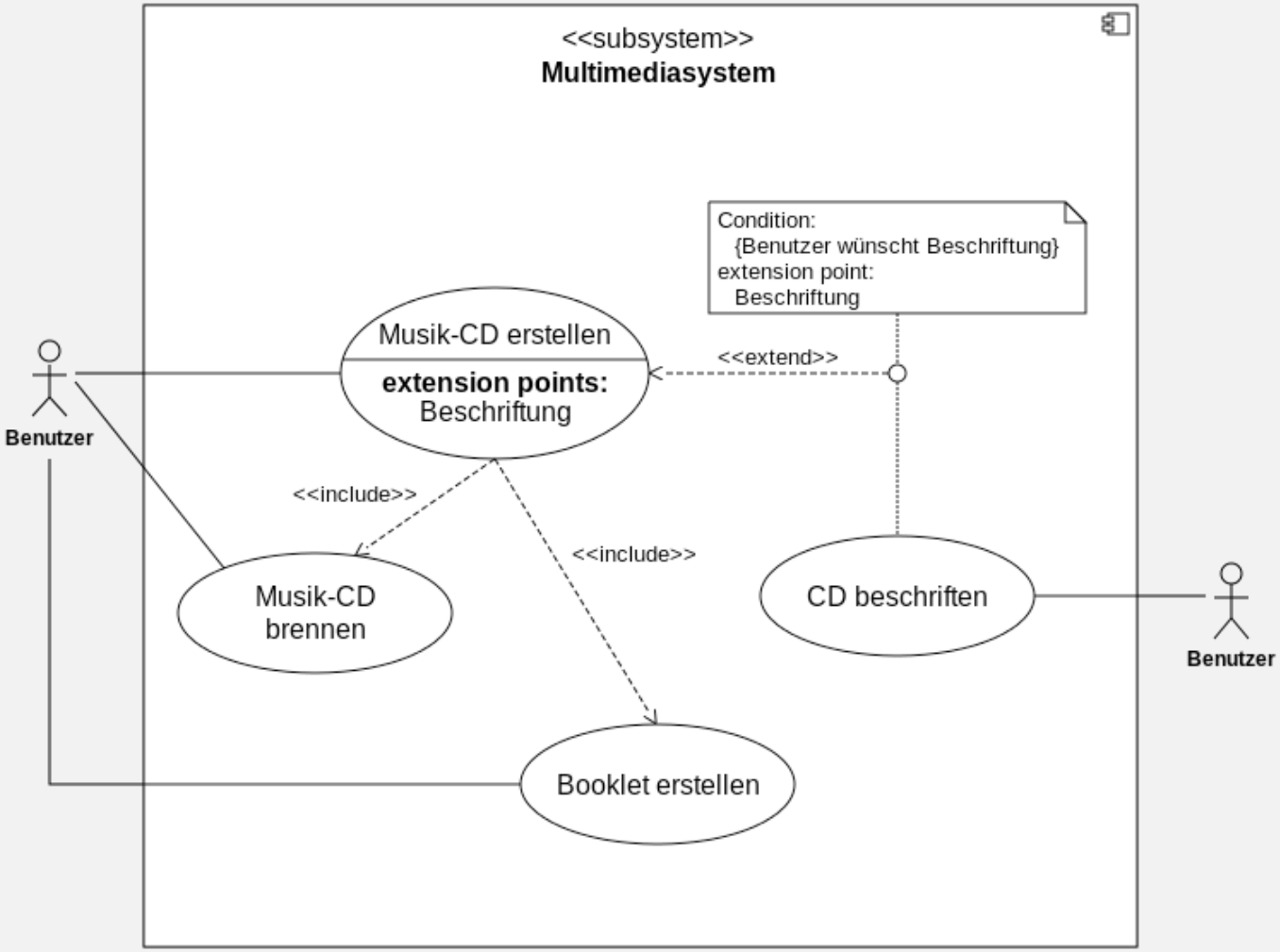
Generalisierung (UML 2)



- Spezialisierung / Generalisierung von Akteuren und Use Cases
- Nicht für Detail-Stufen, sondern für das Zusammenfassen mehrerer Akteure und Abläufe zu einem abstrakteren

https://de.wikipedia.org/wiki/Anwendungsfalldiagramm

Einfaches Beispiel



https://de.wikipedia.org/wiki/Anwendungsfalldiagramm

Use Cases nach Cockburn

- UML fokussiert darauf, welche Ellipsen es gibt
  - Übersicht – wer interagiert wofür mit dem System

- Cockburn kümmert sich darum,
  - was in den Ellipsen passiert
  - wie man Use Cases gut aufschreibt
  - wie man Use Cases ordnet, entwickelt und mit Detailgraden umgeht

Use Cases nach Cockburn

A use case captures a contract between the stakeholders of a system about its behaviour.

[...]

The primary actor initiates an interaction with the system to accomplish some goal.

[...]

Different sequences of behaviour, or scenarios, can unfold [...].

The use case gathers those different scenarios [for the same goal, Anm. Schirmacher] together.

A. Cockburn, Writing Effective Use Cases. Addison-Wesley 2001.

Wozu können Use Cases eingesetzt werden?

- Geschäftsprozesse beschreiben
  - Interaktionen zwischen Stakeholdern zur Beschreibung von Abläufen zur und Analyse, wo/wie ein System Abläufe verbessern könnte
- Interaktion mit einem zu planenden System beschreiben
  - Anforderungen an das Verhalten eines Systems mit noch nicht genau spezifizierter Schnittstelle
- Interaktion mit einem geplanten System beschreiben
  - Spezifikation eines Systems mit gegebener Schnittstelle
  - Z.B. Mockup/Wireframe ist bereits entworfen, Use Cases beschreiben, wie das UI nun für das jeweilige Nutzerziel genau verwendet werden kann

Minimale Use Case Struktur

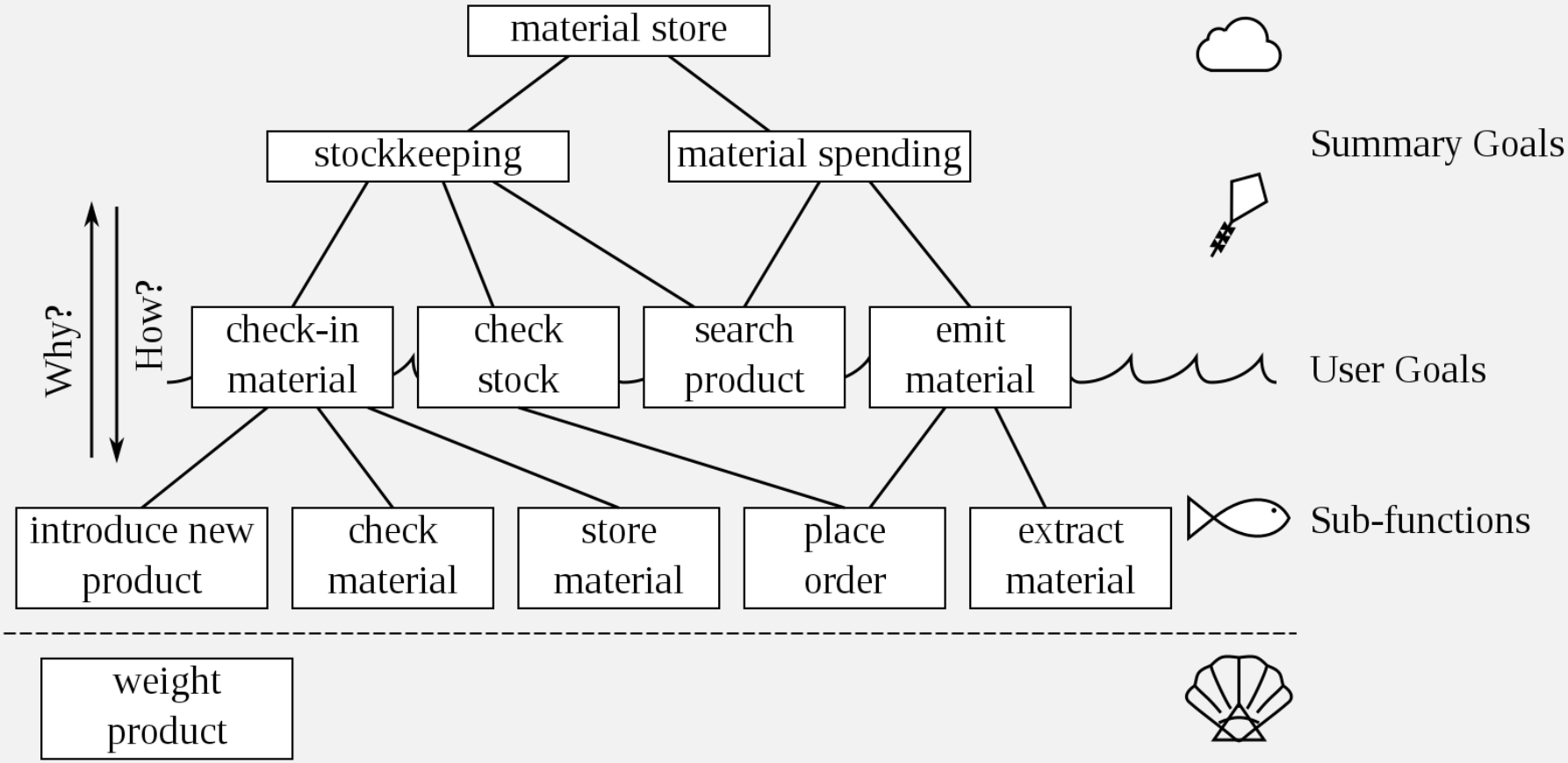
- Titel: Primäres Ziel des Akteurs
- System: Betroffenes System oder Komponente
- Level: Zusammenfassung | User-Ziel | Subfunktion

- Haupterfolgsszenario (Main Success Scenario)
  - Nummerierte Abfolge von 3-9 Interaktionsschritten
  - Aktiv formuliert, Akteure/System werden klar benannt

- Liste der Erweiterungen
  - Bedingung: in welchem Schritt muss welche Bedingung eintreten
  - Ein-Satz-Beschreibung des gewünschten Verhaltens, oder Verweis auf ein separat beschriebenes alternatives Szenario

http://alistair.cockburn.us/Sampler+of+good+and+bad+use+cases

Detail-Stufen (Levels) nach Cockburn



Bildquelle: Wikipedia (rechtefrei)

Effektive Use Cases nach Cockburn

Write something readable (!!!)

- Casual, readable use cases are still useful, whereas unreadable use cases won't get read

Work breadth-first, from lower precision to higher precision

- Precision Level 1: Primary actor’s name and goal
- Precision Level 2: The use case brief, or the main success scenario
- Precision Level 3: The extension conditions
- Precision Level 4: The extension handling steps

A. Cockburn, Writing Effective Use Cases. Addison-Wesley 2001. Auszüge: http://alistair.cockburn.us/get/2465

Good example (1/2)

Title: Register for Courses

Scope/System: Course Enrollment System

Level: User Goal

1. Student requests to construct a schedule.
2. The system prepares a blank schedule form.
3. The system gets available courses from the Course Catalog System.
4. Student selects up to 4 primary course offerings and 2 alternate course offerings from the available offerings.
5. For each course, the system verifies that the Student has the necessary prerequisites, adds the Student to the course, marking Student as “enrolled” for that course in the schedule.
6. When the Student indicates the schedule is complete, the system saves the schedule.

http://alistair.cockburn.us/Sampler+of+good+and+bad+use+cases

Good example (2/2)

1. Student requests to construct a schedule.
2. The system prepares a blank schedule form.
3. The system gets available courses from the Course Catalog System.
4. Student selects up to 4 primary course offerings and 2 alternate course offerings from the available offerings.
5. For each course, the system verifies that the Student has the necessary prerequisites, adds the Student to the course, marking Student as "enrolled" for that course in the schedule.
6. When the Student indicates the schedule is complete, the system saves the schedule.

Extensions:

1a. *Student already has a schedule*:

- System brings up the current version of the Student's schedule for editing instead of creating a new one.

1b. *Current semester is closed and next semester is not yet open*:

- System lets Student look at existing schedules, but not create new ones.

3a. *Course Catalog System does not respond*:

- The system notifies the Student and terminates the use case.

5a. *Course full or Student has not fulfilled all prerequisites*:

- System disables selection of that course and notifies the Student.

http://alistair.cockburn.us/Sampler+of+good+and+bad+use+cases

„Fully Dressed“ Use Case nach Cockburn

Use Case <ID>: <Zielbeschreibung mittels kurzem aktivem Satz>

Nutzungskontext: <längere Beschreibung des Ziels und der Umstände>

Design Scope: <welches Teilsystem ist hier gemeint>

Level: Zusammenfassung | User-Ziel | Subfunktion

Primärer Akteur: <Name der Rolle>

Vorbedingungen: <was wird als erfüllt angenommen>

Minimale Garantien: <was wird erfüllt, egal wie der Use Case endet>

Erfolgsgarantien: <was wird im Erfolgsfall erfüllt>

Trigger: <wodurch wird dieser Use Case ausgelöst>

Haupterfolgsszenario:

<Step #>. <Beschreibung der Aktion>

Erweiterungen:

<Step #a>. <Bedingung>: <Beschreibung der Aktion>

Variationen:

<Step #a>. <Beschreibung der Variation>

Anmerkungen:

<was auch immer für die Interpretation / die Implementierung wichtig sein könnte>

- Vieles optional
  - Die Vorlage soll vor allem helfen, nichts zu übersehen
- Absichtlich nicht in Tabellenform
  - Lenkt zu sehr ab, man soll einfach schreiben können
- Anker für alle Info
  - Kommentare
  - Kontext
  - Zeichnungen
  - Interviews
  - …

Good Use Cases

Are …

- text
- no GUI
- no data formats
- 3 – 9 steps in main scenario
- easy to read
- at user's goal level
- a record of decisions made

Are Not…

- UML use case diagrams
- describing the GUI
- describing data formats
- multiple-page main scenario
- complicated to read
- at program-feature level
- a tutorial on the domain

http://alistair.cockburn.us/Sampler+of+good+and+bad+use+cases

Der Inhalt zählt, nicht die Form

- Cockburn ist ein Verfechter des Prinzips, dass es bei den Use Cases vor allem auf möglichst klaren und nützlichen Inhalt ankommt. Daher auch am besten relativ freien Text erlauben.
- Bei zu detaillierten Formularen wird leicht der Eindruck erweckt, der RE müsse etwas „abarbeiten“ oder „ausfüllen“, anstatt die Nutzerziele und Interaktionsabläufe herauszufinden und zu dokumentieren

Fazit: Use Cases

- Strukturierung zusammenhängender Szenarien
- Fokus auf Inhalt, wie soll mit dem System interagiert werden
- Dokumentation auf verschiedenen Detailstufen sinnvoll
- Verschieden „schwere“ Vorlagen existieren
- Möglichkeit, weiterführende Information mitzupflegen

- Use Case Diagramm ≠ Use Case (!)
- Diagramm nur für den Überblick der wichtigsten Akteure Use Cases

User Stories (und Epics)

User Stories

- Ursprung: Extreme Programming* (XP, Ende der 1990er)
  - XP wurde z.B. auch bekannt durch Pair Programming

- Aufgaben von User Stories:
  - Kommunikation zwischen Entwicklern und Stakeholdern
  - Priorisierung und Abschätzung des Entwicklungsaufwands

- Definitionsversuch (Wikipedia*):
  - Eine User-Story ist eine in Alltagssprache formulierte Software-Anforderung. Sie ist bewusst kurz gehalten und umfasst in der Regel nicht mehr als zwei Sätze.

*) https://de.wikipedia.org/wiki/Extreme_Programming

User Story als „mentaler Anker“

User stories are part of an agile approach that helps shift the focus from writing about requirements to talking about them. All agile user stories include a written sentence or two and, more importantly, a series of conversations about the desired functionality.

Mike Cohn *

- Gespräche sind schneller und effektiver als Aufschrieben, Verteilen, Lesen
- Aufgeschriebene Dinge werden oft nicht mehr hinterfragt
- Beim agilen Vorgehen ist es wichtiger, das Problem genau zu verstehen, als es genau aufzuschreiben
- Die Stories sollen darin erinnern, über was man alles sprechen muss, und was man alles entwickeln muss

*) Mike Cohn, https://www.mountaingoatsoftware.com/agile/user-stories

Sprachschablonen für User Stories

As a <type of user>, I want <some goal> [so that <some reason>]. *

Als <Rolle> möchte ich <Ziel/Aktion> [, um <Geschäftsziel/Nutzen>] .

- Formulierung als Wunsch des primären Akteurs
- Im Prinzip eignen sich auch die üblichen Sprachschablonen für funktionale Anforderungen, z.B. nach Pohl/Rupp oder
- Die Schablone dient im wesentlichen dazu, die bereits vorgestellten Verzerrungen durch Sprachtransformation zu vermeiden

*) Mike Cohn, https://www.mountaingoatsoftware.com/agile/user-stories

Epics und User Stories: Beispiele

- Eine Story, die (je nach Kontext) vom Detailgrad als Epic zählen würde:
  - As a user, I can backup my entire hard drive.

- Verfeinerung in eine Liste kleinerer Stories:
  - As a power user, I can specify files or folders to backup based on file size, date created and date modified.
  - As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.
  - ...

*) https://de.wikipedia.org/wiki/Extreme_Programming

Story Card: Die Vorderseite

Als <Rolle> möchte ich <Ziel/Aktion>
[, um <Geschäftsziel/Nutzen>] .

<Priorität> <Aufwand>

Optionale weitere Angaben:

- Titel, Risikobewertung, Ursprüngliche vs. aktuelle Schätzung, ...

Story Card: Die Rückseite

Akzeptanzkriterien:

- Angenommen <Vorbedingung>, wenn <Aktion> dann <Ergebnis>.
- ...

Referenzen auf weiteres Material:

- ...

- Kriterien sollten so konkret sein, dass man daraus automatische Tests ableiten kann (Behaviour Driven Development, Gherkin-Syntax *)

*) Beispiel siehe https://de.wikipedia.org/wiki/Cucumber_(Software)

Das Prinzip der 3 C's

Card — Die Karte, Vorder- und Rückseite

Conversation — Kommunikation, Gespräche zum Inhalt und den Details der Story

Confirmation — Durchführung der Akzeptanztests

Ron Jeffries, Essential XP: Card, Conversation, Confirmation, 2001
http://xprogramming.com/articles/expcardconversationconfirmation/

Kriterien für gute User Stories

I ndependent	unabhängig von anderen Stories
N egotiable	Details sind verhandelbar: erfasse die Essenz der Anforderung
V aluable	liefert Mehrwert
E stimable	Arbeitsaufwand ist gut abschätzbar
S mall	maximal N Story Points / Personentage
T estable	testbar / konkrete Akzeptanzkriterien

Bill Wake, Invest in Good Stories and Smart Tasks, 2003: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks

User Story vs. Anforderungsspezifikation

It's often best to think of the written part as a pointer to the real requirement. User stories could point to a diagram depicting a workflow, a spreadsheet showing how to perform a calculation, or any other artifact the product owner or team desires.

Mike Cohn *

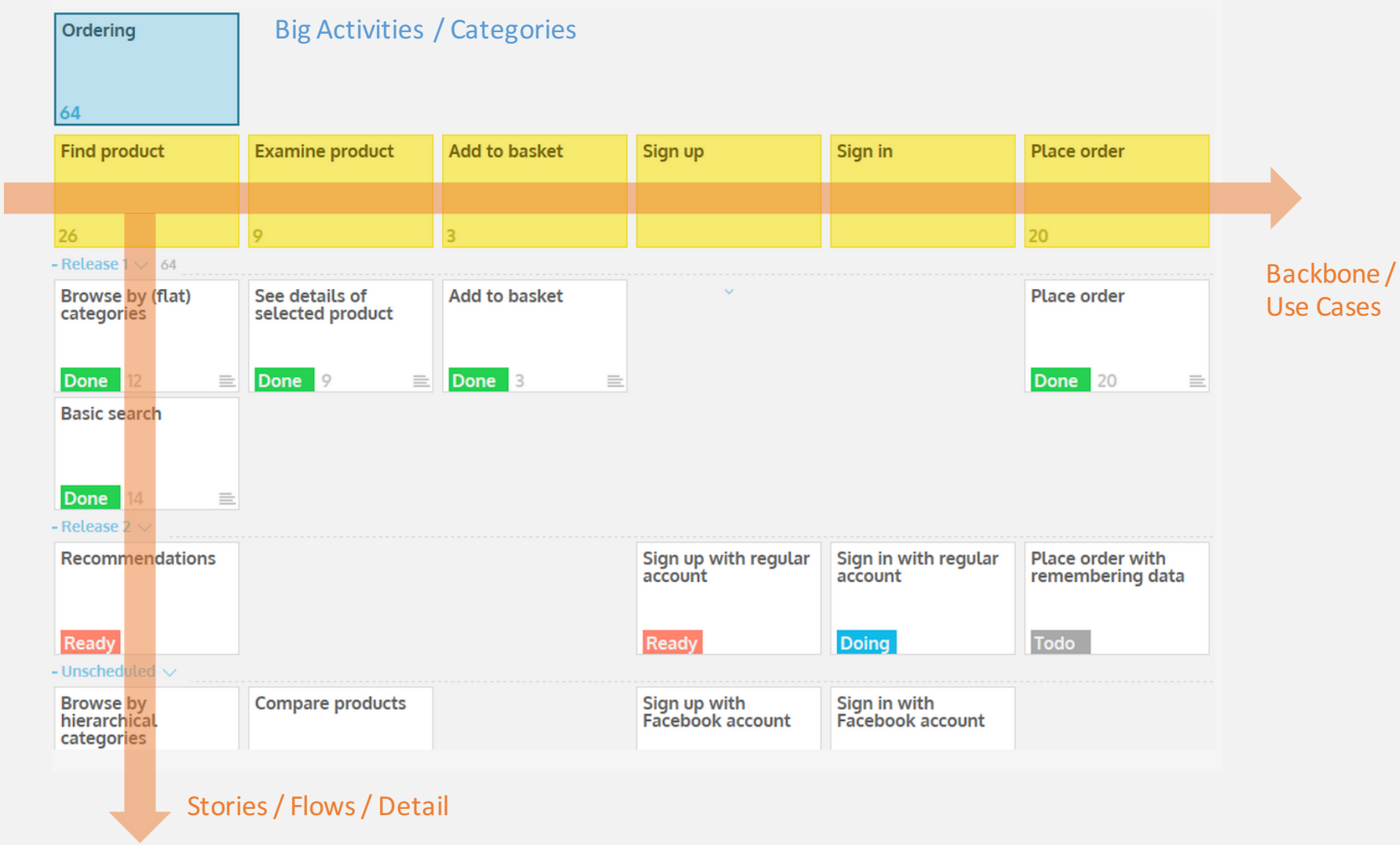
- Meistens ist es nicht ausreichend, nur Stories aufzuschreiben
  - Wenn nicht alle Beteiligten an den Gesprächen teilnehmen können
  - Wenn zu viel Zeit zwischen Gespräch und Implementierung vergeht
  - Wenn der Kontext / Zusammenhang / Überblick verloren geht
  - Wenn der Entwurf einer Lösung nicht trivial ist
  - ...

*) Mike Cohn, https://www.mountaingoatsoftware.com/agile/user-stories

Systemübersicht

Bildmaterial mit freundlicher Genehmigung von Ursula Meseberg, MicroTOOL GmbH

Story Map *



*) Jeff Patton with Peter Economy: *User Story Mapping*, O'Reilly, 2014

Bildmaterial / Beispiel von http://storiesonboard.com/

Fazit: User Stories

- Vor allem für Planung und Priorisierung erdacht
- Verfeinerung von „Epics“ zu implementierbaren „Stories“
- Mentaler Anker zur Repräsentation einer Anforderung
- Beschreibung durch einen Satz
- Ergänzung durch Akzeptanzkriterien

- Strukturierung (im großen) durch Story Maps

- Strukturierung des Inhalts zu einer Story: nicht vorgesehen

Use Case 2.0

Unterscheidung User Story vs. Use Case?

Zu dieser Frage gibt es sehr viele Widersprüchliche Informationen in Literatur und Web. Aus meiner Erfahrung ist die nützlichste und eindeutigste Unterscheidung die folgende von Alistair Cockburn:

„A user story is the *title* of *one* scenario, whereas

a use case is the *contents* of *multiple* scenarios“

A. Cockburn: Why I still use use cases. http://alistair.cockburn.us/Why+I+still+use+use+cases (10/2017)

Warum Use Case 2.0?

Stories

- Sehr gut für agiles Vorgehen: Top-Down-Ansatz, zunächst nur Story-Titel sammeln, bei Bedarf in die Tiefe gehen und verfeinern
- Entwicklung benötigt letztlich (kleine) Stories

Use Cases

- Bieten Zusammenhang, Kontext, Überblick
- Sind geeignetere Einheiten für Änderung und Versionierung
- Sind zu umfangreich für Umsetzung in einem Sprint

Wie kann man beides zusammenbringen?

Use Case 2.0: Sechs Prinzipien

1. Keep it simple by telling stories
2. Understand the big picture
3. Focus on value
4. Build the system in slices
5. Deliver the system in increments
6. Adapt to meet the team's needs

Storytelling zusammen mit konkreten Testcases bilden die Basis.

z.B. mittels Use Case Diagramm, ohne gleich alles auszuarbeiten.

Unterscheidung / Gewichtung Hauptszenario vs. Erweiterungen

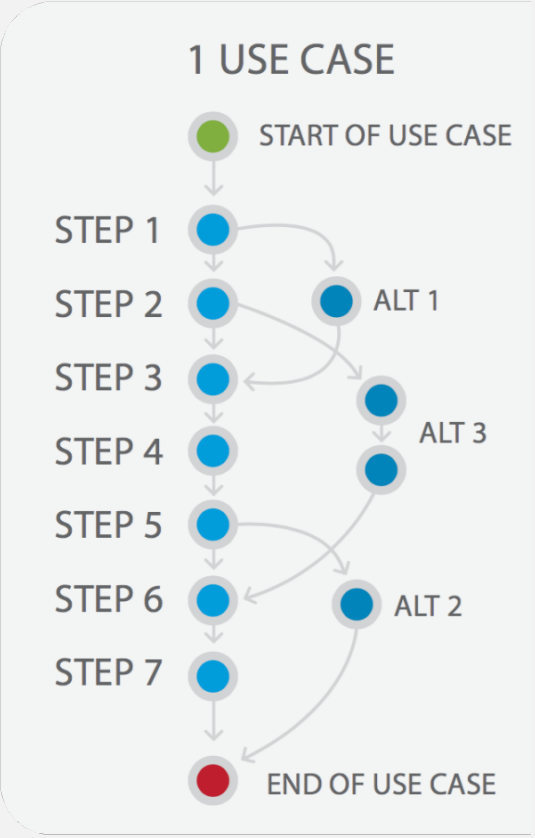
Auftrennen in sinnvolle Teile aus Entwicklungssicht

Agile-inkrementelles Vorgehen: wertvollste Szenarien/Slices zuerst.

I. Jacobson, I. Spence, K. Bittner: *Use-Case 2.0. The Definitive Guide*.

I. https://www.ivarjacobson.com/publications/white-papers/use-case-ebook

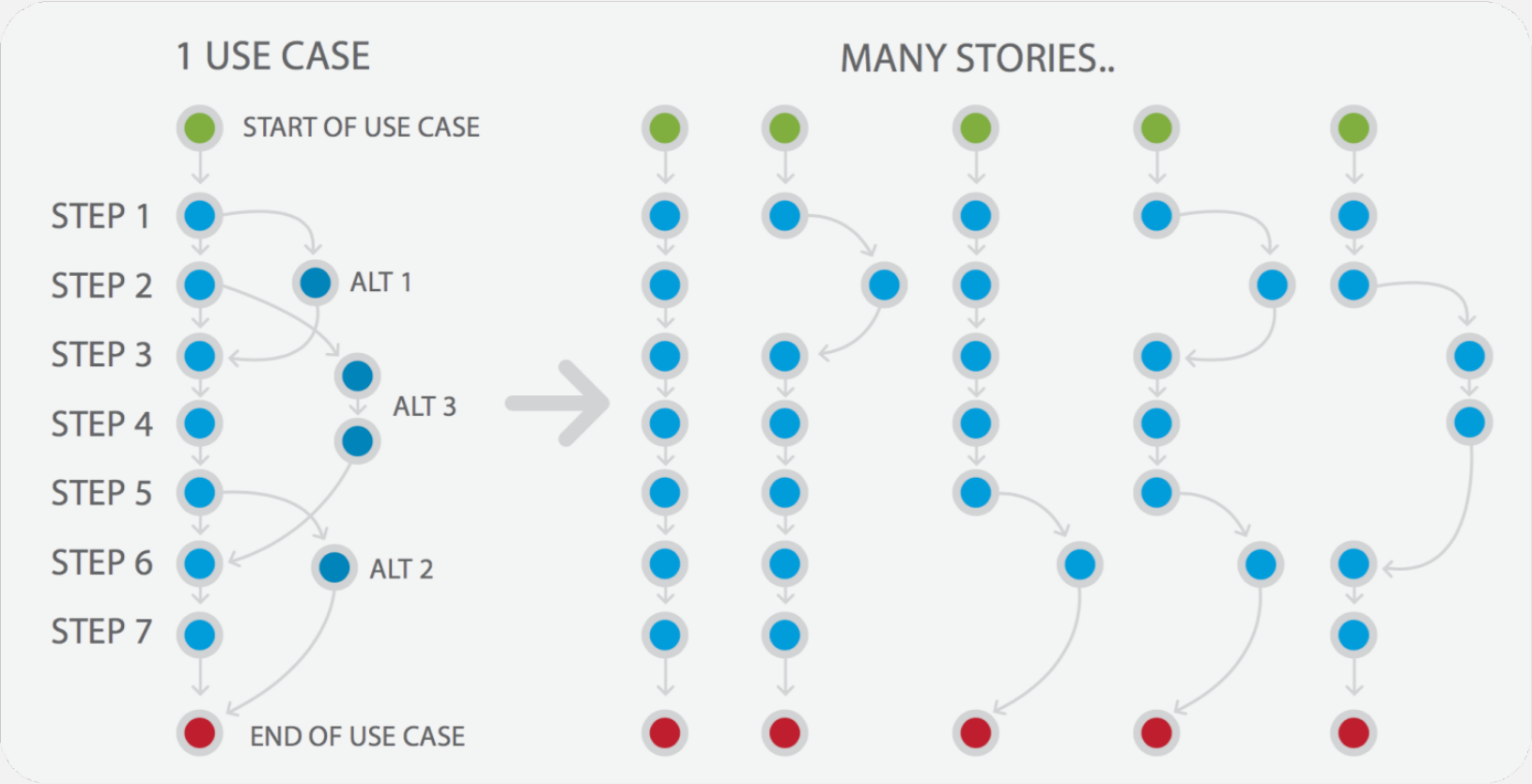
Use Case 2.0: Sammeln mögliche Flows



Für jeden Use Case:

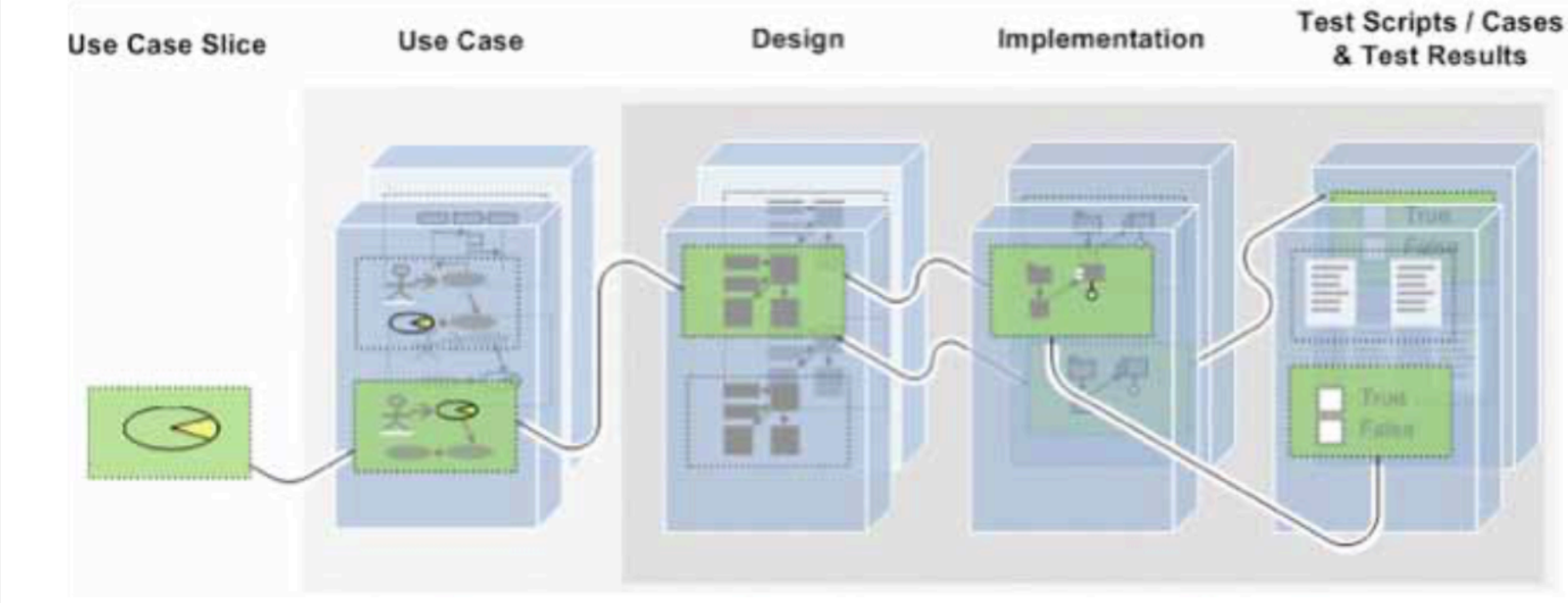
- Sammle mögliche Flows (Abläufe)
- Wie kommt der User zum Ziel
- Und wie nicht (Non-Success Flows)

Use Case 2.0: Story



- Story = Ablauf von Start bis Ende, ein oder mehrere Flows
- + zugehörige Testcase(s)

Use Case 2.0: Slicing



- Beim Aufteilen in entwickelbare „Scheiben“ müssen auch der jeweilige Stand von Architektur, bereits entwickelten Komponenten und Tests berücksichtigt werden.
- Diese Dinge haben oft wesentlichen Einfluss auf die Umsetzungs-Reihenfolge

Bildquelle: I. Jacobson, I. Spence, K. Bittner: *Use-Case 2.0. The Definitive Guide*.

I. https://www.ivarjacobson.com/publications/white-papers/use-case-ebook

Wie soll ich Slices erzeugen?

So, dass

- jeder Slice zu einer funktionierenden Software führt
- jeder Slice einen zusätzlichen Wert schafft
- jeder Slice ein Feedback vom Anwender erzeugen kann
- jeder Slice klein genug ist, um innerhalb eines Sprints umgesetzt zu werden

(Scrum-Grundsätze bzgl. der zu erzeugenden Produktinkremente)

HOW TO SPLIT A USER STORY

1 PREPARE THE INPUT STORY

Does the big story satisfy INVEST* (except, perhaps, small)?

NO → Combine it with another story or otherwise reformulate it to get a good, if large, starting story.

YES → Is the story size 1/10 to 1/4 of your velocity?

→ You're done.

→ Continue. You need to split it.

2 APPLY THE SPLITTING PATTERNS

start here

WORKFLOW STEPS

Does the story describe a workflow?

Can you split the story so you do the beginning and end of the work-flow first and enhance with stories from the middle of the workflow?

Can you take a thin slice through the workflow first and enhance it with more stories later?

DEFER PERFORMANCE

Does the story get much of its complexity from satisfying non-functional requirements like performance?

Could you split the story to just make it work first and then enhance it to satisfy the non-functional requirement?

OPERATIONS

Does the story include multiple operations? (e.g. is it about "managing" or "configuring" something?)

Can you split the operations into separate stories?

BUSINESS RULE VARIATIONS

Does the story have a variety of business rules? (e.g. is there a domain term in the story like "flexible dates" that suggests several variations?)

Can you split the story so you do a subset of the rules first and enhance with additional rules later?

VARIATIONS IN DATA

Does the story do the same thing to different kinds of data?

Can you split the story to process one kind of data first and enhance with the other kinds later?

SIMPLE/COMPLEX

Does the story have a simple core that provides most of the value and/or learning?

Could you split the story to do that simple core first and enhance it with later stories?

MAJOR EFFORT

When you apply the obvious split, is whichever story you do first the most difficult?

Could you group the later stories and defer the decision about which story comes first?

INTERFACE VARIATIONS

Does the story have a complex interface?

Is there a simple version you could do first?

Does the story get the same kind of data via multiple interfaces?

Can you split the story to handle data from one interface first and enhance with the others later?

BREAK OUT A SPIKE

(last resort)

Are you still baffled about how to split the story?

Can you find a small piece you understand well enough to start?

→ Write that story first, build it, and start again at the top of this process.

→ Can you define the 1-3 questions most holding you back?

→ Write a spike with those questions, do the minimum to answer them, and start again at the top of this process

→ Take a break and try again.

3 EVALUATE THE SPLIT

Are the new stories roughly equal in size?

NO → Try another pattern on the original story or the larger post-split stories.

YES → Is each story about 1/10 to 1/4 of your velocity?

→ Do each of the stories satisfy INVEST?

→ Try another pattern.

→ Are there stories you can deprioritize or delete?

→ Try another pattern. You probably have waste in each of your stories.

→ Is there an obvious story to start with that gets you early value, learning, risk mitigation, etc.?

→ Try another pattern to see if you can get this.

→ You're done, though you could try another pattern to see if it works better.

* INVEST - Stories should be:
Independent
Negotiable
Valuable
Estimable
Small
Testable

AGILE FOR ALL
www.agileforall.com

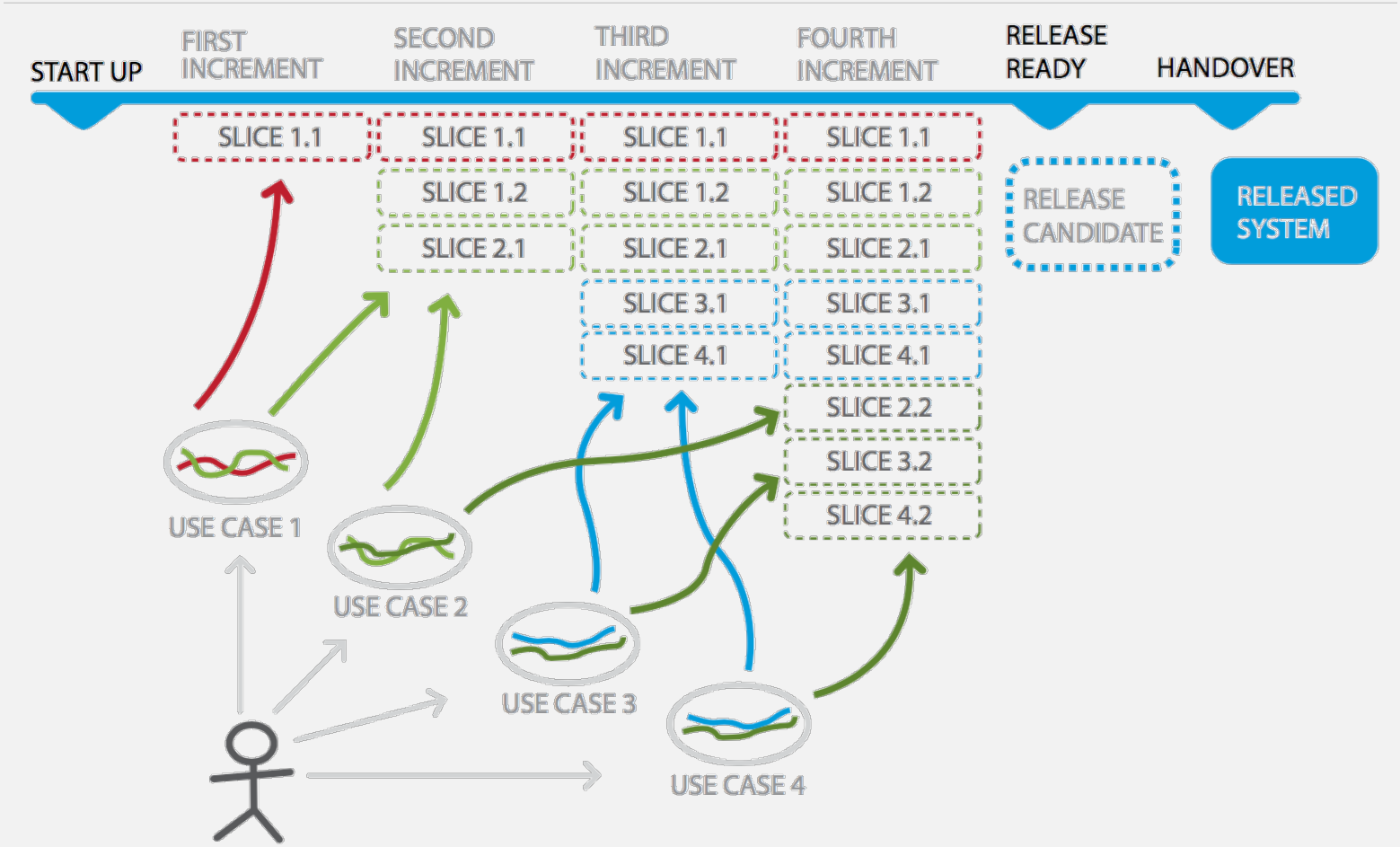
Visit http://www.richardlawrence.info/splitting-user-stories/ for more info on the story splitting patterns

Copyright © 2011-2013 Agile For All. All rights reserved.

Last updated 3/26/2013

Richard Lawrence 2009, http://agileforall.com/patterns-for-splitting-user-stories/

Inkrementelle Entwicklung und Auslieferung



Bildquelle: I. Jacobson, I. Spence, K. Bittner: *Use-Case 2.0. The Definitive Guide*.

https://www.ivarjacobson.com/publications/white-papers/use-case-ebook

Use Cases vs. User Stories
–
was tun?

Warum sind Use Cases so unbeliebt?

- Analyse / Vereinfachung / Modellierung zu früh bzw. überbetont
  - Beeinflusst durch UML, welche kaum auf den Inhalt von Use Cases eingehen, eher auf das Diagramm und die Beziehungen
- Wenn man nur den reinen Inhalt des Use Cases betrachtet, gehen viele Details verloren, die z.B. beim UI Design wieder wichtig wären
  - Details eines Beispielszenarios
  - Welches Ziel / welcher Wunsch treibt den Benutzer gerade?
  - Wie fühlt sich der User bei welchem Schritt / bei welcher Reaktion ?
- Daher die Ablehnung der Use Cases in den Anfängen der agilen Entwicklung (Extreme Programming)
  - Lieber gleich nur Stories sammeln
  - Keine „unnötige“, langsame Modellierung und Analyse
  - Die wahren Details hat man (wer?) im Kopf, man kann nicht alles aufschreiben. Diese Details sind wichtig, auch wenn sie nicht aufgeschrieben sind!

Diese Aussagen stellen die persönlich Meinung / Erfahrung des Dozenten dar!

Cockburns Kritik an User Stories

- Writing good use cases (or any other requirements) requires thinking, communicating, and thinking again. It is much easier just to write user-story tags on index cards and let the project blow up later.
- If you don't want to think, find a new profession. Software development requires thinking, at all levels.

A. Cockburn: Why I still use use cases. http://alistair.cockburn.us/Why+I+still+use+use+cases (10/2017)

Wie macht man es denn nun richtig (1/2)?

- Sammle und visualisiere am Anfang Szenarien
  - Erfasse die Gesamtheit der
  - Ob Du die Überschriften „Stories“ nennst, ist egal
  - Eine Story Map ist eine schöne Übersicht, und erfasst mehr Detail als ein Use Case Diagramm
- Arbeite die Szenarien aus
  - Um zu prüfen, ob Du ein Szenario verstanden hast, schreibe die essenziellen Schritte davon auf.
  - Es kommt hier auf den Inhalt an, nicht auf die Form. Wie gut kann man sich in die Interaktion hineinversetzen?
  - Akzeptanzkriterien: sind immer wichtig, egal ob Use Case oder User Story
- Raum für Kontext, freie Kommentare und Beispiele lassen
  - Links und Text im Kommentarfeld des Use Cases

Wie macht man es denn nun richtig (2/2)?

- Lagere große redundante Teile in eigene Szenarien aus
  - Siehe „include“-Beziehung bei Use Cases
- Verfeinere Deine Szenarien bis zur Implementierbarkeit
  - So gut ausarbeiten, dass man auf dieser Basis ein UI entwerfen könnte
  - Gegeben ein Screen Design: durchlaufe alle Use Cases, um zu testen, wie angemessen das Design für die gewünschte Interaktion ist.
- Slice Deine Szenarien ggf. so, dass sie in einen Sprint passen
  - Slicing ähnlich wie in Use Case 2.0
  - Schätzung sollte leicht möglich sein und unter 10 (besser: bis zu 5) Personentagen liegen
  - Demonstration und Tests gemäß der Akzeptanzkriterien
  - Verlinke Deine Slices zu den Use Cases mit allen Informationen und mit den verwandten Flows / Szenarien

Darstellung und Tools?

- Story Map für die funktionalen / Interaktionsanforderungen erstellen
  - Leichtgewichtig, schnell
  - Gut zur Kommunikation
  - Storiesonboard * funktioniert ganz gut
- Nichtfunktionale Anforderungen sammeln!
  - Sind orthogonal zu den Stories / Use Cases
- Spezifikation für Planung (Architektur, Releases)
  - Spezifikationsdokumente, Modellieren von Zuständen, Prüfung auf Vollständigkeit funktioniert besser in großen Einheiten (Use Cases) als in kleinen Stories
- Sprint-Planung und Durchführung, Bugtracking:
  - Slicen und Priorisierung der Use Cases / Szenarien aus Entwicklungssicht
  - Backlog gefüllt mit Sprint-gerechten Slices
  - Schätzung, Tracking des Status, alles jeweils für die nächsten 2-3 Sprints
  - Tools wie Jira

*) http://storiesonboard.com/

Fragen?