UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Software Entwicklung & Programmierung

Requirements Engineering

PALUNO
The Ruhr Institute for Software Technology

Disclaimer

Bilder und Texte der Veranstaltungsfolien und -unterlagen sowie das gesprochene Wort innerhalb der Veranstaltung und Lehr-Lern-Videos dienen allein dem Selbst- bzw. Gruppenstudium. Jede weiterführende Nutzung ist den Teilnehmenden der Moodle-Kurse untersagt, z.B. Verbreitung an andere Studierende, in sozialen Netzwerken, dem Internet!

Darüber hinaus ist ein studentischer Mitschnitt von Webkonferenzen im Rahmen der Lehre nicht erlaubt.

Zielsetzung

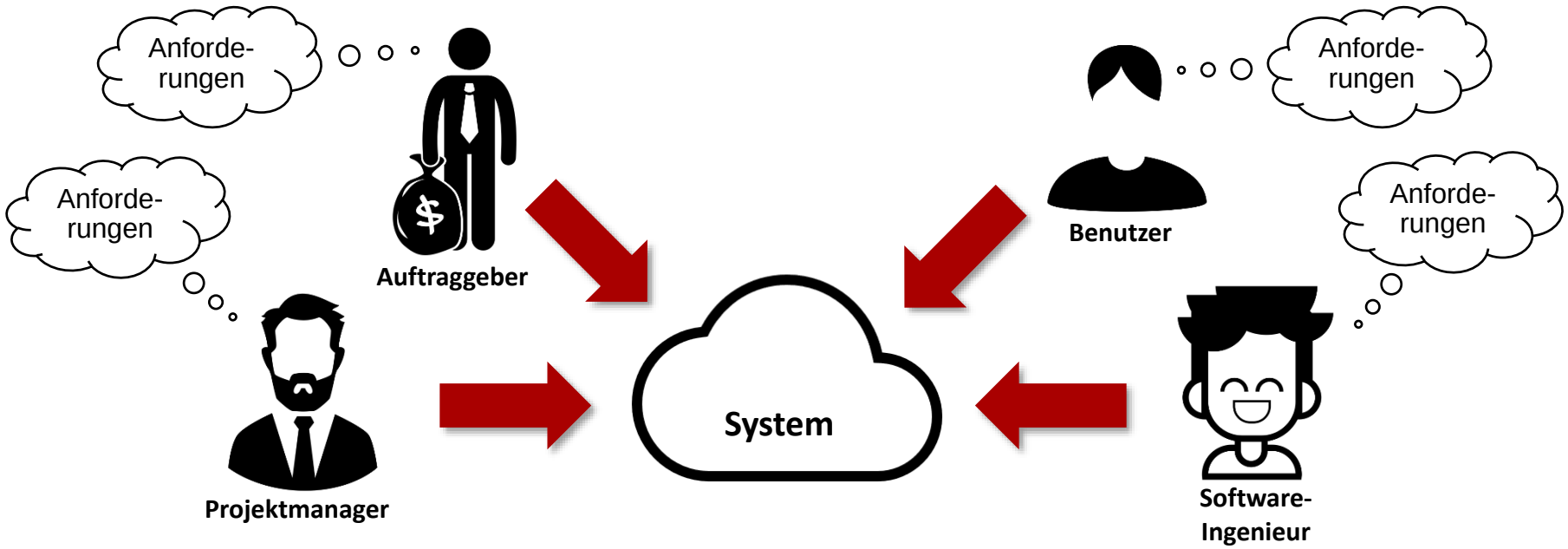
- Am Ende dieser Lerneinheit könnt Ihr:
  - Anforderungen eines Software-Produktes erheben
  - Anforderungen an ein Software-Produkt in Form von User Stories dokumentieren
  - Szenarien zu den Anforderungen eines Software-Produktes erarbeiten
  - Szenarien in Form von basic Message Sequence Charts dokumentieren
  - Papierprototypen erstellen

Agenda

1. **Grundlagen des Requirements Engineering**
2. User Stories
3. Szenarien
4. Papierprototypen

Motivation für das Requirements Engineering

- **Requirements Engineering** = ingenieursmäßige Herangehensweise an **Anforderungen** eines zu entwickelnden Software-Systems
- Im Requirements Engineering werden die Wünsche der Stakeholder gewonnen, dokumentiert und abgestimmt, um als Basis für die weitere Entwicklung zu dienen.



Probleme durch schlechtes RE (1/2)

- Anforderungen **fehlen** oder sind **unvollständig**
  → Kundenwünsche bleiben unerfüllt

- Verschiedene Nutzer haben unterschiedliche Sichten
  → Anforderungen werden **missverstanden**

- **Zusätzliche** Anforderungen werden umgesetzt (“Gold Plating”)
  → unnötiger Aufwand und ggf. ungewollte Nebeneffekte

- → Das implementierte **System erfüllt seinen Zweck möglicherweise nicht**!

Probleme durch schlechtes RE (2/2)

- Viele Projekte **scheitern** oder führen zu **unvollständiger Software** und/oder **Budgetüberschreitungen** aufgrund von fehlerhaftem Requirements Engineering [Chaos Report 1995]

- **Fehler in den Anforderungen** sind im weiteren Projektverlauf weitaus **kostspieliger zu beheben** als im RE
  - **Faktor 5** für erst im Akzeptanztest aufgedeckte Anforderungsfehler [Boehm und Basili 2001]

Agenda

1. Grundlagen des Requirements Engineering
2. **User Stories**
3. Szenarien
4. Papierprototypen

User Stories - Motivation und Einordnung

- User Stories dienen zunächst dazu, natürlichsprachliche Anforderungen **strukturiert** niederzuschreiben
  - Einfaches und leicht zu erlernendes Hilfsmittel
- User Stories **unterstützen** jedoch auch explizit…
  - … die **Kommunikation** über den Inhalt der Stories
    - Fokus auf Nutzer**ziele**, weniger auf mögliche Lösungen
  - … die Konzeption **initialer Tests** als Detaillierung der User Stories und zu deren Validierung

User Stories - Notation

- Satzschablone für User Stories [Cohn 2004]:

  Als *<Rolle>* möchte ich *<Ziel>, um/sodass <Nutzen>*.

- Beispiele:

  Begründung

  "Als Arzthelferin möchte ich eine Meldung erhalten, wenn ein Patient fünf Minuten vor einem vereinbarten Termin nicht angemeldet ist, um den Arzt rechtzeitig zu informieren."

  "Als Arzt möchte ich rechtzeitig informiert werden, wenn ein Patient einen Termin nicht wahrnimmt, sodass ich andere Patienten vorziehen kann."

Eigenschaften „guter“ User Stories (INVEST)

I – Independent: Möglichst wenig Abhängigkeiten und Überlappungen zwischen Stories, um Planungssicherheit zu gewährleisten und die Priorisierung zu unterstützen

N – Negotiable: Anregung der Kommunikation anstelle von präskriptiven, zu detaillierten und einschränkenden Aussagen

V – Valuable: Vermeidung von Aussagen, die nur für Entwicklung einen “Wert” haben, keine unnötigen Lösungsdetails

E – Estimatable: Genug Informationen zur Prioritätsbestimmung und Implementierungsreihenfolge

S – Small: Realisierungsaufwand und -zeit überschaubar

T – Testable: Tests zur Überprüfung der Realisierung der User Story definierbar

Gute und schlechte User Stories (1)

- Schlechte User Story:

  "Als Arzthelferin möchte ich Medikamente bestellen, die in der Praxis vorrätig sein müssen, sowie Verbrauchsberichte an Krankenkassen übermitteln können."

  → **Nicht small!**

- Korrektur (Aufteilung in 2 User Stories):

  "Als Arzthelferin möchte ich Medikamente bestellen können, um den nötigen Vorrat zu gewährleisten."

  "Als Arzthelferin möchte ich Berichte über den Medikamentenverbrauch an Krankenkassen schicken können."

Gute und schlechte User Stories (2)

- Schlechte User Story:

  "Als Arzthelferin möchte ich Patientendaten schnell abrufen können, um die Wartezeit der Patienten zu reduzieren."

  → **Nicht testable!**

- Korrektur (Präzisierung):

  "Als Arzthelferin möchte ich Patientendaten nach Abruf angezeigt bekommen, um die Wartezeit der Patienten zu reduzieren."

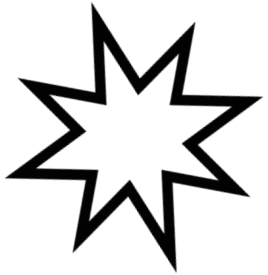
Agenda

1. Grundlagen des Requirements Engineering
2. User Stories
3. **Szenarien**
4. Papierprototypen

Motivation für Szenarien

- Bisher: User Stories zur Dokumentation von **Nutzerzielen** und **Begründungen** für geforderte Systemfunktionen und Qualitätseigenschaften
  - Sehr hohe Abstraktionsebene, wenig Details

- Ausreichend zur **Implementierung** und zur Definition konkreter **Tests**?
  - **Wie** werden die Nutzerziele erreicht?
  - Konkrete, **beispielhafte** (Nutzungs-)**Abläufe**?
  - Alle **Bedingungen** und **Ausnahmesituationen** erfasst?
    → “Kann mein Ziel auch **nicht** erfüllt werden? Wenn ja, wie?”
  - Grundlage für Systemtests: Lässt die Implementierung die Ausführung der geplanten Abläufe zu?



Ergänzung durch **Szenarien**!

➔ Spezifikation von Szenarien **zu jeder** User Story

Verschiedene Abstraktionsgrade in Szenarien

- Szenarien beschreiben **konkrete Abläufe**, die illustrieren, wie **Nutzerziele erreicht** oder **nicht erreicht** werden können

- Verschiedene **Klassifikationskriterien** für Szenarien, u.a. [Rolland et al. 1998]:
  - Unterscheidung bzgl. Kontext
    - **Systeminterne Szenarien**: Systeminterne Abläufe, Interaktion zwischen Systembestandteilen
    - **System-Kontext-Interaktionen**: Interaktionen zwischen externen Entitäten im Kontext (d.h. der Systemumgebung, bspw. Systemnutzer) und dem System

Fokus!

- In SEP wird ausschließlich die Interaktion des Nutzers mit dem System modelliert
  - Systeminterne Szenarien werden **nicht** betrachtet

Drei Arten von Szenarien

- **Hauptszenario**: Typische Folge von Interaktionsschritten zur Erreichung des Ziels bzw. der Ziele einer oder mehrerer User Stories

- **Alternativszenario**: Alternative Interaktionsschritte, die das Hauptszenario (oder Teile davon) ersetzen und ebenfalls zur Erfüllung der Ziele hinter den User Stories führen

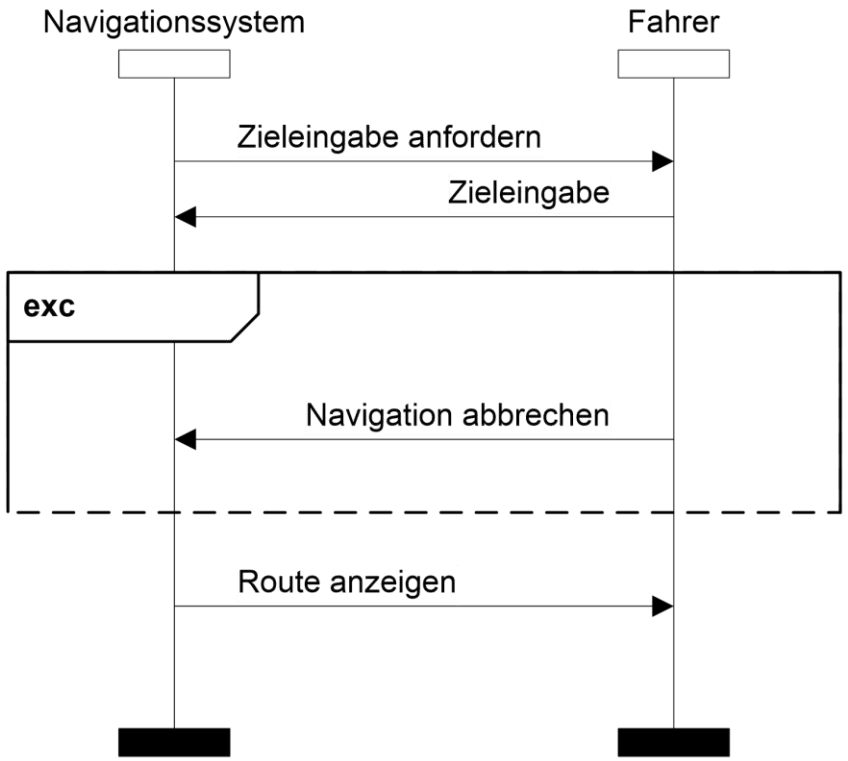
- **Ausnahmeszenario**: Interaktionen in Ausnahmefällen, die dazu führen, dass nicht alle Ziele hinter den abgebildeten User Stories erreicht wird

➔ Zu **jeder User Story** sollte **mindestens ein Szenario** spezifiziert sein!

Grafische Modellierung von Szenarien

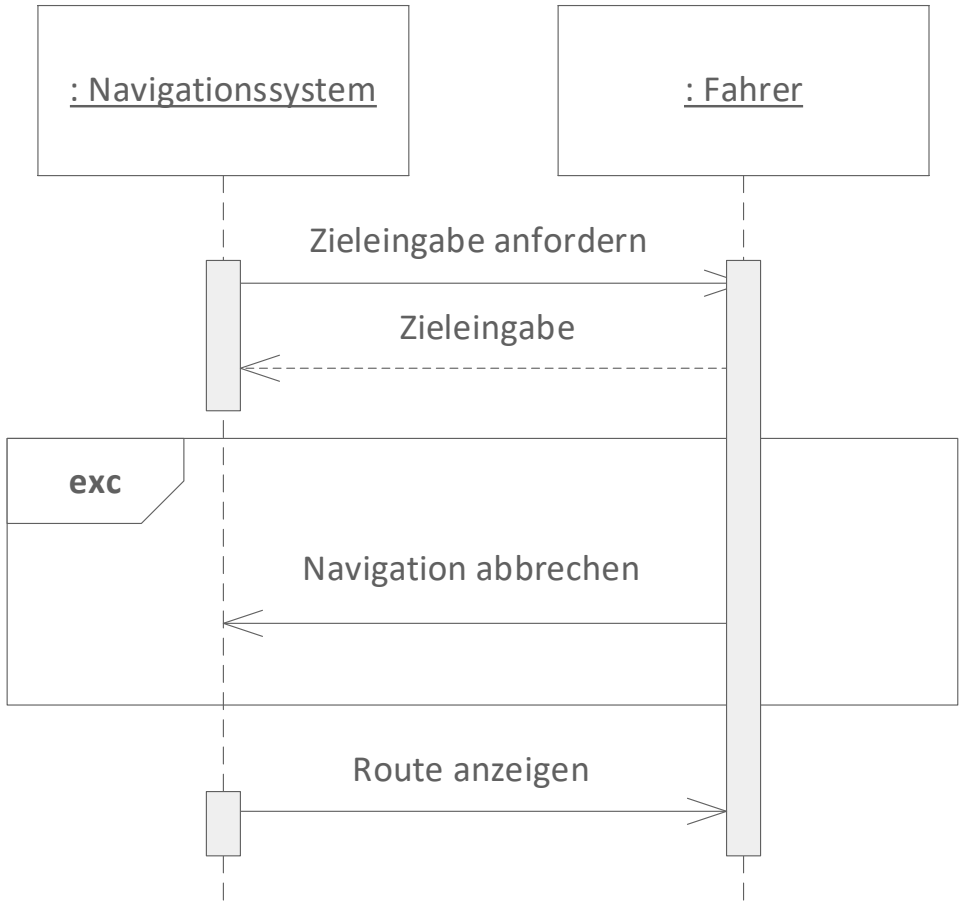
Basic Message Sequence Charts (bMSC)

[ITU 2011]



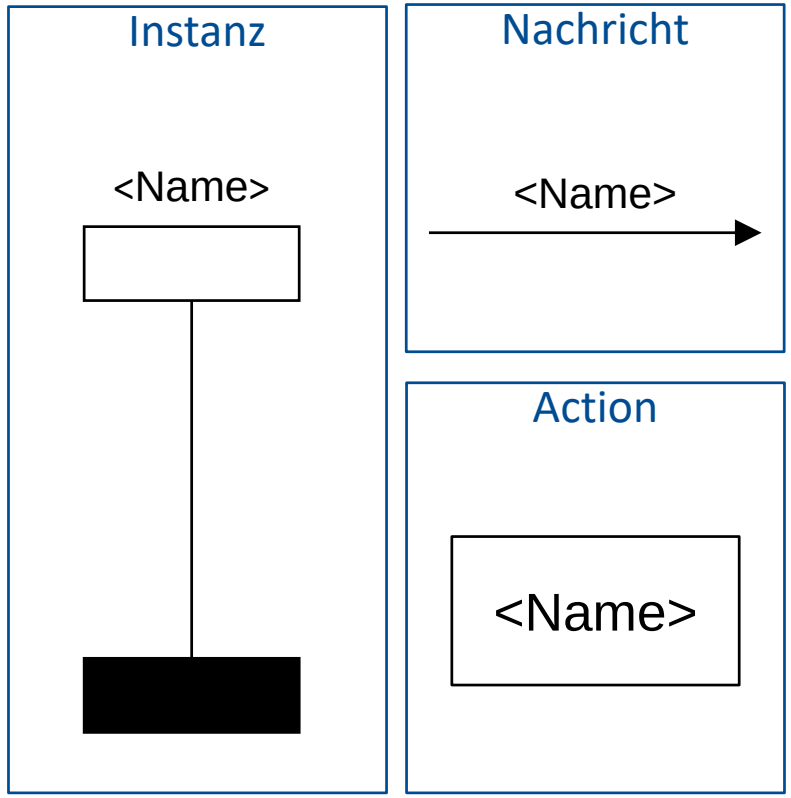
Fokus!

UML-Sequenzdiagramme



bMSC - Konstrukte und Notation (1)

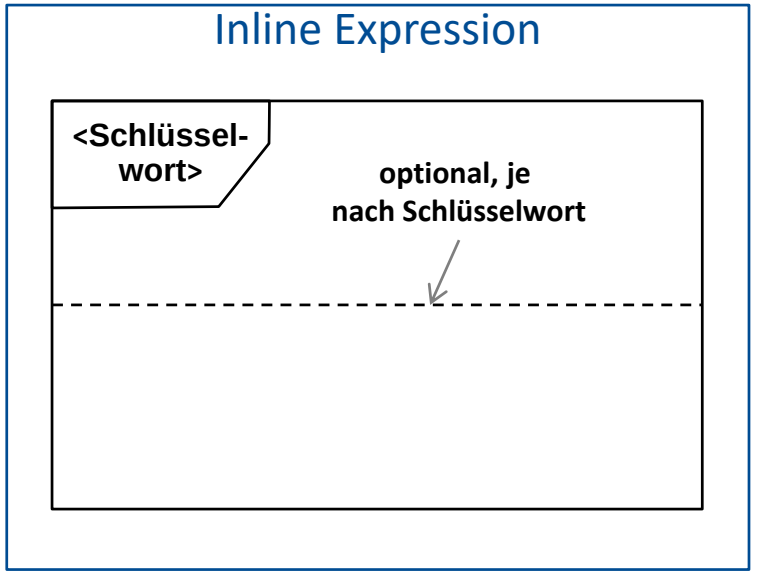
- **Instanz**: beschreibt eine Entität (in unserem Fall: den/die Nutzer und das zu entwickelnde System)
  - Vertikale Dimension: **Zeit**
- **Nachricht**: wird zwischen zwei Instanzen (Sender und Empfänger) übertragen
- **Action**: beschreibt eine interne Aktivität einer Instanz



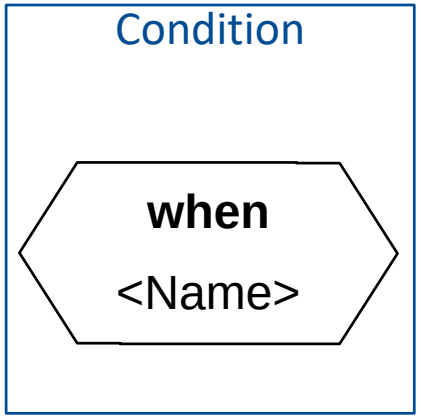
Basiert auf [ITU 2011]

bMSC - Konstrukte und Notation (2)

- **Inline Expression**: beschreibt verschiedene Strukturen innerhalb eines bMSCs
  - Schlüsselwörter:
    - **opt**: optionale Interaktionen
    - **alt**: alternative Interaktionen
    - **par**: parallele Interaktionen
    - **loop**: wiederholte Interaktionen (Schleifen)
    - **exc**: Ausnahmefälle

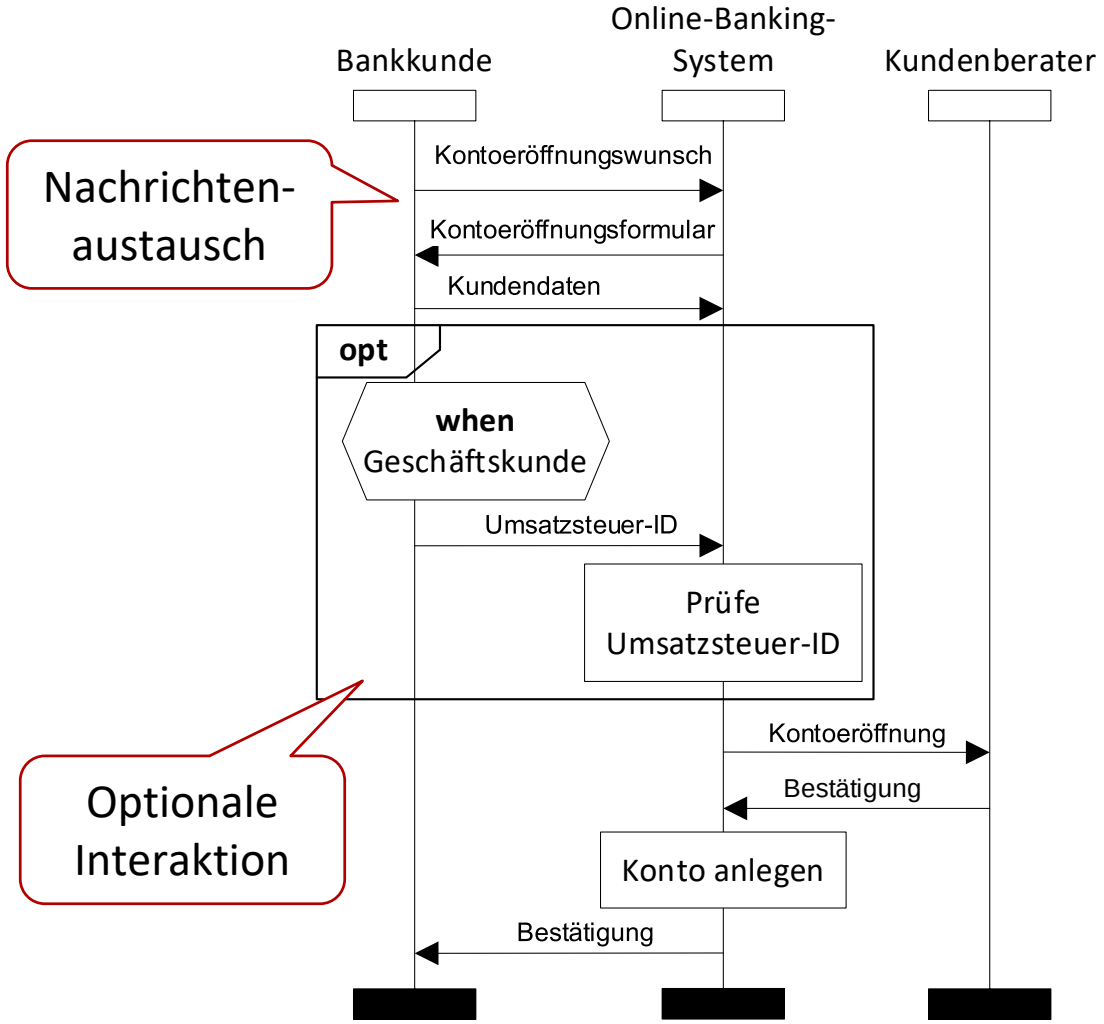


- **Condition**: beschreibt Bedingung, wann Bestandteile einer Inline Expression ausgeführt werden
(optional, kann auch durch Nachrichten gesteuert sein)

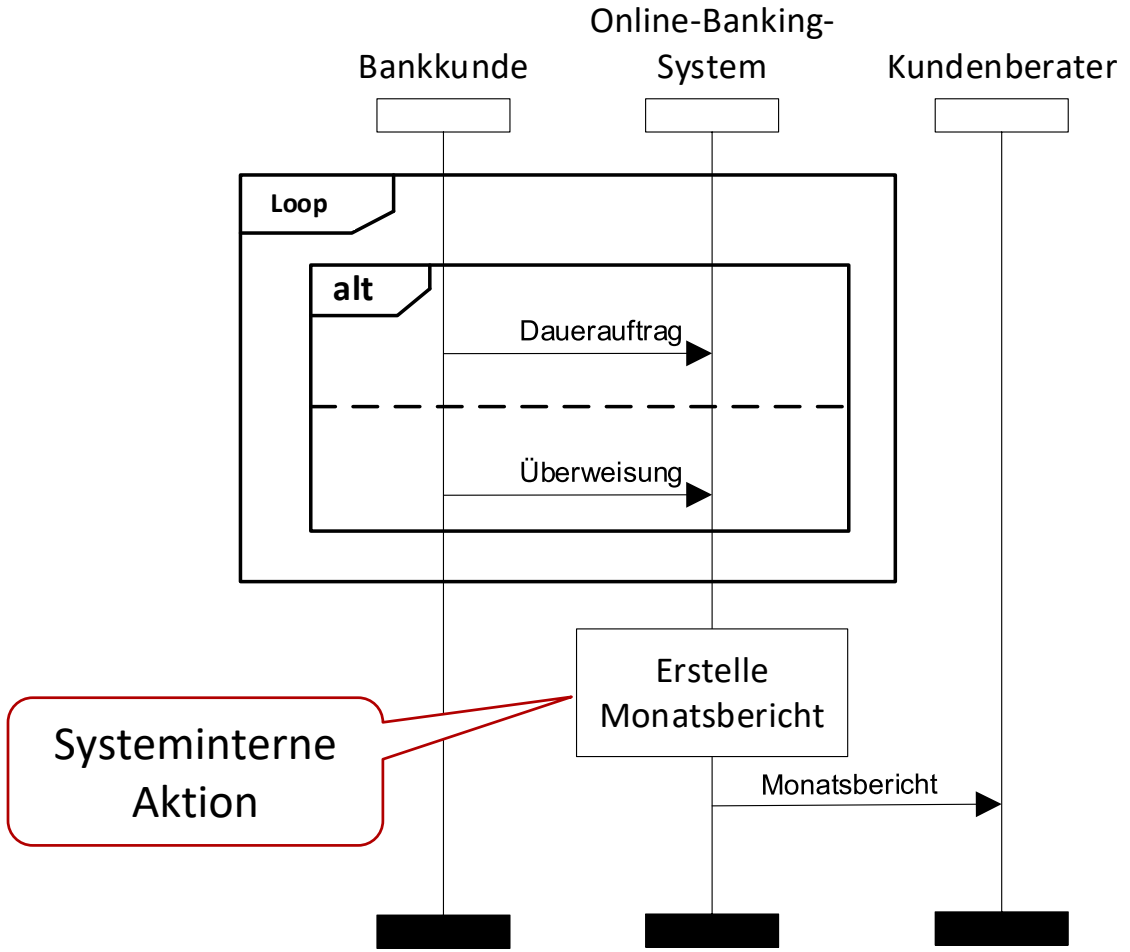


bMSC - Beispiele

Hauptszenario zur User Story „Konto eröffnen“



Hauptszenario zur User Story „Monatsbericht erstellen“

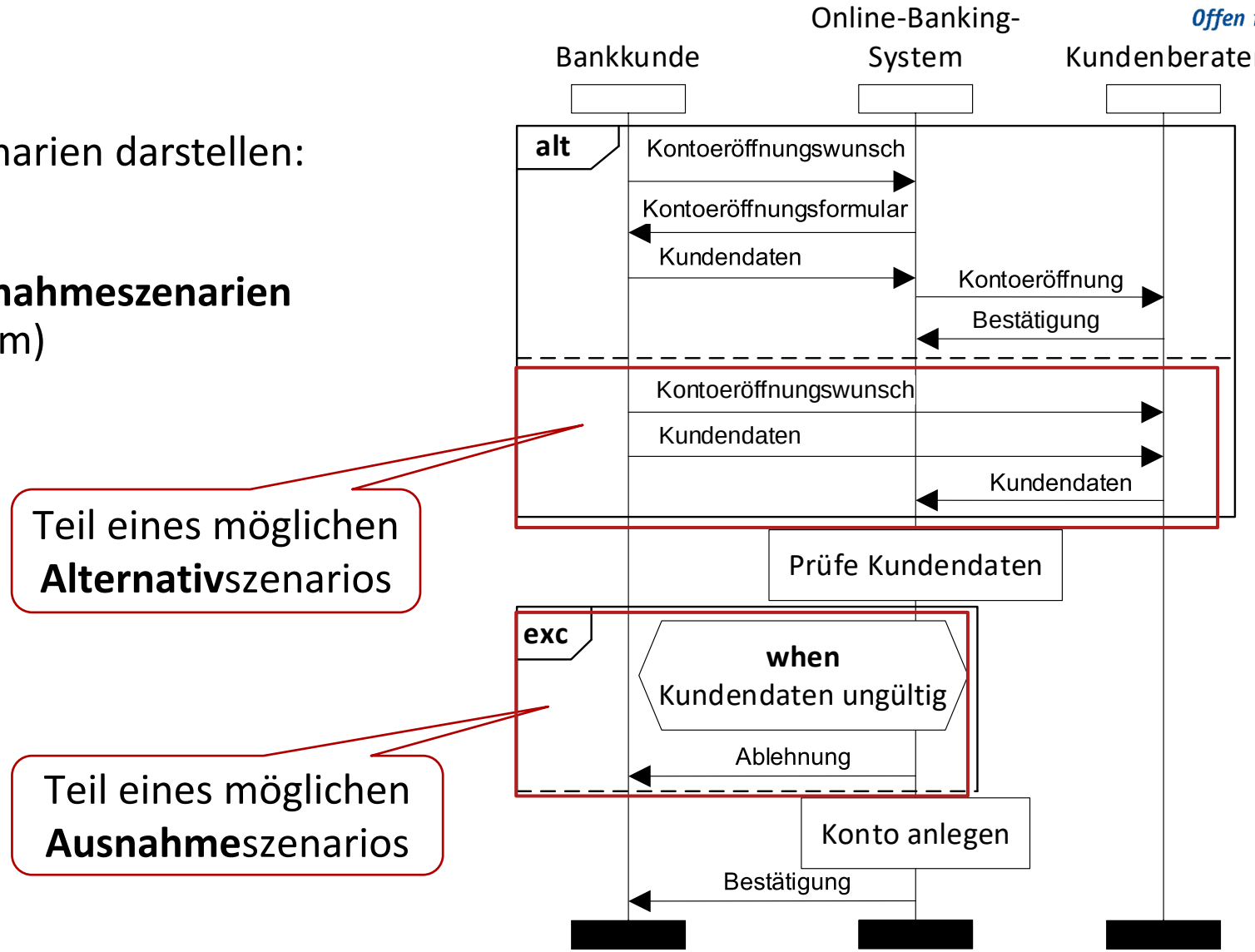


bMSC - Haupt-, Alternativ- und Ausnahmeszenarien

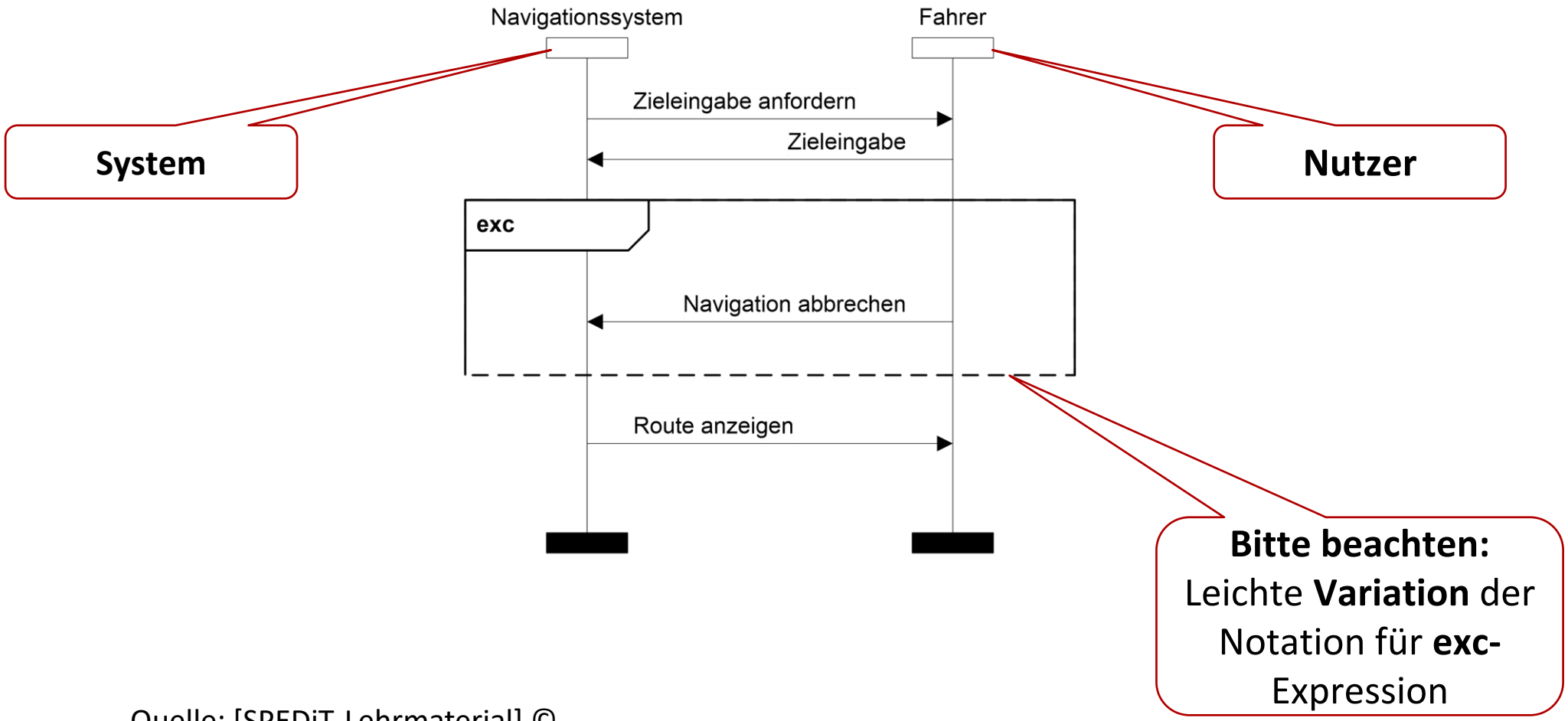
- Ein bMSC kann mehrere Szenarien darstellen:
- **Haupt-**, **Alternativ-** und **Ausnahmeszenarien** integriert (in einem Diagramm)

➔ Alternativ:

mehrere bMSCs

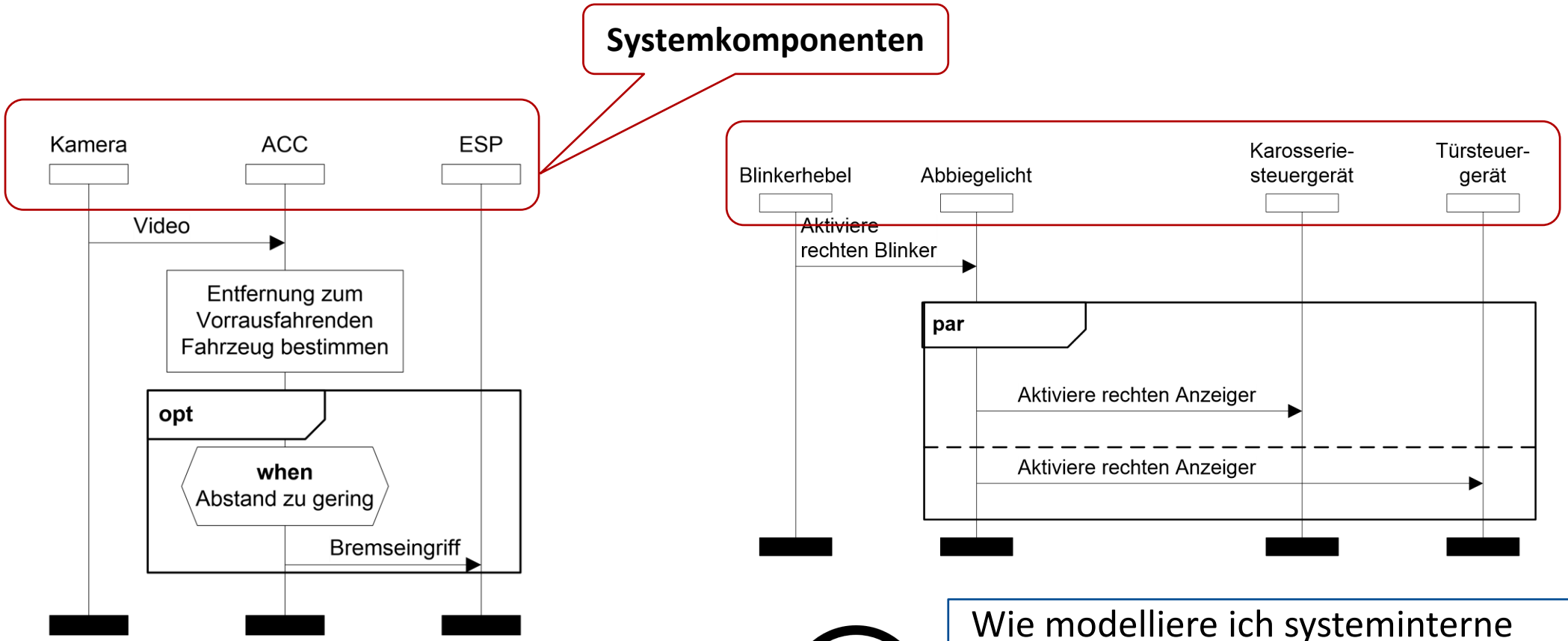


Interaktion zwischen System und Nutzer



Quelle: [SPEDiT-Lehrmaterial] ©

Exkurs: bMSCs können auch systeminterne Interatkionen dokumentieren



Quelle: [SPEDiT-Lehrmaterial] ©

Wie modelliere ich systeminterne Aktionen, ohne das System zu dekomponieren?
→ **Actions (siehe bspw. Folien 19 und 21)**

Modellierungstools für bMSCs

- Zur grafischen Modellierung von MSCs findet Ihr eine **Visio-Schablone** im moodle-Kurs
- Alternativ empfehlen wir für die Modellierung auch **draw.io** (kein Download notwendig)
  - http://draw.io/
- Eine Studentenlizenz für **Microsoft Visio** sowie das Programm zum Herunterladen erhaltet ihr kostenlos

Optionale Übungsaufgabe: Erstellung eines bMSC zu folgendem Szenario

User Stories:
„Als Nutzer möchte ich mich am System anmelden können, um Fremdzugriffe auszuschließen.“

Hauptszenario “Nutzeranmeldung”:

1. Nutzer gibt seine Nutzerkennung und PIN ein.
2. System prüft Gültigkeit der Nutzerkennung und der PIN per Aufruf des Authentifizierungsservers.
3. Authentifizierung meldet Gültigkeit der Nutzerdaten.
4. System vermerkt Anmeldungszeitpunkt.
5. Nutzer ist am System angemeldet.

Alternativszenario:

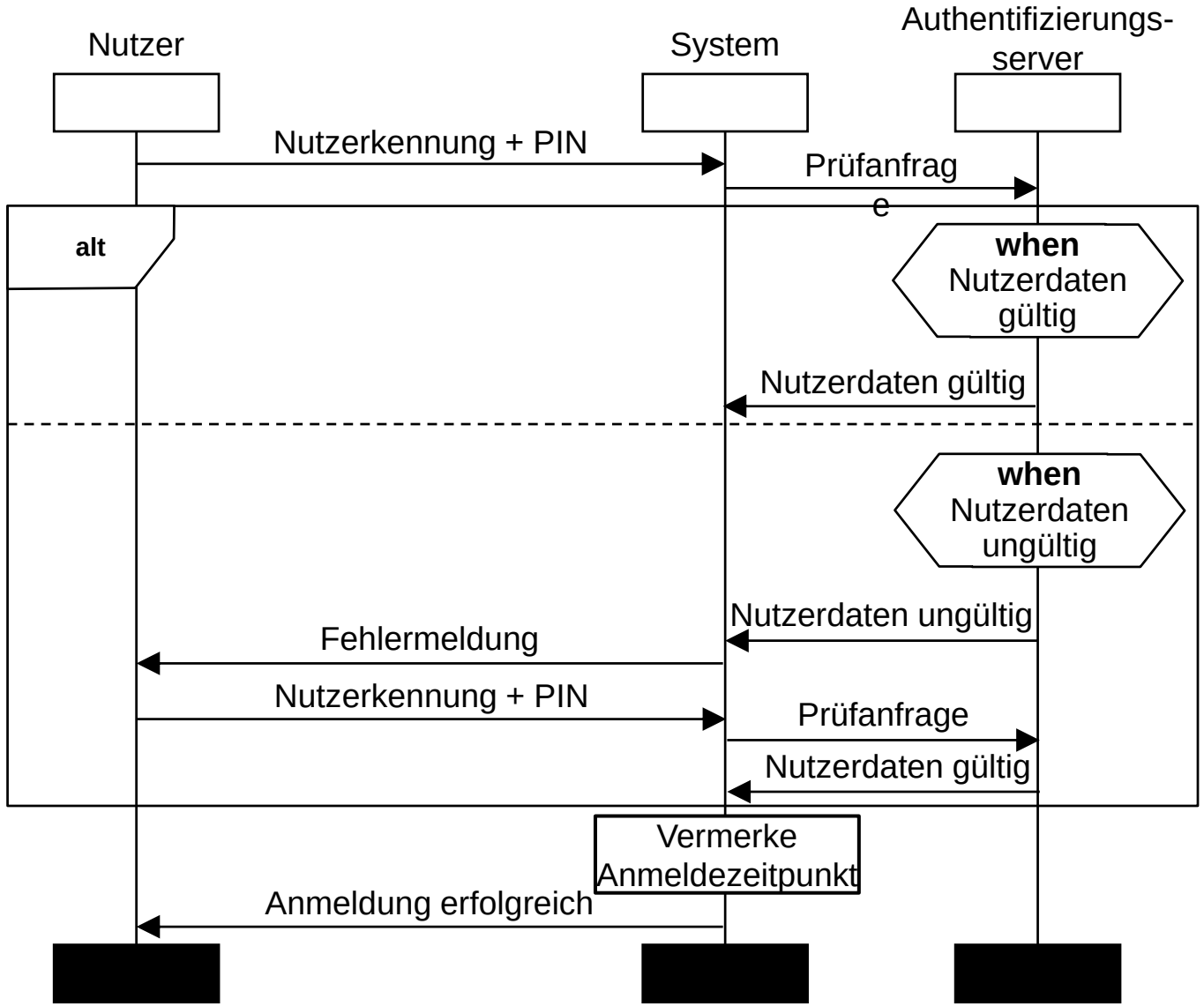
3a. Nutzerkennung oder PIN sind nicht korrekt.

3a1. System gibt Fehlermeldung aus.
3a2. Nutzer gibt seine Nutzerkennung und PIN nochmals ein.
3a3. System prüft Gültigkeit der Nutzerkennung und der PIN per Aufruf des Authentifizierungsservers.
3a4. Authentifizierung meldet Gültigkeit der Nutzerdaten.

(weiter mit Schritt 4 des Hauptszenarios)

Mögliche Lösung auf der nächsten Folie

Mögliche Lösung zur Übungsaufgabe



Dieses Beispiel enthält sowohl Interaktionen zwischen System und Nutzer als auch systeminterne Interaktionen

Agenda

1. Grundlagen des Requirements Engineering
2. User Stories
3. Szenarien
4. **Papierprototypen**

Was ist ein Papierprototyp?

„Im weitesten Sinne kann das Erstellen eines Papierprototypen als Methode des Brainstormings, Designs, Herstellens, Testens und des Kommunizierens von User Interfaces (UI) aufgefasst werden.“

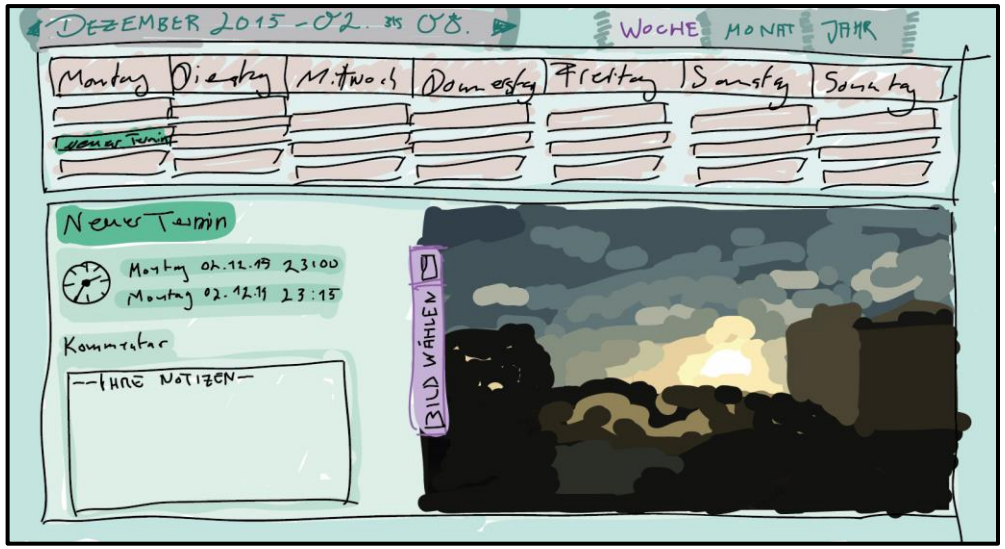
[* Carolyn Snyder: „Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces“ 2003]

Ablauf:

1. Auswahl typischer Abläufe
2. Skizzieren verschiedener Fenster, Menüs, etc.
3. Durchführen eines Benutzbarkeitstests

Vorteile von Papierprototypen

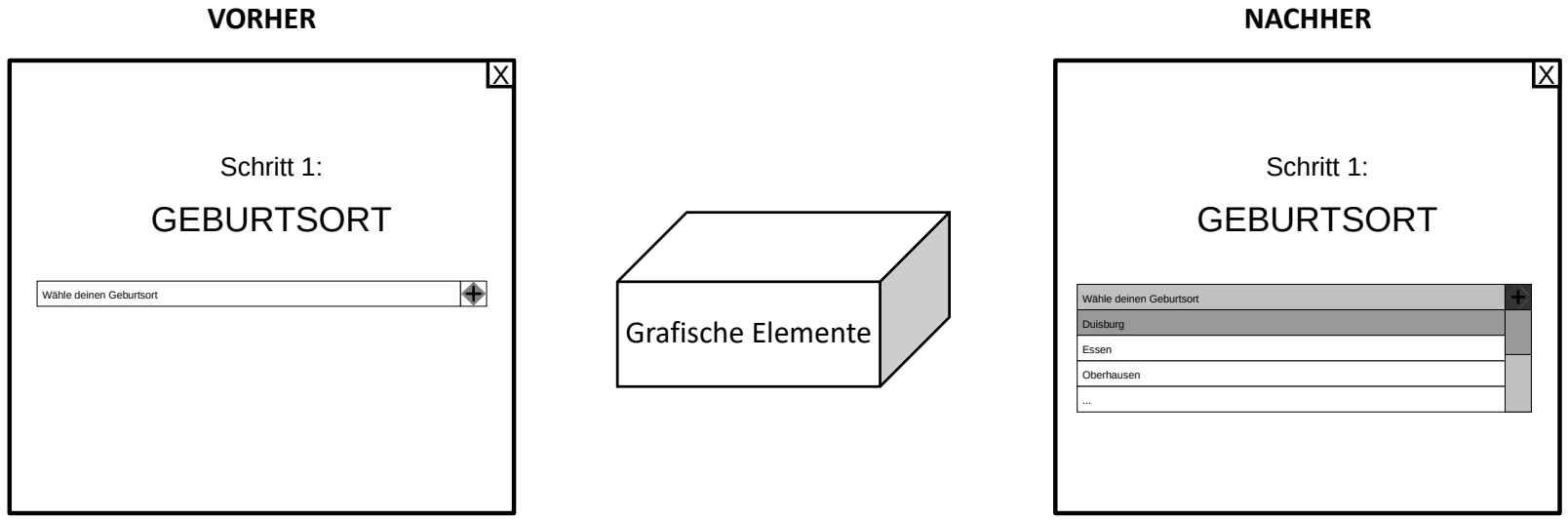
- Hilft bei der **Ideenkommunikation** im Entwicklungsteam
- Benötigt keine technischen Kenntnisse
- Ermöglicht die Probe vieler verschiedener **Designideen**
- Fördert die **Kreativität** im gesamten Team



Papierprototyp einer Kalenderapp

Benutzbarkeitstests

- Testen des eigenen Designs durch außenstehende Probanden
  - Proband führt Aktionen auf dem Papierprototyp aus.
  - Entwickler führt die Funktionen des Computers aus.
  - **Wichtig**: Alle grafischen Elemente der Benutzeroberfläche müssen getrennt auf Papier vorliegen.



Papierprototyp einer Auswahloberfläche

Eure Aufgaben …

1. Studiert eure **Aufgabenstellung**
   - Bei Unklarheiten sofort nachfragen!

2. Erstellt passende **User Stories** zu **allen Anforderungen** eurer Aufgabe.

3. Stellt sicher, dass **jede** eurer User Stories durch mindestens ein Hauptszenario als bMSC abgedeckt ist
   - Für jede User Story sollte es ein **Hauptszenario** geben
   - Wo es sinnvoll ist, sollten entsprechende **Alternativszenarien und Ausnahmeszenarien** definiert werden
   - Dokumentiert dabei jeweils nur die Interaktion zwischen dem Nutzer und dem System (**keine** systeminternen Interaktionen dokumentieren!)

4. Erstellt **Papierprotypen** für alle sichtbaren Oberflächen

Worauf ihr achten solltet

- Bei den **User Stories**:
  - **Vollständigkeit** (alle Anforderungen abgedeckt)
  - Eigenschaften **guter User Stories** (vgl. Folie 11)
- Bei den **Szenarien**:
  - **Vollständigkeit** (je ein Haupt-, Alternativ- und Ausnahmeszenario pro User Story)
  - Syntaktische **Korrektheit** der bMSCs
  - **Lesbarkeit** der bMSCs
- Bei den **Papierprototypen**:
  - **Alle** sichtbaren Oberflächen sind dokumentiert
  - **Übergänge** zwischen den einzelnen Oberflächen sind beschrieben

Quellen

- [Cohn 2004] M. Cohn: User stories applied: for agile software development. Addison Wesley 2004.
- [Wake 2003] B. Wake: INVEST in Good Stories, and SMART Tasks. URL: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/, Exploring Extreme Programming, XP123, 2003.
- [Lucassen et al. 2016] G. Lucassen, F. Dalpiaz, J.M.E.M. van der Werf, S. Brinkkemper: The Use and Effectiveness of User Stories in Practice. In: M. Daneva and O. Pastor (Eds.): REFSQ 2016, LNCS 9619, pp. 205–222, 2016.
- [Pohl 2010] K. Pohl: Requirements Engineering – Fundamentals, Principles, and Techniques. Springer, 2010.
- [Rupp 2009] C. Rupp, die Sophisten: Requirements Engineering und Management – Professionelle, iterative Anforderungsanalyse für die Praxis. 4. Auflage, Hanser, 2009.
- [Pohl und Rupp 2016] K. Pohl, C. Rupp: Basiswissen Requirements Engineering - Aus- und Weiterbildung zum »Certified Professional for Requirements Engineering«. 4., überarbeitete Auflage. dpunkt, 2016.
- [Natt och Dag und Gervasi 2005] Johan Natt och Dag, Vincenzo Gervasi: Managing Large Repositories of Natural Language Requirements. Kapitel 10 in: Aybüke Aurum, Claes Wohlin (Hrsg.), Engineering and Managing Software Requirements. Springer, 2005.
- [ITU 2011] International Telecommunication Union: Recommendation Z.120 Message Sequence Chart (MSC). ITU, 2011.
- [CHAOS 1995] The Standish Group: CHAOS. Standish Group, 1995.
- [Boehm und Basili 2001] B. B. Boehm, V. Basili: Software Defect Reduction Top 10 List. In: IEEE Computer, Vol 34, No. 1, S. 135-137. IEEE, 2001.

Quellen

- [Rolland et al. 1998] Rolland, C., Ben Achour, C., Cauvet, C. et al. A proposal for a scenario classification framework. Requirements Eng 3, 23–47 (1998).
- [SPEDiT-Lehrmaterial] Marian Daun: Ziel- und Szenariobasiertes Requirements Engineering. Foliensätze LZL2_TG03-01 - LZL2_TG03-05. SPEDiT-Konsortium, 2017.

Verwendete Grafiken

- Grafiken von https://thenounproject.com/
  - Programmer by Oksana Latysheva
  - Man Money by Gan Khoon Lay
  - User by Aleksandr Vector
  - Manager by Presenttas
  - Cloud by Roselin Christina.S
  - Service Package by Creative Stall
  - Question by unlimicon
  - Explosion by Nolan Paparelli
  - Binoculars by Postcat
  - Book by Gan Khoon Lay

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Vielen Dank für Eure Aufmerksamkeit

PALUNO
The Ruhr Institute for Software Technology