

Software Engineering

Teil 3: Überblick Requirements Engineering und
Objektorientierte Anforderungsanalyse

April 2018

Dr. Christian Bartelt



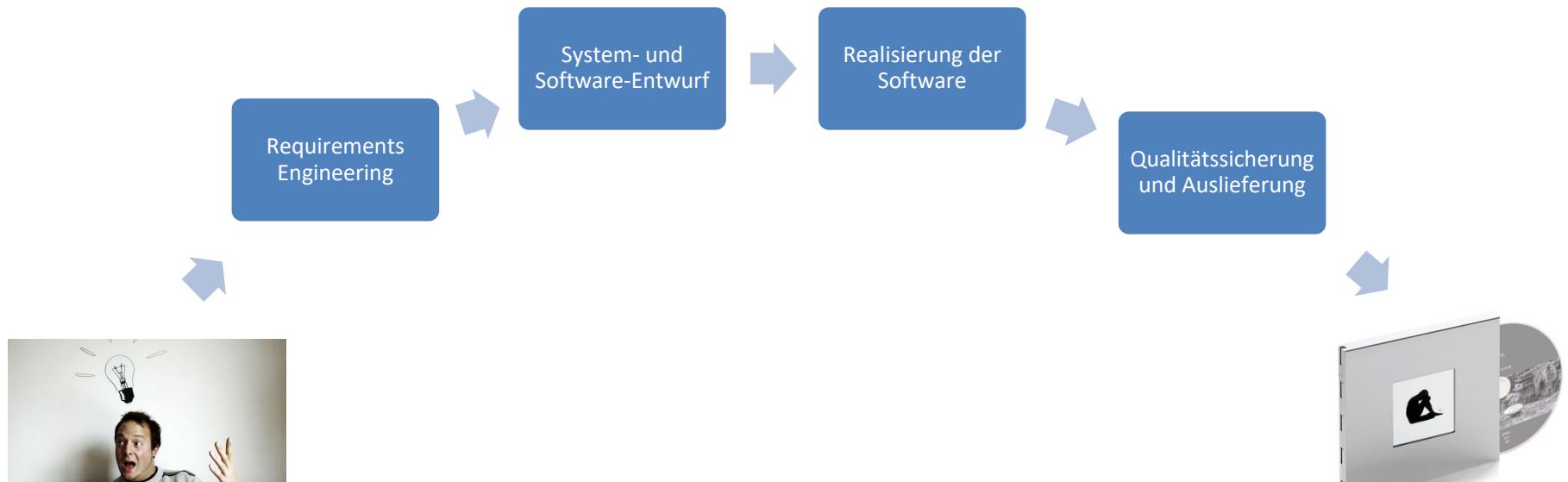
UNIVERSITATEA
BABEŞ-BOLYAI

Universität Mannheim
Institut für Enterprise Systems InES
Schloss
68131 Mannheim / Germany

UNIVERSITY OF
MANNHEIM

Zur Orientierung

Aufgabenbereiche in der Softwareentwicklung



Von der Idee...

...zum richtigen Softwareprodukt!

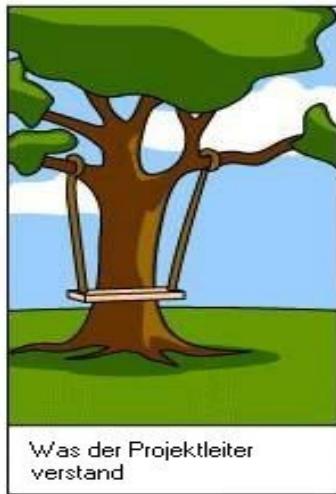
Inhalt

- Motivation und Herausforderungen
- Überblick und Konzepte
- Anforderungsanalyse in der OOA
- Anforderungsspezifikation in der OOA

Motivation



Was der Kunde erklärte



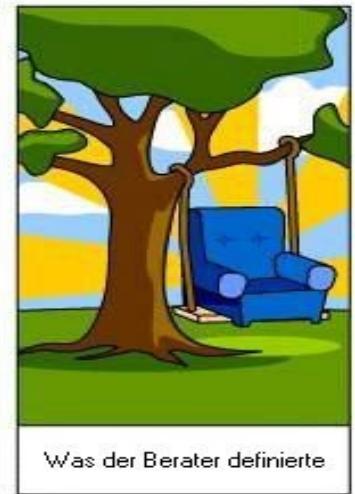
Was der Projektleiter verstand



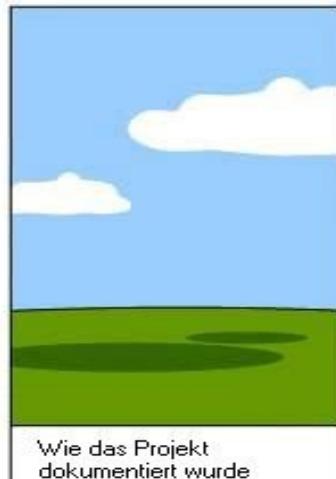
Wie es der Analytiker entwarf



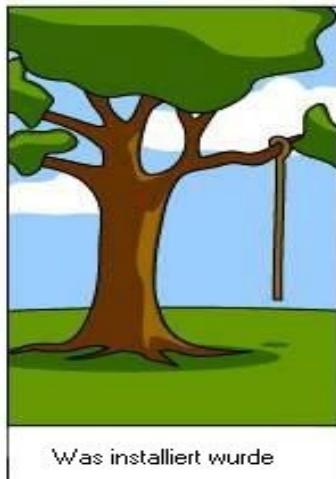
Was der Programmierer programmierte



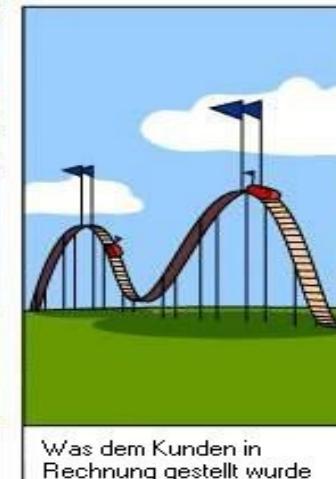
Was der Berater definierte



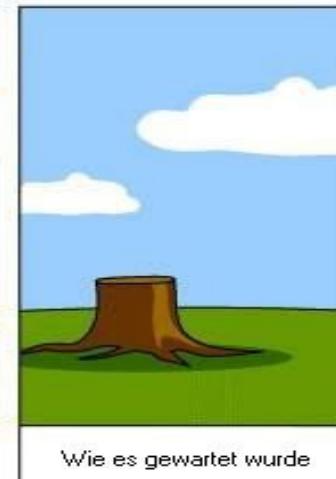
Wie das Projekt dokumentiert wurde



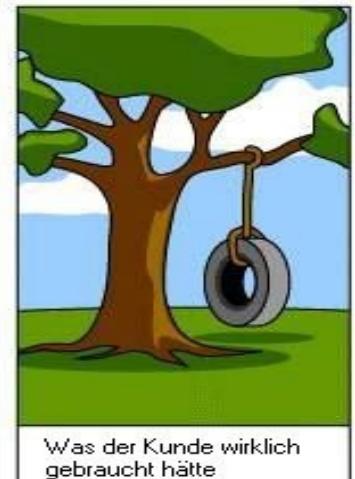
Was installiert wurde



Was dem Kunden in Rechnung gestellt wurde



Wie es gewartet wurde

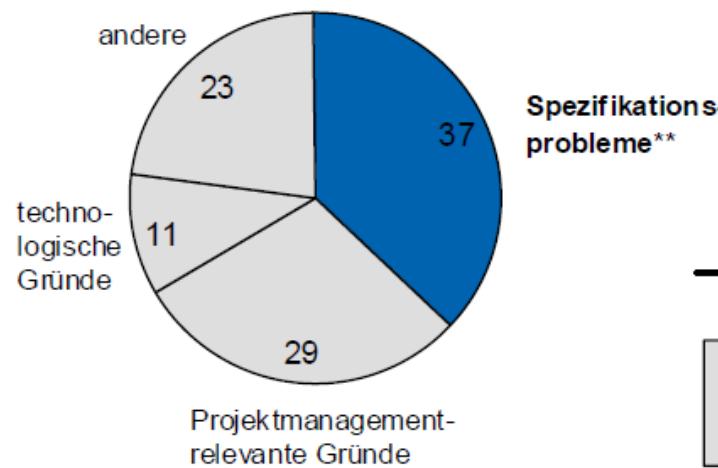


Was der Kunde wirklich gebraucht hätte

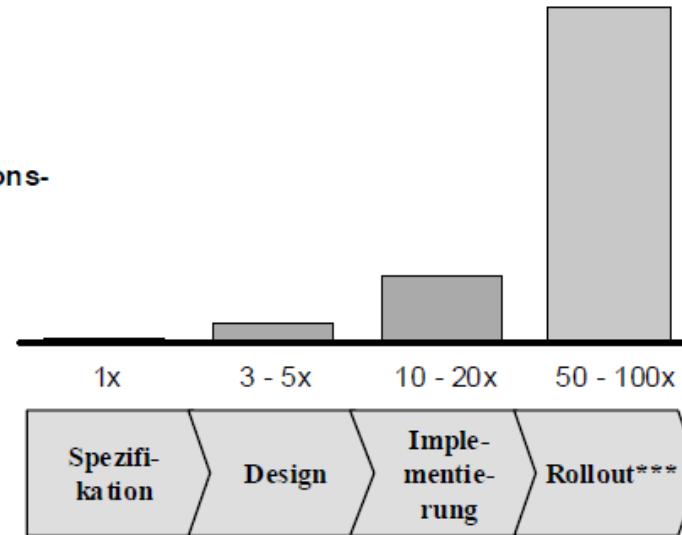
Motivation

Ungenügende Spezifikation ist Ursache für 1/3 aller gescheiterten Projekte

Gründe für gescheiterte* Projekte
in Prozent



Kosten für Behebung von Spezifikationsfehlern nach Entdeckung



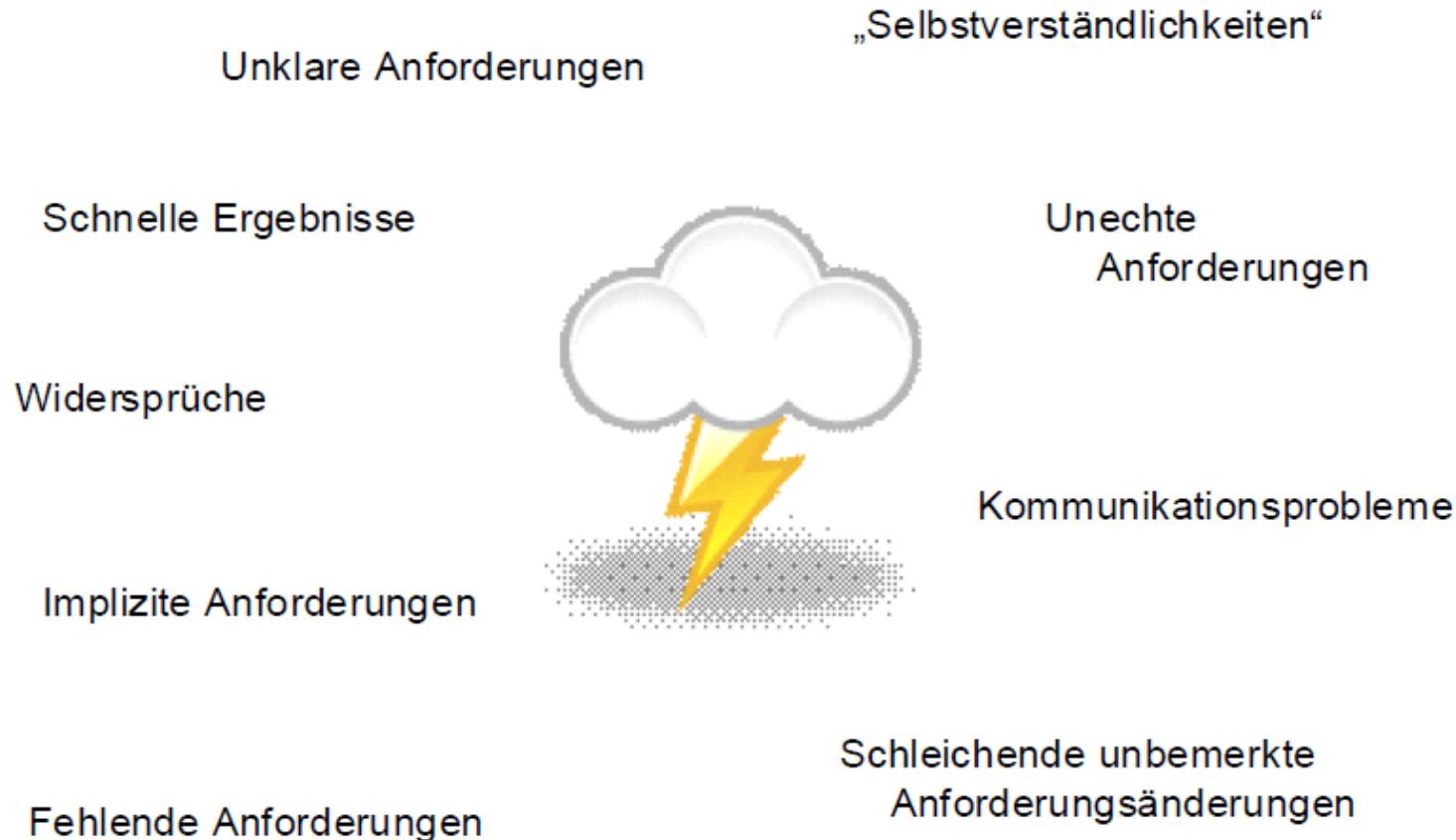
* Verspätete Projekte, die das Budget überschreiten und die Erwartungen nicht erfüllen

** Typische Probleme: fehlende Einbeziehung der Benutzer, unvollständige Anforderungen und Spezifikationen, veränderliche Anforderungen und Spezifikationen

*** Einschließlich Kosten für Rückrufe, Nachbesserung und Ersatz

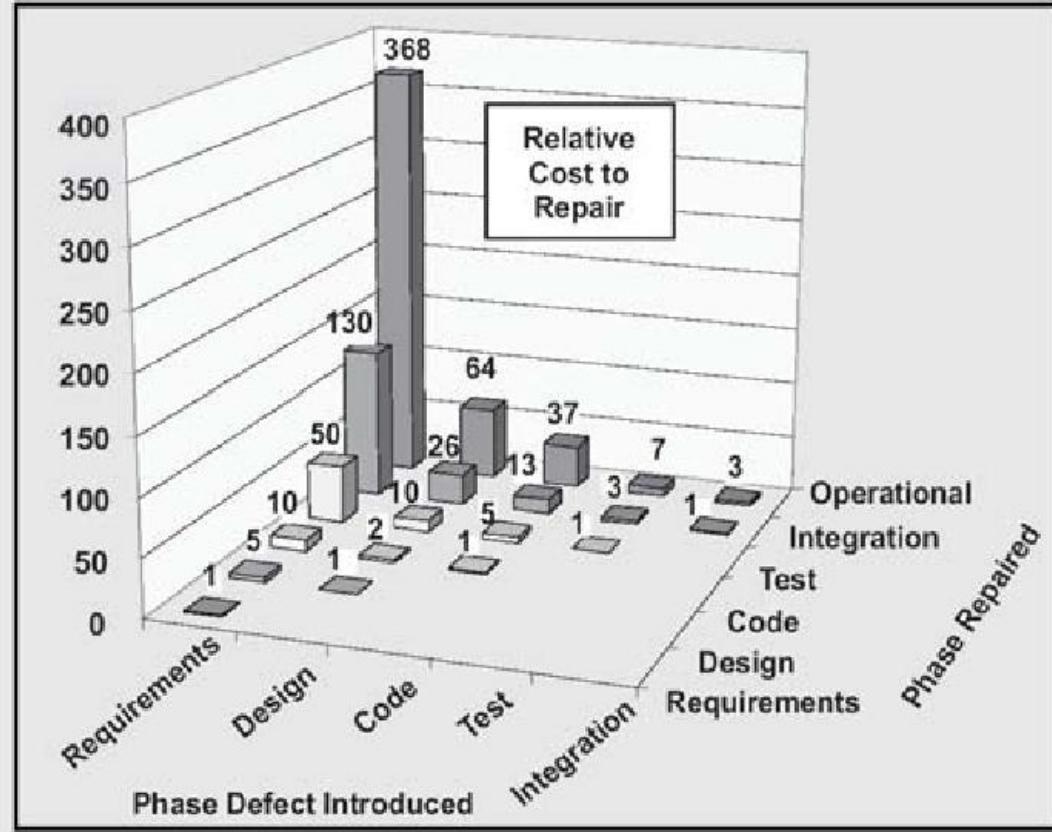
Quelle: Standish Group, Leffingwell, Widrig, McKinsey

Spezifikationsproblem?



Bedeutung von Requirements Engineering und Anforderungsspezifikation

Figure 4: *Relative Cost of Software Fault Propagation*



Je später Fehler gefunden werden, um so teurer ist die Behebung!

WHISCY Problem:
Why in the hell isn't Sam coding yet?

Herausforderungen des Requirements Engineering (1)

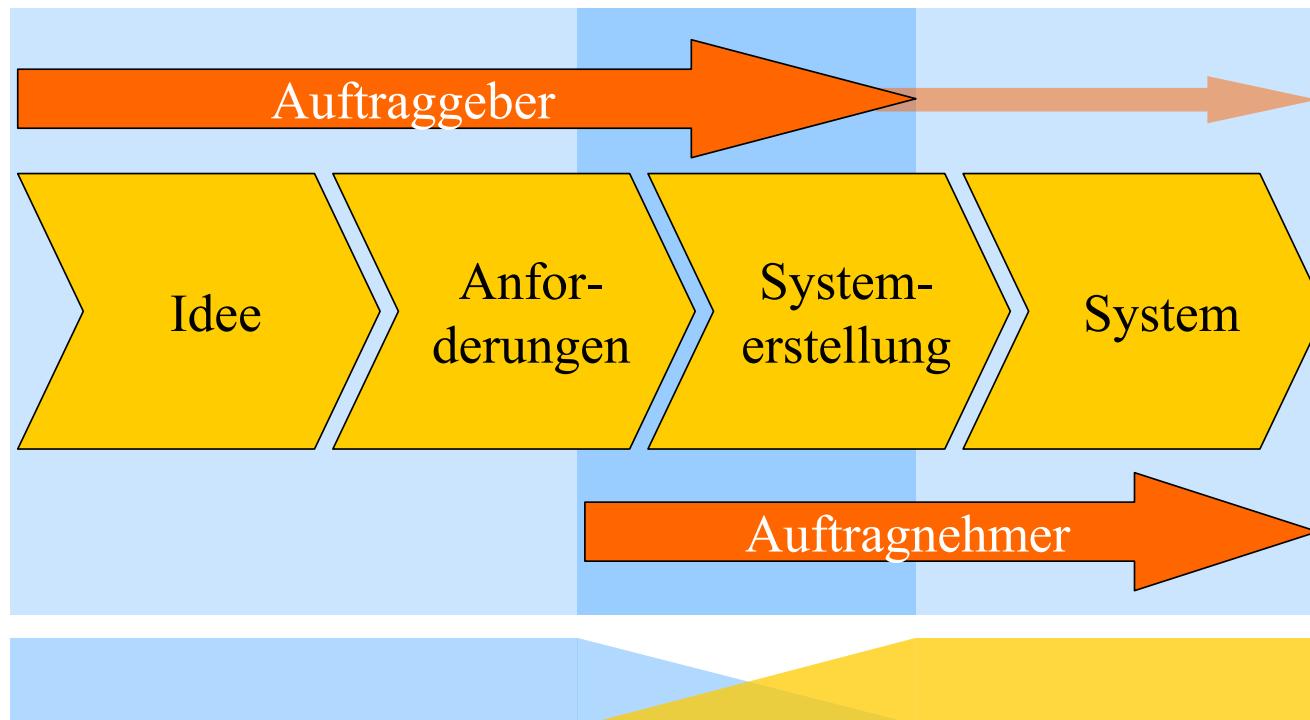
- Anforderungen so „ergattern“ und festhalten, dass sie
 - verständlich sind (Anwender/Entwickler)
 - eindeutig sind (konsistent, vollständig)
 - minimal sind
 - überprüfbar und realisierbar sind



Was der Anwender wollte	Wie es der Anwender dem Programmierer sagte	Wie es der Programmierer verstanden hat
Was der Programmierer bauen wollte	Was der Programmierer tatsächlich gebaut hat	Was der Anwender tatsächlich gebraucht hätte

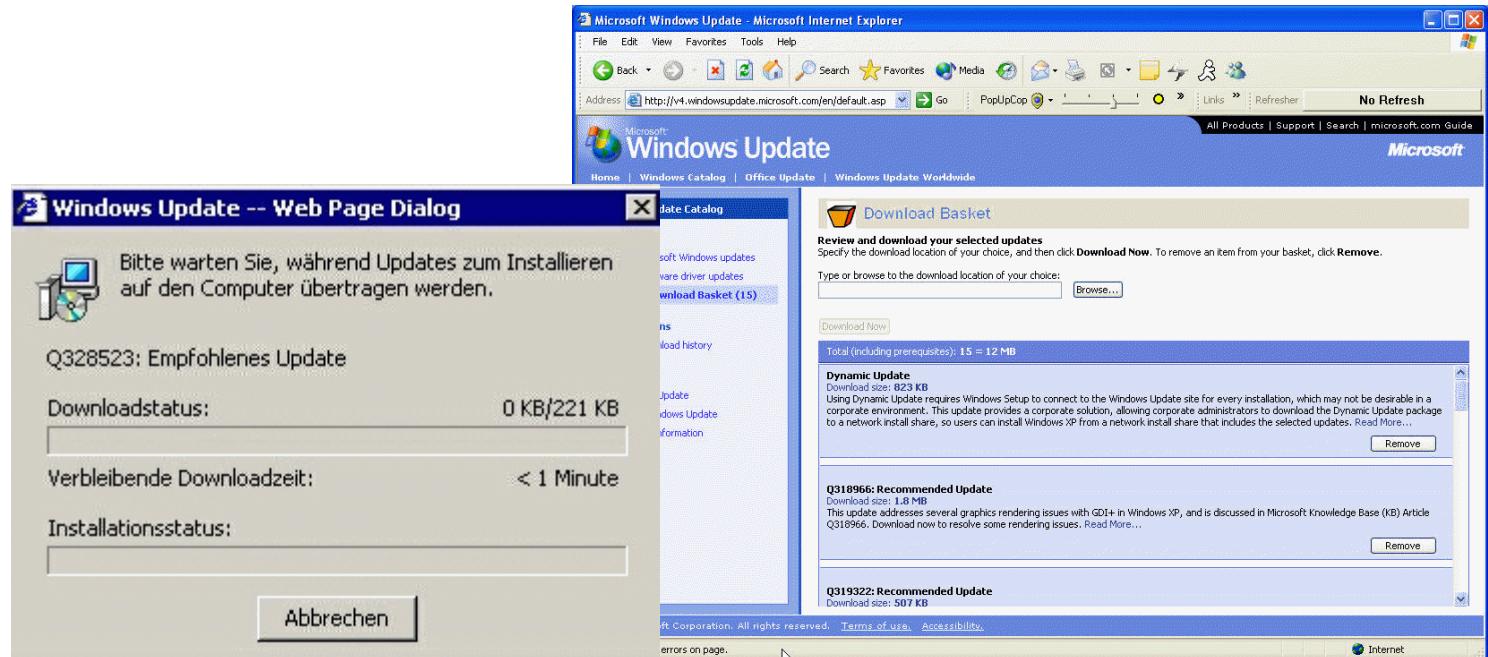
Herausforderungen des Requirements Engineering (2)

- Anforderungen transportieren und überprüfen
 - Medienbruch: AG liefert LH, AN macht dann alles neu im PH
 - Nach einigen Monaten/Jahren kann man dann erste Ergebnisse überprüfen



Herausforderungen des Requirements Engineering (3)

- Anforderungen ändern sich während der Laufzeit eines Projektes
 - Erfassung, Bewertung, Planung und Steuerung der Anforderungsänderungen?
 - Anforderungsänderungen bei einem Werkvertrag?



Inhalt

- Motivation und Herausforderungen
- Überblick und Konzepte
- Anforderungsanalyse in der OOA
- Anforderungsspezifikation in der OOA

Definition: Requirements Engineering

1. ein systematischer Ansatz um Anforderungen an ein System zu **ermitteln**, zu **organisieren** und zu **dokumentieren** und
2. ein Prozess, welcher eine Übereinkunft zwischen dem Auftraggeber und dem Projektteam über **Änderungen von Anforderungen** an das System erzielt und aufrecht erhält

Requirements Engineering ist umfassend und beinhaltet

- **Anforderungsanalyse** und
- **Requirements Management** (u.a. auch Verfolgung, Change Management)

Quelle: Dean Leffingwell: „Managing software requirements“, Addison-Wesley, 1999

Definition: Anforderungsanalyse

Nach IEEE610.12-1990 ist die Anforderungsanalyse

1. das Untersuchen der **Anwenderbedürfnisse**, um zu einer **Definition** von System-, Hardware- oder Software-Anforderungen zu gelangen
2. das Untersuchen und **Verfeinern** von System-, Hardware oder Software-Anforderungen

Es umfasst damit:

- Identifizieren
- Durchdringen & Verstehen sowie
- Modellieren (oder anderweitiges Dokumentieren)

von Funktionalen, Daten- und Nicht-Funktionalen Anforderungen

Definition: Requirements Management

Mit **Requirements Management** ist gemeint

1. das Verwalten von Anforderungen sowie
2. das Steuern der Lebenszyklen von Anforderungen

Es umfasst daher auch:

- Zurückverfolgbarkeit (Traceability)
- Überwachung
- Change Management
- Risiko Management

Was sind Anforderungen?

Definition: Anforderung

Eine Anforderung ist:

Eine Eigenschaft oder Bedingung, üblicherweise vom Kunden festgelegt, um ein Problem zu lösen oder ein Ziel zu erreichen.

Eine Eigenschaft oder Bedingung, die ein System oder eine Systemkomponente erfüllen muss, um einen Vertrag, einen Standard, eine Spezifikation oder andere formal festgelegte Dokumente zu erfüllen.

Eine dokumentierte Repräsentation einer Eigenschaft oder Bedingung wie in den vorigen Punkten beschrieben

Beispiele

Das System muss dem Administrator die Möglichkeit bieten die Benutzerliste zu drucken.

Klassen

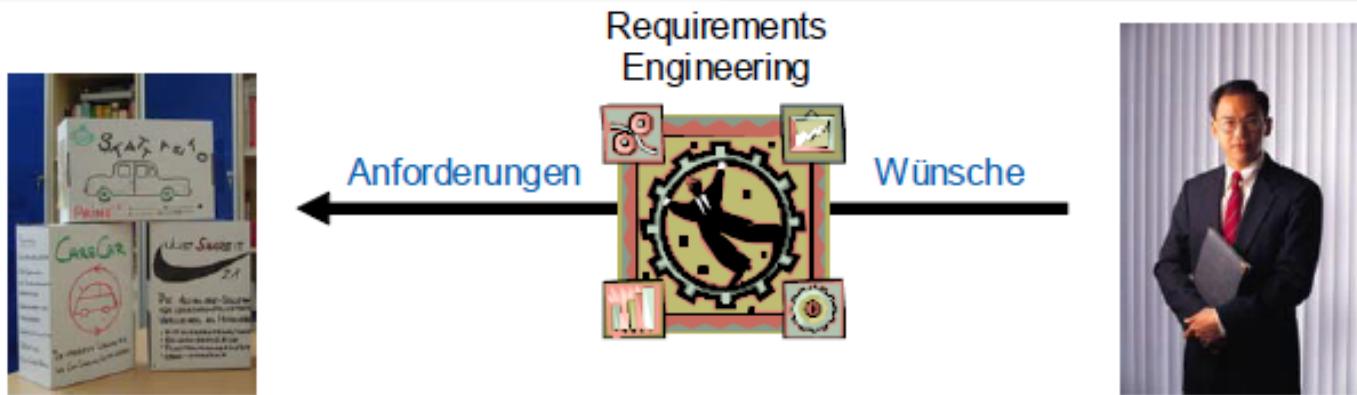
Funktionale Anf., nicht funktionale Anf.

Definition: Stakeholder

- Der Stakeholder ist eine Person oder Organisation, die Interesse am Projekt und potenziell Anforderungen hat.



Stakeholder Aufgabe „= Anforderungsträger + Anforderungsvertreter“



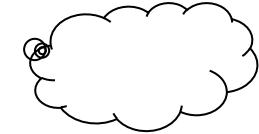
Die Anforderungen stellen Eigenschaften des zu entwickelnden Systems dar. Sie folgen aus den **Bedürfnissen** und **Wünschen** der Stakeholder.

Ein **Bedürfnis** ist das Verlangen, einem empfundenen oder tatsächlichen Defizit Abhilfe zu schaffen.

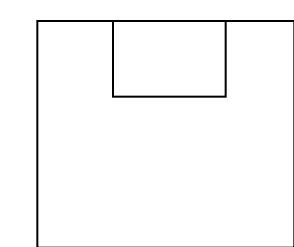
Ein **Wunsch** ist eine optionale Anforderung, die verzichtbar und nicht essenziell ist.

Wie entstehen Anforderungen?

Präzisierung der Anforderungen

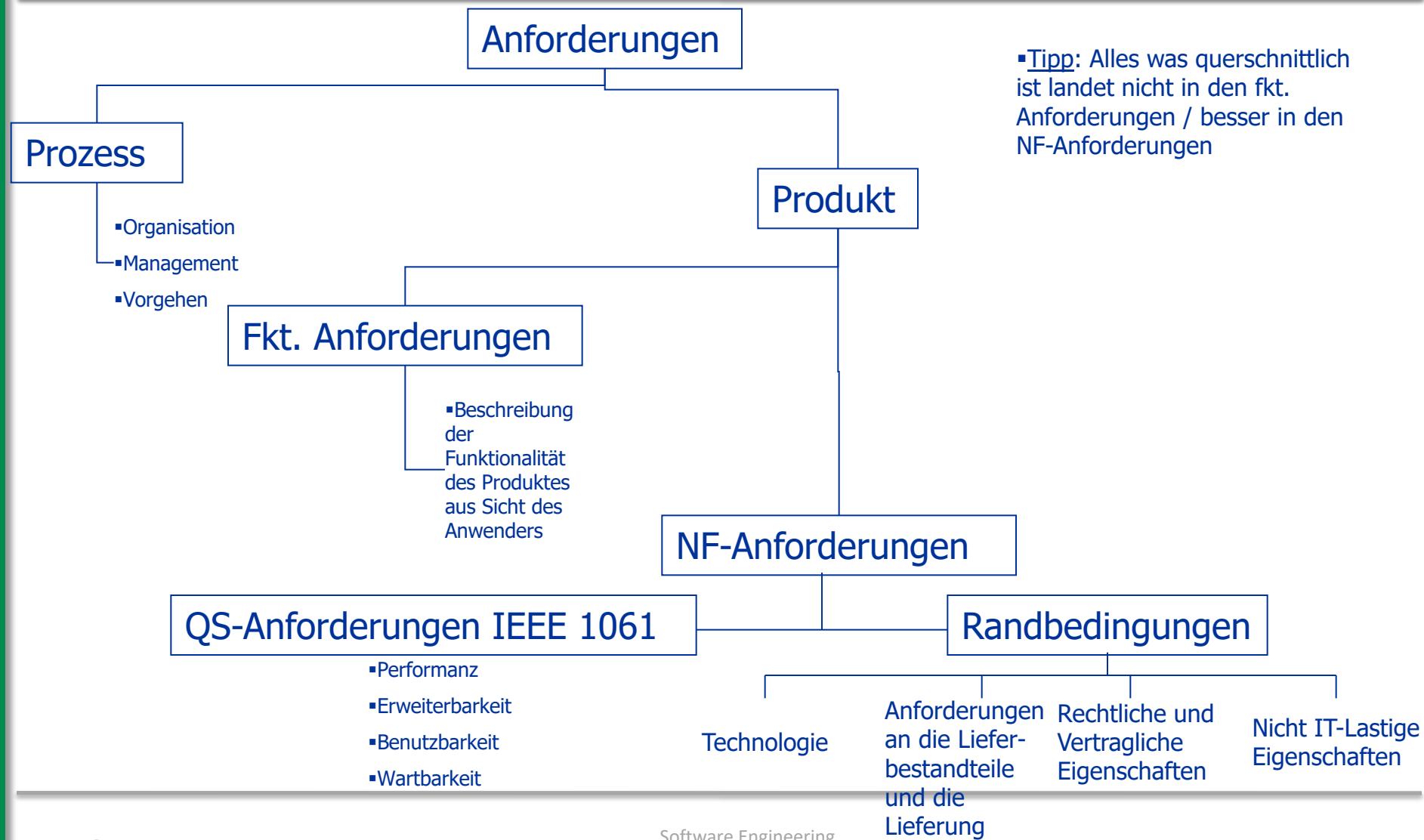


Problemraum



Lösungsraum

Kategorien von Anforderungen



Ausgangspunkt der Anforderungssammlung

Basis für Anforderungen ist eine Vielzahl von unterschiedlichen Informationsquellen von unterschiedlichen Stakeholdern

- Aber: i.d.R. verbesserungswürdige Qualität
- Mehrdeutigkeit
- Redundanzen
- Widersprüche
- Unspezifizierte Features
- Unklare/nicht abgestimmte Anforderungen
- Goldrandlösungen bei Pfennigbudget

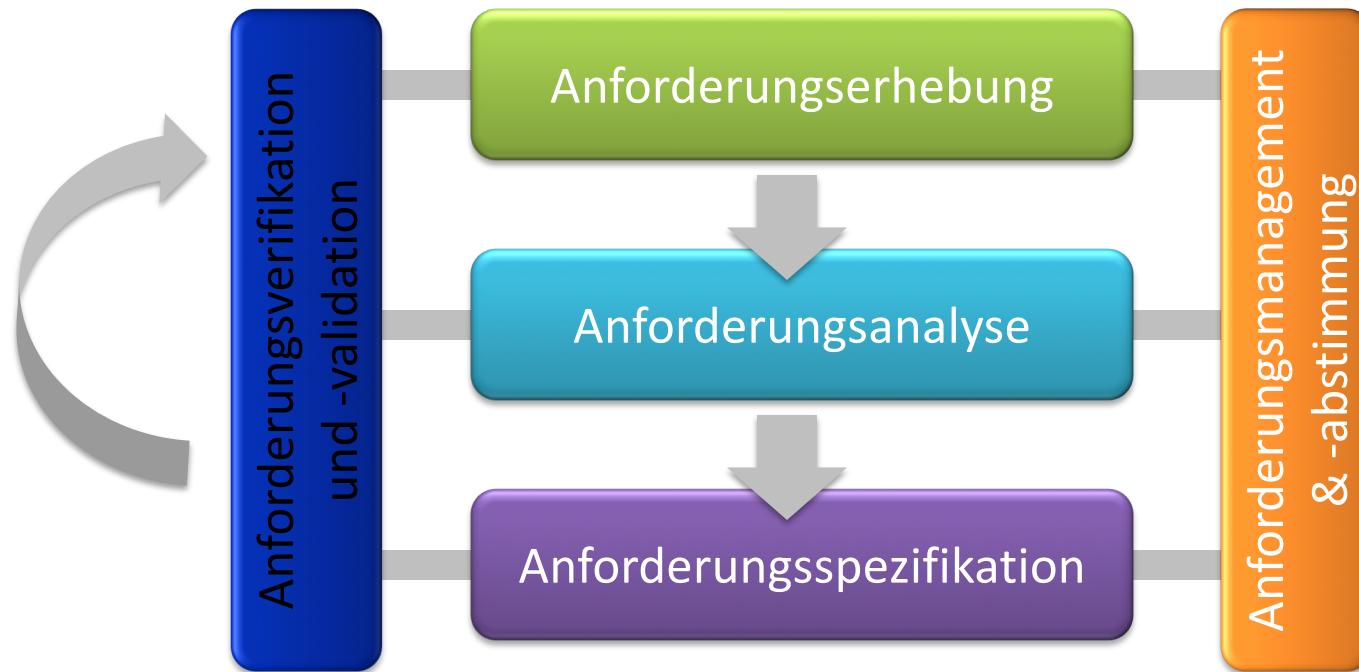
Qualitätskriterien: Wann ist eine Anforderung gut?

- Anforderungen sollten
 - ↳ Verständlich sein (Anwender/Entwickler)
 - ↳ Eindeutig sein (nicht interpretierbar/konsistent)
 - ↳ Minimal sein
 - ↳ Überprüfbar und
 - ↳ umsetzbar sein
 - Bsp.: Das System sollte einen Login-Screen haben.
 - Besser: Das System muss einen Mechanismus zur Identifizierung von Anwendern beinhalten.
 - Bsp.: Akzeptable Performanz
 - Besser: Identifizierung muss innerhalb von 5 Sekunden erfolgen.
-

Zielsetzung Requirements Engineering

- Von allen Interessenvertretern (Stakeholder) und allen Projektbeteiligten sowohl auf Auftraggeber- wie auch auf Auftragnehmer-Seite soll
 - eine gemeinsame Sprache,
 - gegenseitiges Verständnis und
 - von allen akzeptierte Zieledefiniert werden
- Verständlich, präzise, vollständig und widerspruchsfrei zu beschreiben WAS das System leisten soll
- Erste Grundstruktur des Gesamtsystems entwerfen und Realisierbarkeit überprüfen
- Anforderungen zuordnen und Verfolgbarkeit sicherstellen
- Umfang der Leistung und Lieferung definieren
- Bedingungen und Kriterien für die Abnahme festlegen

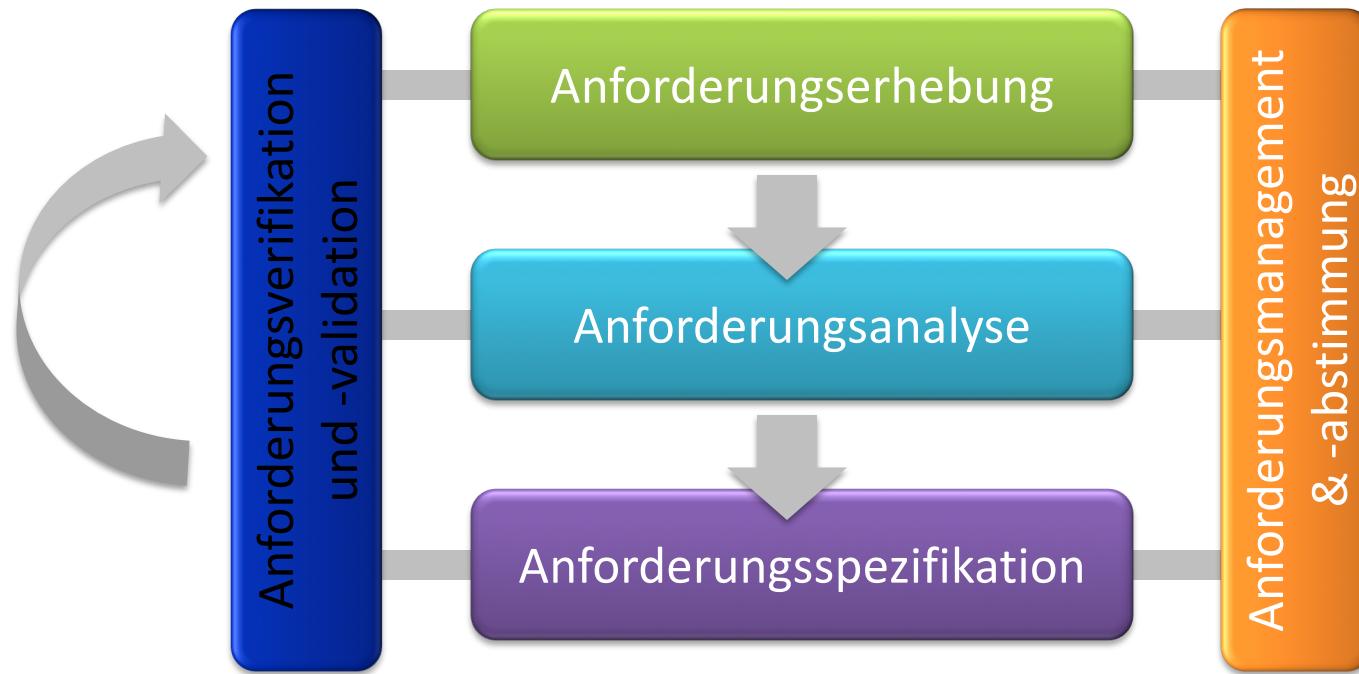
Die Aufgaben im Requirements Engineering



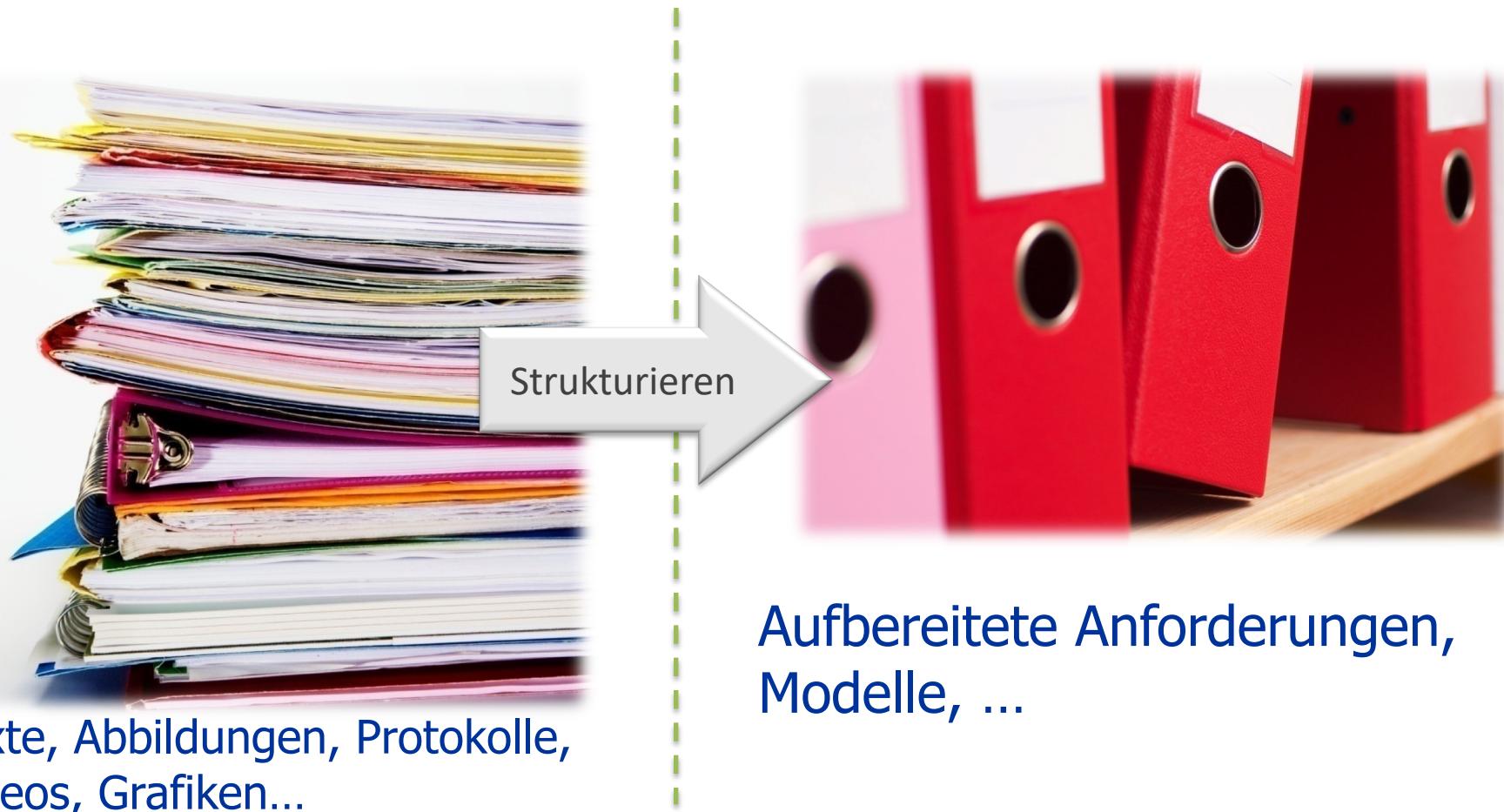
Inhalt

- Motivation und Herausforderungen
- Überblick und Konzepte
- Anforderungsanalyse in der OOA
- Anforderungsspezifikation in der OOA

Die Aufgaben im Requirements Engineering



Anforderungsanalyse: Was ist das Problem?



Anforderungsanalyse: Worum geht es?

- Konflikte zwischen Anforderungen erkennen und beheben.
 - Anforderungen eindeutig festlegen und abstimmen.
- Die Systemgrenze identifizieren und herausfinden, wie das System mit der Umgebung interagiert.
 - Systemgrenze identifizieren
 - Systemumfeld modellieren (Objektorientierte Analyse (OOA), Unified Modeling Language (UML), ...)

Wie arbeite ich die Inhalte auf?

- Strukturierte Analyse (z.B. Structured Analysis and Design Technique)
 - Denken in Datenflüssen und Funktionen
 - Grafische Darstellungen
 - Top-Down-Verfeinerung
- Objektorientierte Analyse (OOA) als objektorientierte, strukturierte Analyse
 - Teilaufgabe Domänenanalyse
- Szenarienbasierte Analyse
 - Kann Teil oder Vorstufe zur OOA sein
- Aufgabenorientierte Analyse (task oriented...)
- Zielorientierte Analyse (goal oriented ...)
- ...

Informationen / Input aus der Anforderungserhebung sichten ...

- Von wem kommt der Input
 - Stakeholder, existierende Anforderungen, vorhandene Spezifikationen, Geschäftsziele, Unternehmensstrategie, Qualitätsaspekte, künftiges Systemumfeld, Problemerichte, Helpdesk, Trainer und Consultants, Vorschläge / Klagen der Nutzer, vom Nutzer durchgeführte Verbesserungen, unbeabsichtigte Verwendung des Produktes, Konkurrenzprodukte, ...
- Woher kommt der Input
 - Kreativitätstechniken, Beobachtungstechniken, Befragungstechniken, Evolutionstechniken, Feedbacktechniken, Ziel-orientierte Techniken, ...
- Welcher Art ist der Input
 - Dokumente, Protokolle, Bilder, Flip-Charts, Fotos, Audio-Aufzeichnungen, Videos, Software, ...
- Qualität des Inputs
 - Mehrdeutigkeit, Redundanzen, Widersprüche, Unspezifizierte Features, Unklare/nicht abgestimmte Anforderungen, Goldrandlösungen bei Pfennigbudget, i.d.R. verbesserungswürdige Qualität

Verb-Substantiv-Methode (1)

Bestimmung von Klassen und deren Attributen sowie Methoden aus textuellen Problembeschreibungen (in unserem Fall Domänen-Entitäten)

Beispiel für textuelle Aufgabenstellung: Übungsanmeldung

Es soll ein Programm zur Verwaltung von Studenten und Übungen erstellt werden. Eine Übung besteht aus maximal 10 Übungsgruppen, zu denen der Raum und die Uhrzeit gespeichert ist. Jeder Raum hat eine Raumnummer und eine bestimmte Anzahl von Plätzen. Für jeden Studenten wird Name, Matrikelnummer und Email–Adresse erfaßt. Ein Student kann für eine der Gruppen angemeldet sein. In einer Gruppe ist die Zahl der angemeldeten Studenten nur durch die Zahl der Plätze limitiert.

Verb-Substantiv-Methode (2)

Heuristiken:

- Jedes Substantiv ist ein **Kandidat für eine Klasse im Domänenmodell**
- Ein Substantiv mit einem einfachen Wert (d.h. kein zusammengesetztes Objekt) ist ein **Kandidat für ein Attribut einer Domänenentität**

Es soll ein Programm zur Verwaltung von Studenten und Übungen erstellt werden. Eine Übung besteht aus maximal 10 Übungsgruppen, zu denen der Raum und die Uhrzeit gespeichert ist. Jeder Raum hat eine Raumnummer und eine bestimmte Anzahl von Plätzen. Für jeden Studenten wird Name, Matrikelnummer und Email–Adresse erfaßt. Ein Student kann für eine der Gruppen angemeldet sein. In einer Gruppe ist die Zahl der angemeldeten Studenten nur durch die Zahl der Plätze limitiert.

Verb-Substantiv-Methode (3)

Von Klassen-/Attributkandidaten zu Klassen und Attributen

- Streiche Substantive, die zum Beschreibungstext, aber nicht zum Problem gehören (—)
- Streiche doppelte Kandidaten (—)
- Betrachte immer die Singular-Form (z.B. Student statt Studenten)

Es soll ein Programm zur Verwaltung von Studenten und Übungen erstellt werden. Eine Übung besteht aus maximal 10 Übungsgruppen, zu denen der Raum und die Uhrzeit gespeichert ist. Jeder Raum hat eine Raumnummer und eine bestimmte Anzahl von Plätzen. Für jeden Studenten wird Name, Matrikelnummer und Email–Adresse erfaßt. Ein Student kann für eine der Gruppen angemeldet sein. In einer Gruppe ist die Zahl der angemeldeten Studenten nur durch die Zahl der Plätze limitiert.

Verb-Substantiv-Methode (4)

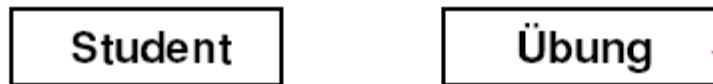
- Es soll ein Programm zur Verwaltung von Studenten und Übungen erstellt werden.

Student

Übung

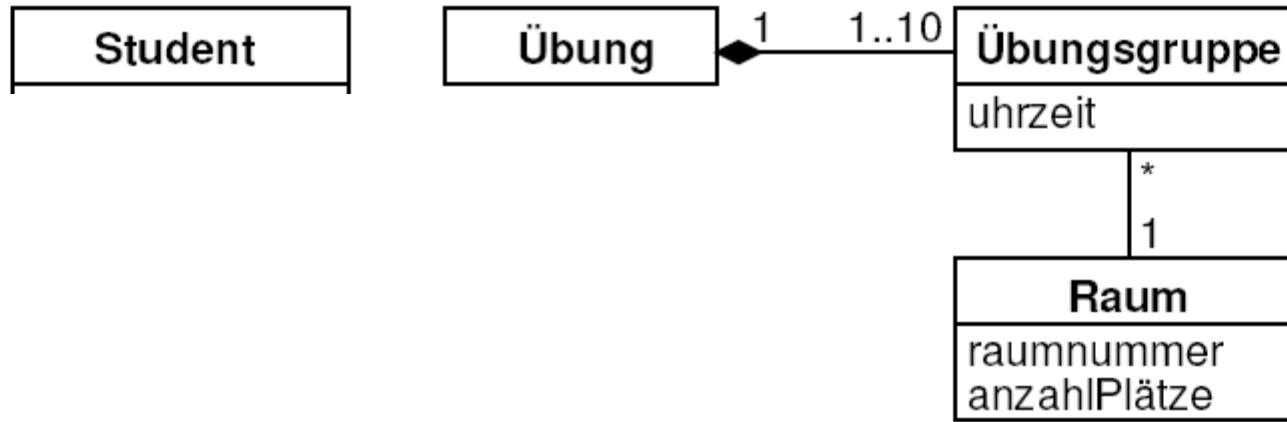
Verb-Substantiv-Methode (5)

- Eine Übung besteht aus maximal 10 Übungsgruppen , zu denen der Raum und die Uhrzeit gespeichert ist. Jeder Raum hat eine Raumnummer und eine bestimmte Anzahl von Plätzen .



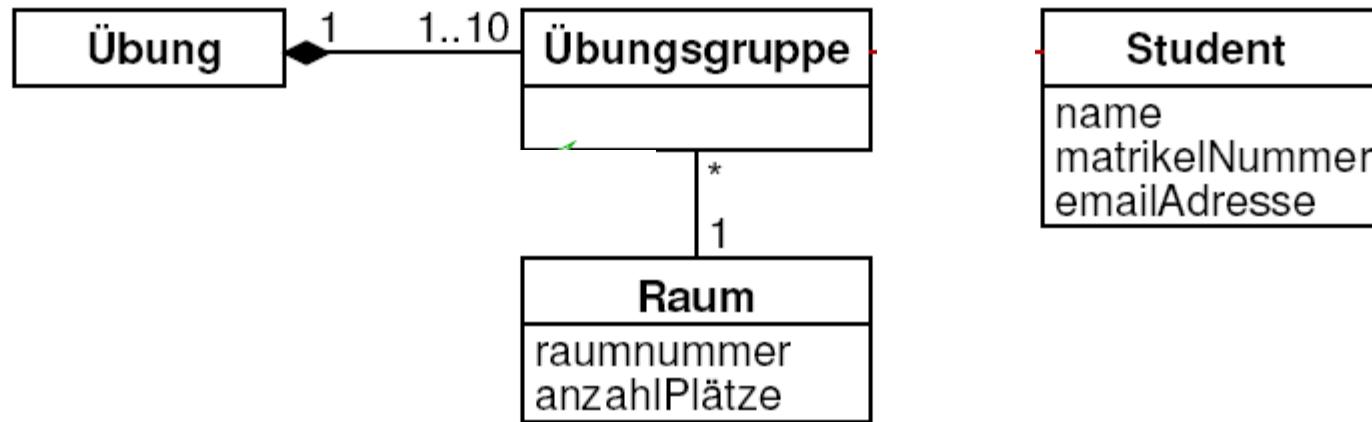
Verb-Substantiv-Methode (6)

→ Für jeden Studenten wird Name, Matrikelnummer und Email-Adresse erfaßt.



Verb-Substantiv-Methode (7)

- Ein Student kann für eine der Gruppen angemeldet sein. In einer Gruppe ist die Zahl der angemeldeten Studenten nur durch die Zahl der Plätze limitiert.



Verb-Substantiv-Methode (8)

Weitere Heuristik:

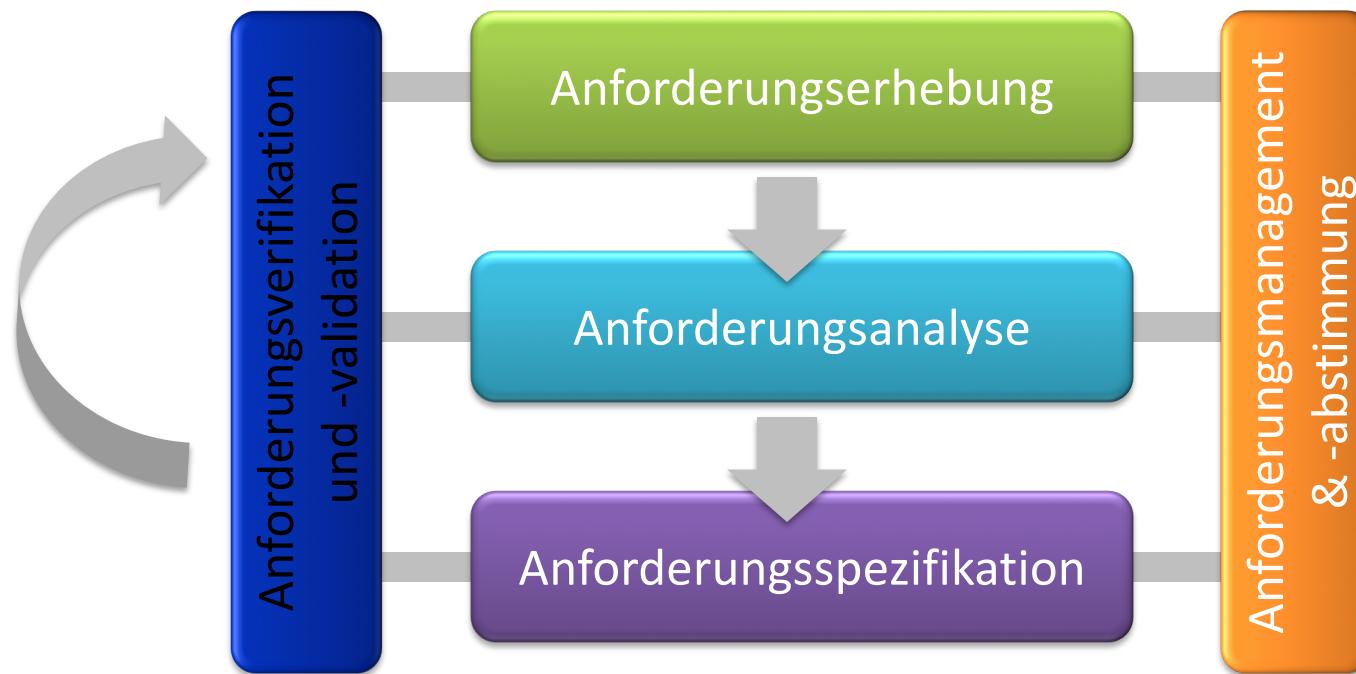
- Verben geben Hinweise auf **Use-Cases**

Es soll ein Programm zur Verwaltung von Studenten und Übungen erstellt werden. Eine Übung besteht aus maximal 10 Übungsgruppen, zu denen der Raum und die Uhrzeit gespeichert ist. Jeder Raum hat eine Raumnummer und eine bestimmte Anzahl von Plätzen. Für jeden Studenten wird Name, Matrikelnummer und Email–Adresse erfaßt. Ein Student kann für eine der Gruppen angemeldet sein. In einer Gruppe ist die Zahl der angemeldeten Studenten nur durch die Zahl der Plätze limitiert.

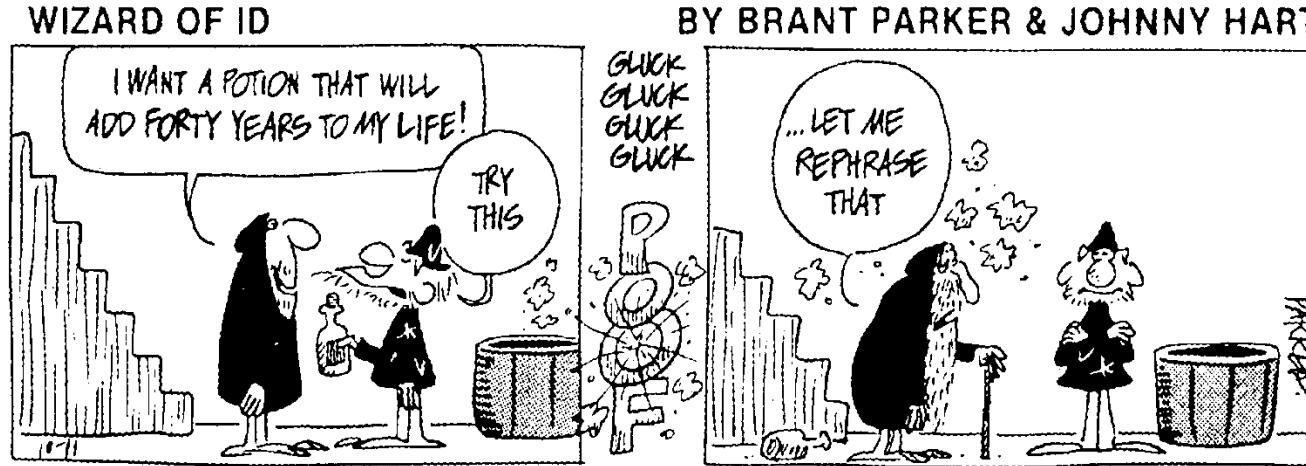
Inhalt

- Motivation und Herausforderungen
- Überblick und Konzepte
- Anforderungsanalyse in der OOA
- Anforderungsspezifikation in der OOA

Die Aufgaben im Requirements Engineering



Anforderungsspezifikation: Wo ist das Problem?

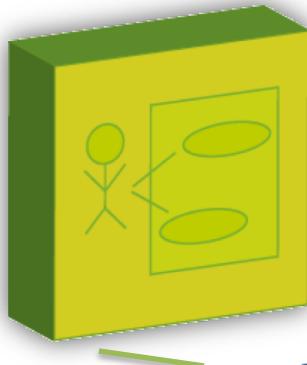


Aufgabe: syntaktisch und semantisch korrekte Anforderungen

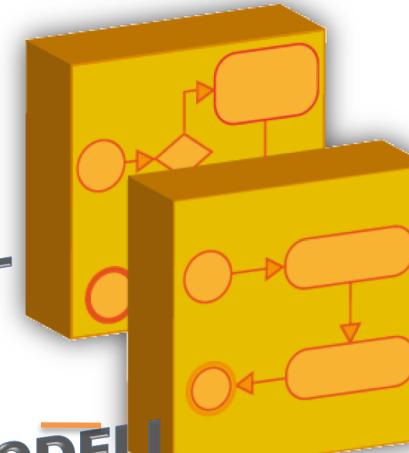
Herausforderung: Anforderungen so aufzuschreiben, dass der Leser der Spezifikation das gleiche Bild von dem zu entwickelnden System hat wie der Verfasser

Anforderungsspezifikation: Elemente der Anforderungsspezifikation

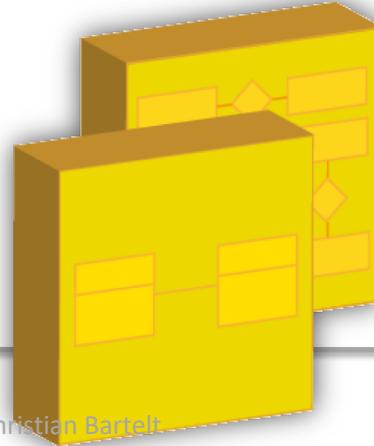
Textuelle Anforderungen und Stakeholder erfassen



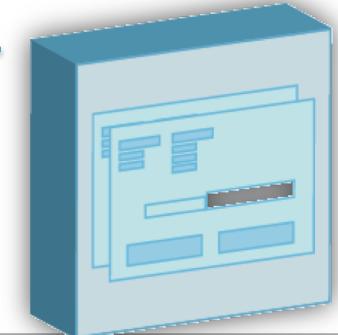
**ANWENDUNGSFALLMODELL
/ DATENMODELL**



AKTIVITÄTSMODELL



**Nicht-funktionale
Anforderungen**



Textuelle Anforderungen: Funktionale und Nicht-Funktionale (1)

- Textbasierte Anforderung bedeutet, jede Anforderung ist eine textuelle Einheit
- Jede einzelne Anforderung bekommt eine ID (Identifikator), z.B.
 - eine durchlaufende Nummer („241“)
 - eine Dezimalklassifikation („2.4.1“),
 - symbolische Namen („Navi-15“),
 - oder eine Kombination („Navi 2.4.1“)
- Die Vergabe der ID muss koordiniert werden, damit keine doppelt vorkommt.
- Mit Hilfe der symbolischen Namen kann jedes Team einen eigenen Namensraum bekommen und damit IDs unabhängig voneinander vergeben.
- Tools können die ID automatisch vergeben.
- Das Verfahren zur Identifikation muss bereits vor Beginn der Analyse feststehen.
- Die eindeutige Identifizierung von Anforderungen von der Entstehung bis nach Abschluss eines Projekts stellt einen Kernpunkt in der Verwaltung von Anforderungen dar.
 - Dies ermöglicht jederzeit den eindeutigen Bezug auf Anforderungen,
 - ist eine wichtige Grundlage für das Änderungsmanagement,
 - ermöglicht die Rückverfolgbarkeit.

Textuelle Anforderungen: Funktionale und Nicht-Funktionale (2)

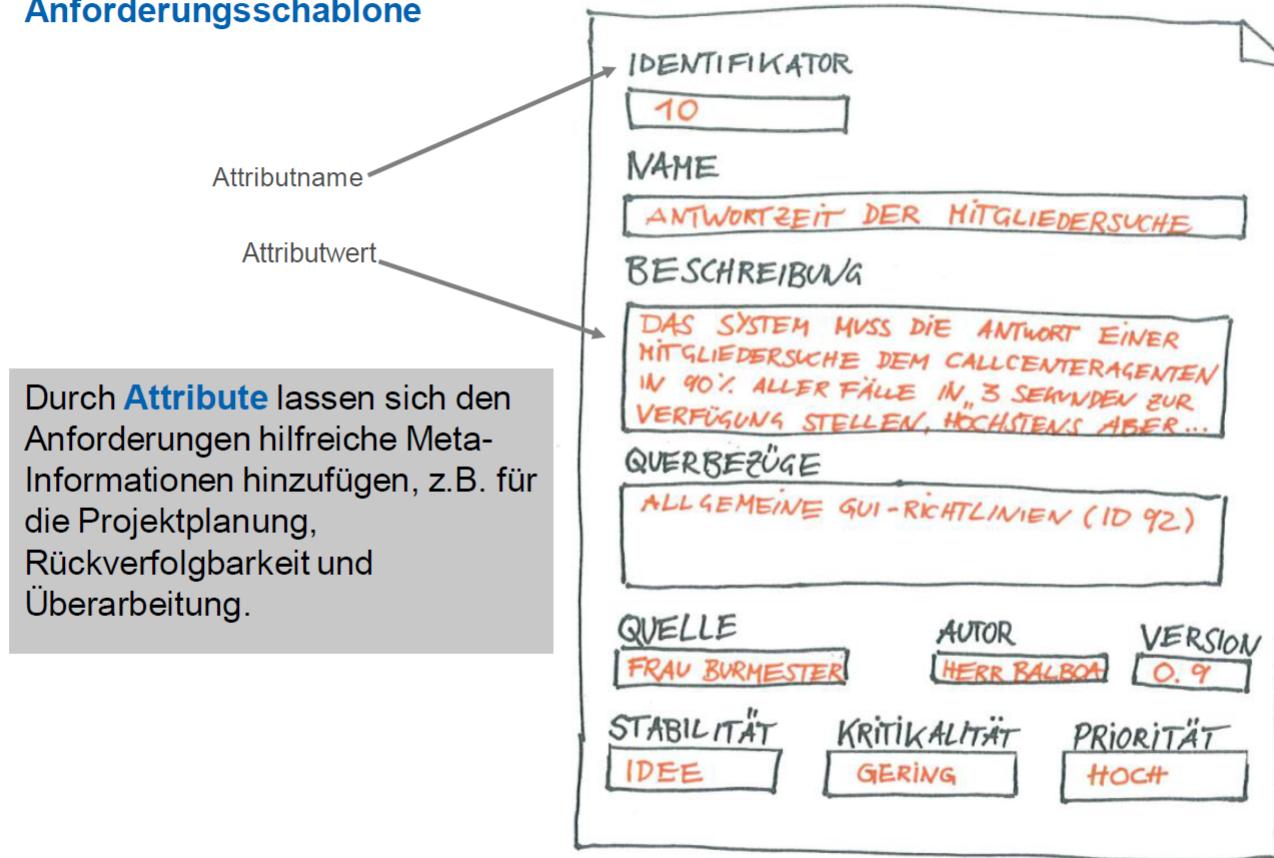
Funktionale und Nicht-Funktionale Anforderungen

- Beschreibung der Funktionalität des Systems aus der Sicht des Anwenders (WER und WAS!)
- Textuelle Anforderungen sind
 - Eindeutig nummeriert
 - Ggf. hierarchisch strukturiert
- Gegenstand von: Verwaltung, Nachverfolgung, Überprüfung u. Abnahme
- BT: Prosa, Tabellen, strukturierter Text,

FA9
FA9.1
FA9.2

Textuelle Anforderungen: Funktionale und Nicht-Funktionale (3)

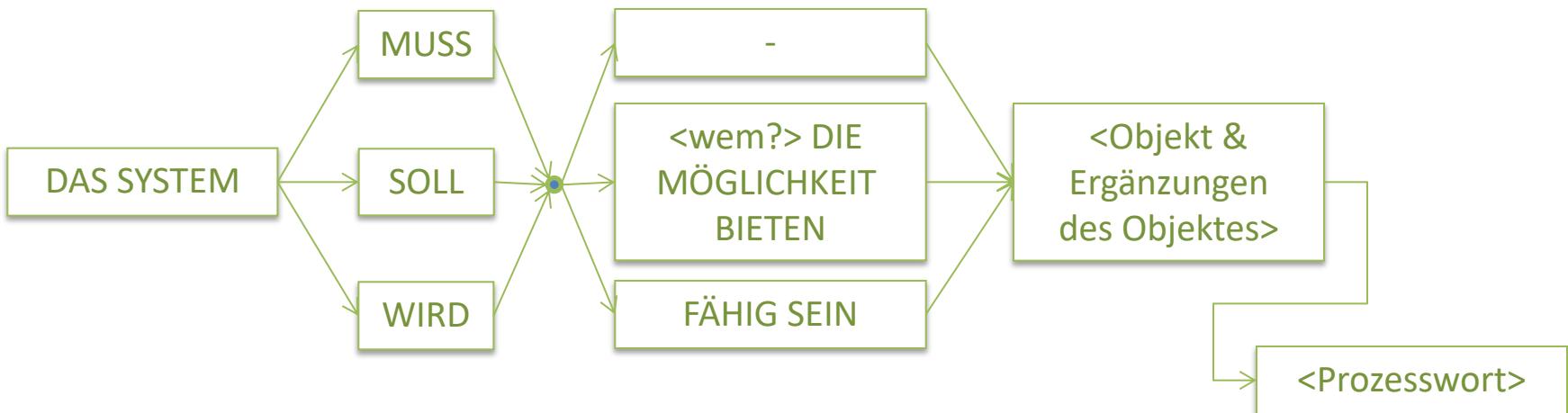
Anforderungsschablone



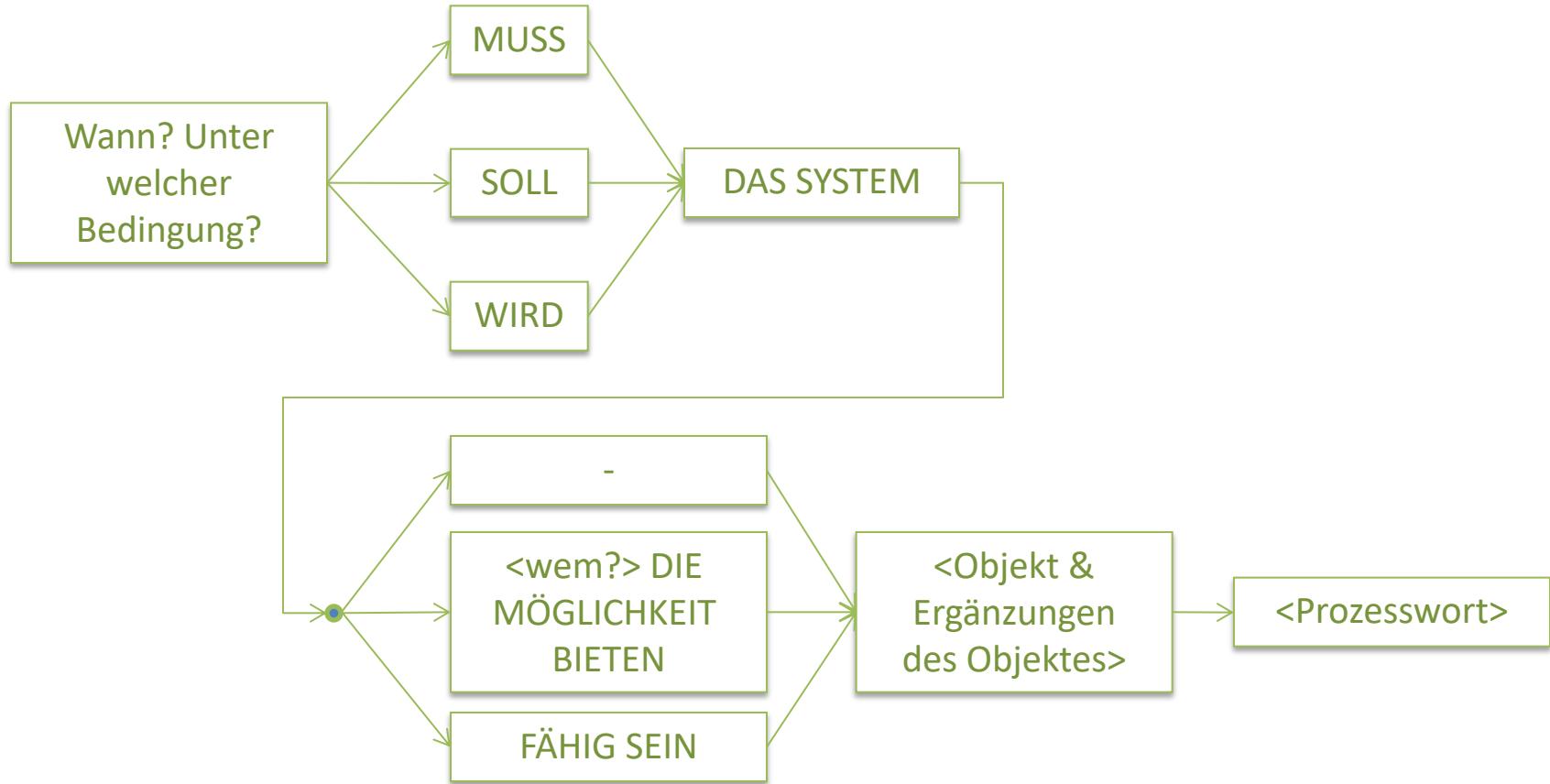
Stakeholder erfassen

Rolle der Stakeholder	Beschreibung	Vertreter	Verfügbarkeit	Wissensgebiet	Begründung
Anwender	Sind die Benutzer des Systems	Fr. Maier Tel.: 063..	Tägl. 10-12	Staff-Management	Anwender muss mit System arbeiten
Management	Nennt Projektziele + Budget

Beispiel: Textbasierte Anforderungen mit Text-Schablone (1)



Beispiel: Textbasierte Anforderungen mit Text-Schablone (2)



Beispiel: Textbasierte Anforderungen mit Text-Schablone (3)

1. Bestimmen des Prozesswortes

2. Charakterisieren der Aktivität des Systems

- Selbstständige Systemaktivität (-)
- Benutzerinteraktion (die Möglichkeit bieten)
- Schnittstellenanforderung (fähig sein)

3. Festlegen der rechtlichen Verbindlichkeit

- Zwingende Anforderung (Muss)
- Optionale Anforderung (Soll)
- Zusätzliche Anforderung (Kann)

4. Feinschliff für den Prozess

5. Formulierung von logischen und zeitlichen Bedingungen

Beispiel: Während der Pausenplanung muss das System die aktuellen Pausenaufsichtstatistiken anzeigen.

Beispiel: User Story Cards – Was sind User Story

- A concise, written description of a piece of functionality that will be valuable to a user (or owner) of the software.
- Stories are:
 - User's needs
 - Product descriptions
 - Planning items
 - Tokens for a conversation
 - Mechanisms for deferring conversation
 - ...

Aus was bestehen User Story Cards

- 1. Description** - A written description of the user story for planning purposes and as a reminder
- 2. Conversation** - A section for capturing further information about the user story and details of any conversations
- 3. Confirmation** - A section to convey what tests will be carried out to confirm the user story is complete and working as expected

Textschablone für User Story (1)

- **As a [user role] I want to [goal]
so I can [reason]**
- **As a [type of user] I want to [perform some task] so
that I can [reach some goal]**

Example:

- As a **registered user** I want to **log in**
so I can access subscriber-only content

Textschablone für User Story (1)

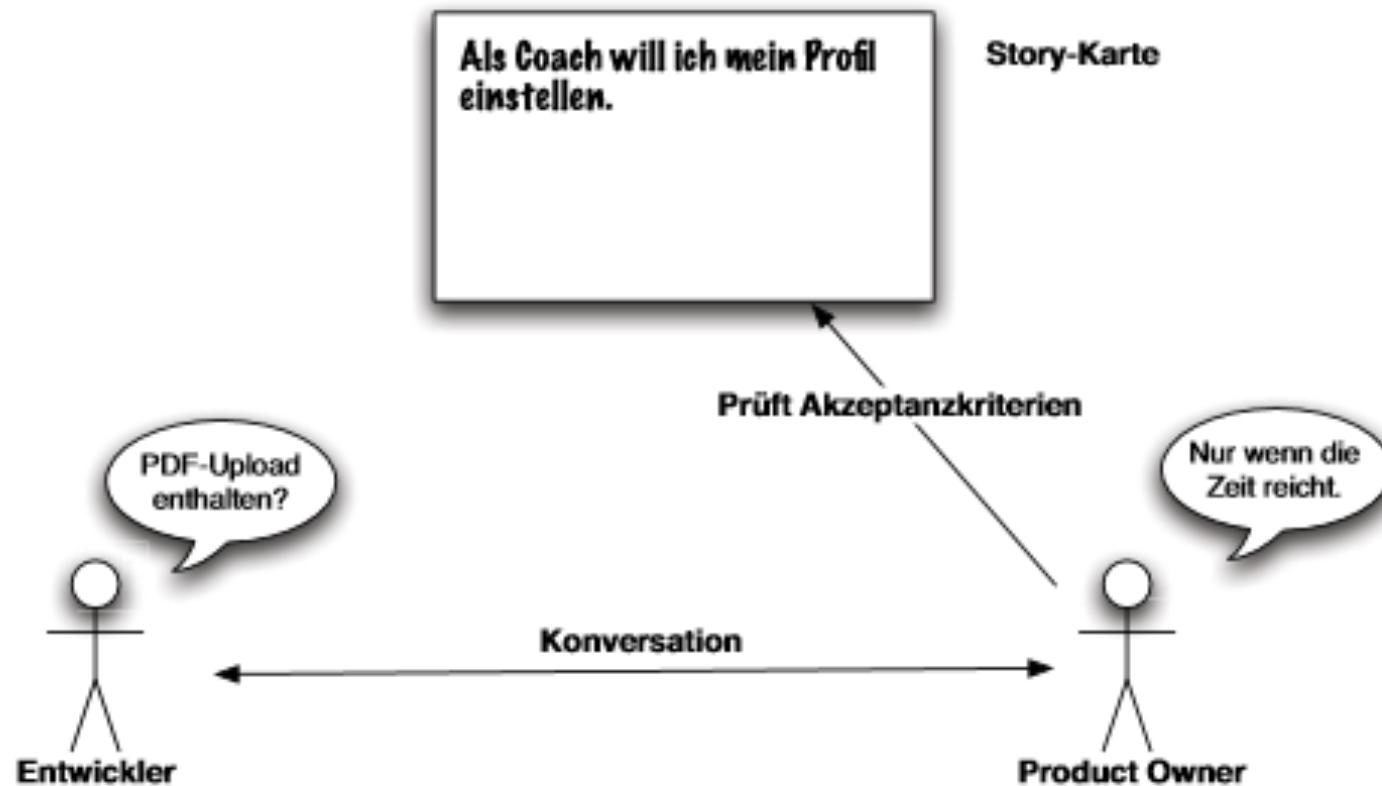
- **Who** (user role)
- **What** (goal)
- **Why** (reason)
 - gives clarity as to why a feature is useful
 - can influence how a feature should function
 - can give you ideas for other useful features that support the user's goals

Wie beschreibt man eine User Story

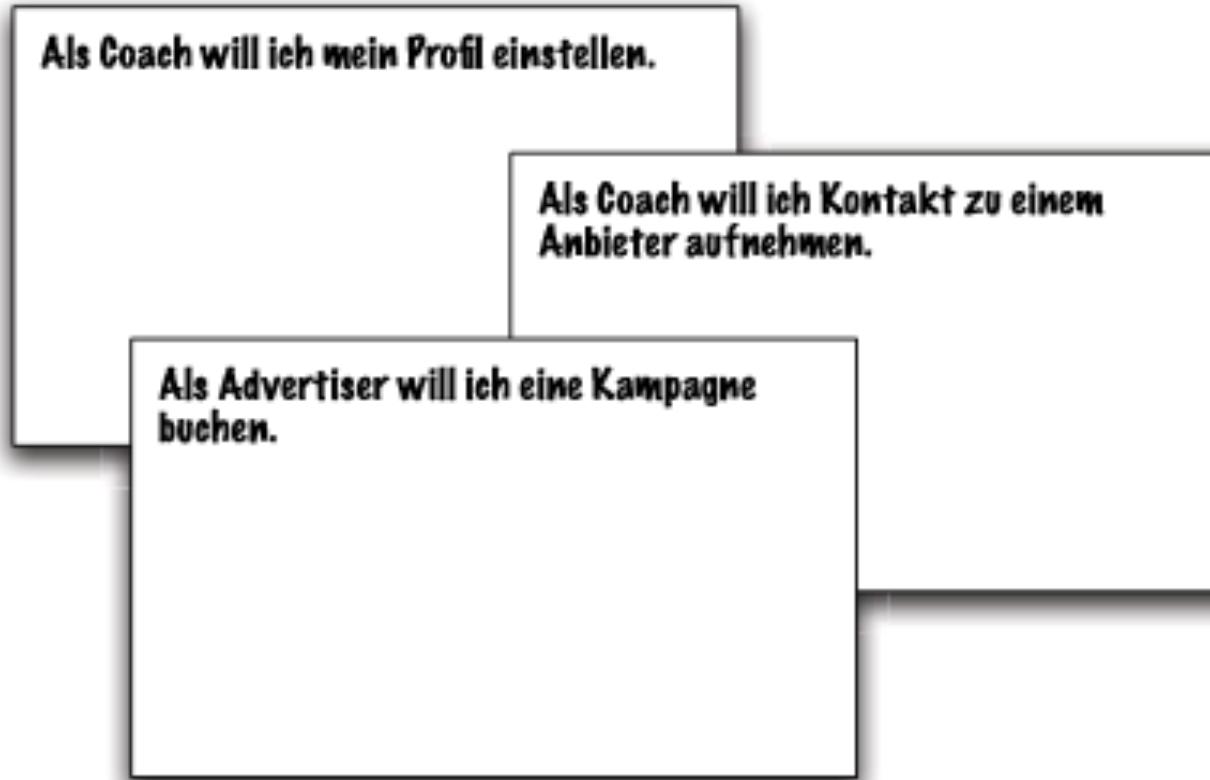
Steps:

- Start with a title.
- Add a concise description using the templates.
- Add other relevant notes, specifications, or sketches
- Before building software write acceptance criteria (how do we know when we're done?)

Beispiel: User Story Card (1)



Beispiel: User Story Card (2)

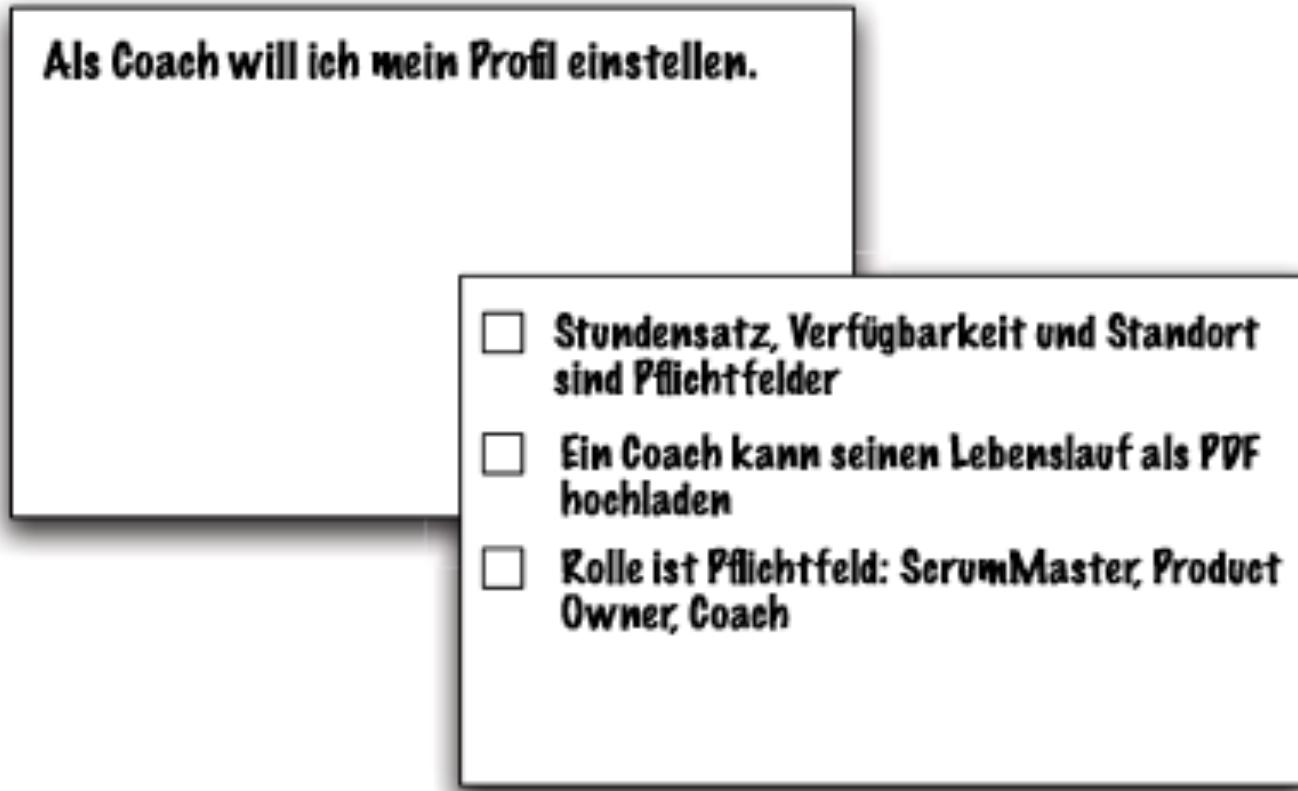


Beispiel: User Story Card (3): Conversation: Zusätzliche Informationen (Hinweise, Spezifikationen, Bilder, offene Punkte, etc.)

Als Coach will ich mein Profil einstellen.

- PDF-Upload implementieren
- Welche Felder sind Pflicht?
- Müssen Profile vom Admin freigegeben werden?

Beispiel: User Story Card (4): Confirmation: Akzeptanzkriterien (relevant für die Abnahme!)



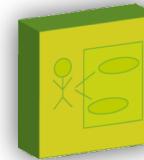
Objektorientierte Analyse mit UML

Erstellen eines Anforderungsmodells

- Verschiedene Sichten auf Anforderungen

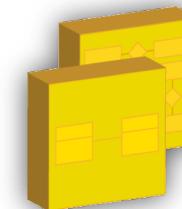
- Anwendungsfallmodell

- Außensicht
 - Funktionen, Aufgaben
 - *UseCases (Anwendungsfälle)*



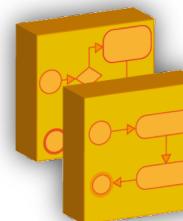
- Domänenmodell

- Domänenentitäten
 - *Klassendiagramme*



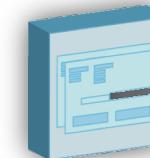
- Aktivitätsmodell

- Abläufe
 - *Aktivitäten*



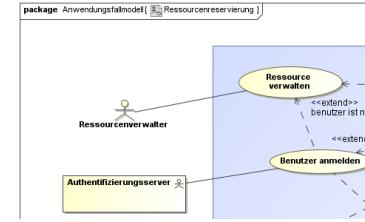
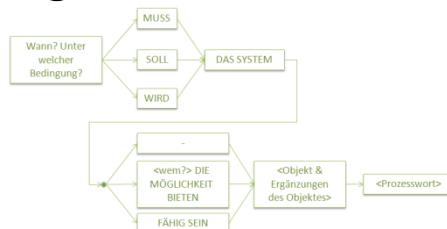
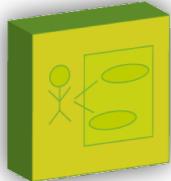
- Screen-Mockup-Modell

- UI für Prototyp, zur Visualisierung der Abläufe
 - *UI-Skizzen*



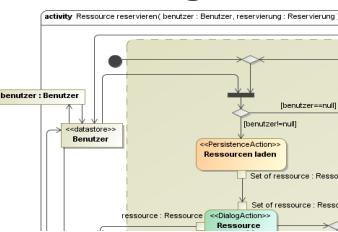
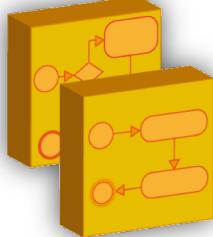
Informationsmodell (2)

Anwendungsfälle

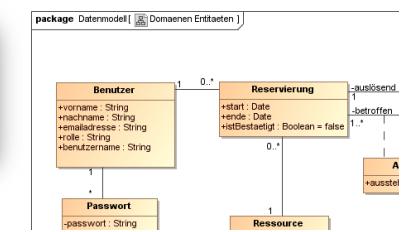
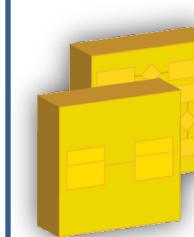


Benutzer anmelden	
Kurzbeschreibung	Das System muss dem Benutzer die Möglichkeit bieten, sich anzumelden.
Akteure	Benutzer
Kategorie	Muss
Auslöser	Der Reservierende betätigt den Button „anmelden“
Vorbedingung	Der Benutzer ist nicht angemeldet.
Eingabe	Der Benutzer hat ein gültiges Benutzerkonto (Benutzername, Passwort).
Ausgabe	Benutzername, Passwort
Nachbedingung	Der Benutzer ist erfolgreich angemeldet.
Standardablauf	<p>Das System präsentiert dem Benutzer ein Dialogfenster, in dem der Benutzer seinen Benutzernamen und sein Passwort eingeben kann.</p> <p>Der Benutzer gibt seinen Benutzernamen ein.</p> <p>Der Benutzer gibt sein Passwort ein.</p> <p>Der Benutzer bestätigt die Eingabe.</p> <p>Das System überprüft Benutzername und Passwort.</p> <p>Das System meldet den Benutzer an.</p> <p>Das System zeigt dem Benutzer die Information an, dass er angemeldet ist.</p>

Ablaufbeschreibungen

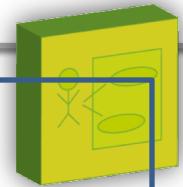


Domänenentitäten

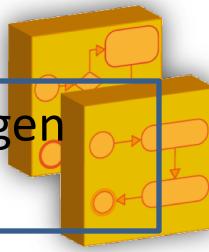


Informationsmodell (3)

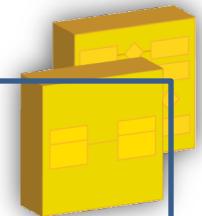
Anwendungsfälle



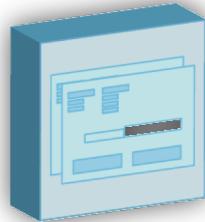
Ablaufbeschreibungen



Domänenentitäten



Interface-Mock-Ups



Benutzer angemeldet?

System überprüft:
Benutzer angemeldet?
 Der Benutzer ist nicht angemeldet:
benutzer = null
 Der Benutzer ist angemeldet:
benutzer != null

[Entscheidung simulieren](#)

Reservierungskonflikt?

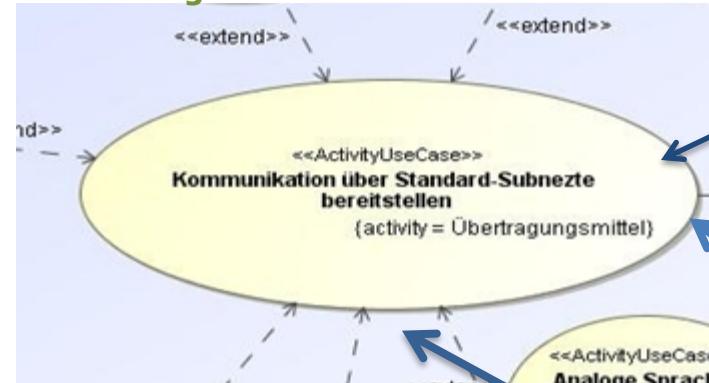
System überprüft:
Gibt es einen Reservierungskonflikt?
 Es gibt keinen Konflikt:
konflikt == null
 Es gibt einen Konflikt:
konflikt != null

[Entscheidung simulieren](#)

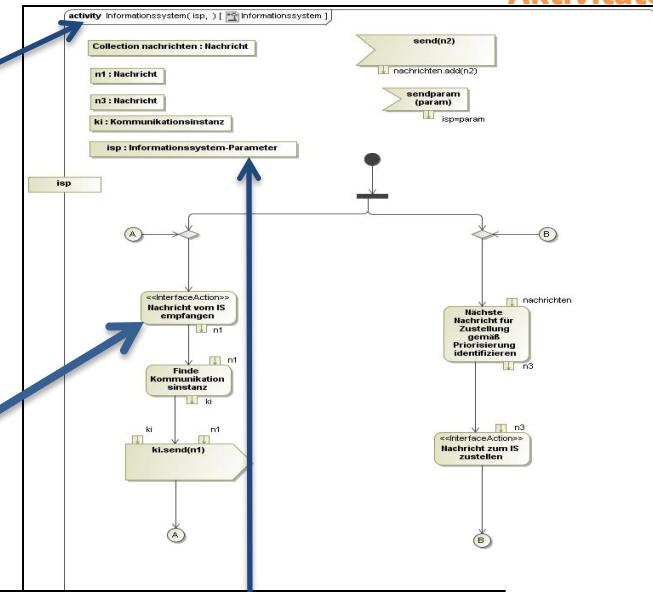


Anforderungsmodell bis zum Lastenheft

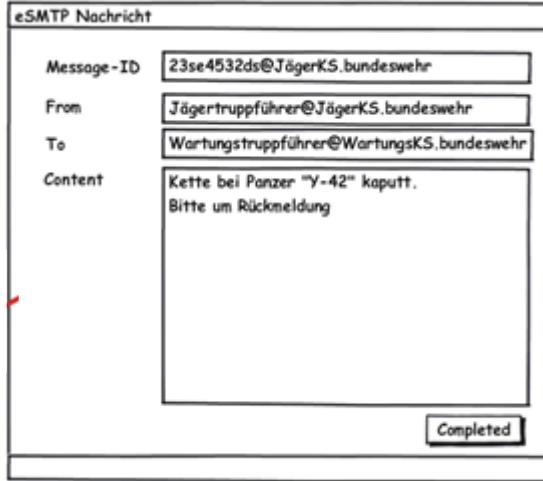
Anwendungsfälle



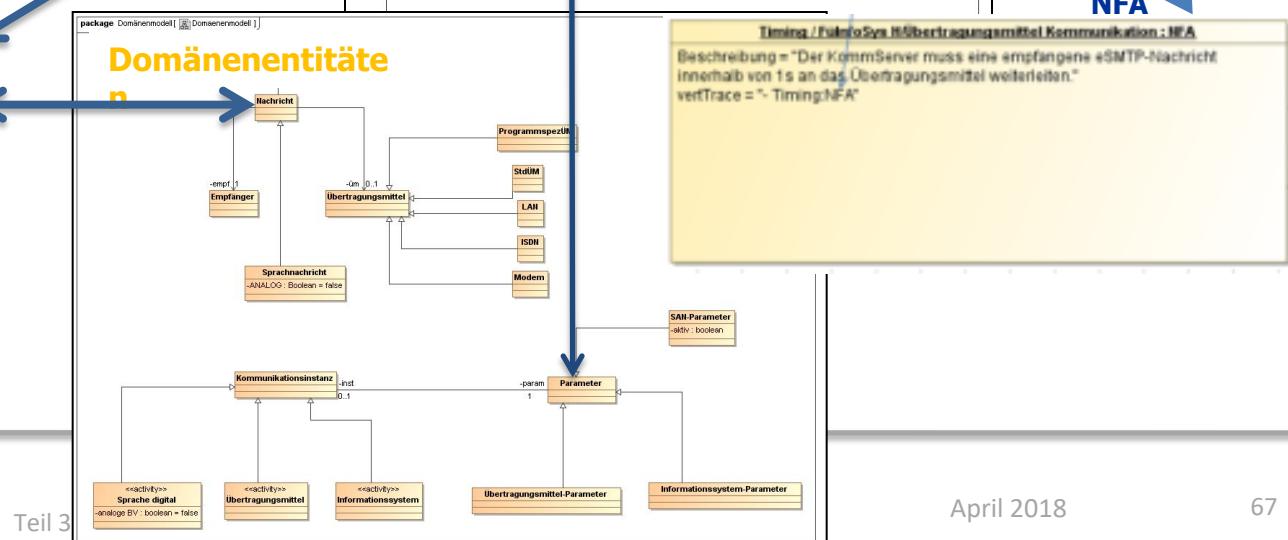
Aktivitäten



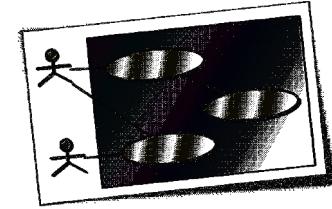
Schnittstellenmodell



Domänenentitäten



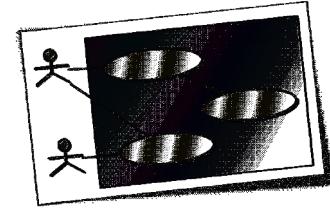
Textschablonen für Use Cases



Nummer, Name

Bezeichner	Eine eindeutige Bezeichnung, Identifier
Kurzbeschreibung	ein kurzer erklärender (strukturierter) Satz zur Übersicht
Akteure	Personen (Rollen) oder externe Systeme, die aktiv mit dem System interagieren oder einen Nutzen von dem Anwendungsfall haben. Ein Anwendungsfall kann mit mehreren Akteuren verbunden sein.
Kategorie	muss, soll, oder kann der Anwendungsfall realisiert werden?
Auslöser	ein Akteur oder eine Funktion [...] die den Ablauf startet.
Vorbedingung	eine Bedingung, die erfüllt sein muss, damit der Ablauf gestartet wird – beim Auto muss zum Beispiel der Schlüssel im Schloss stecken, damit man starten kann.
Eingabe	für den Ablauf benötigte Informationen
Ausgabe	Ergebnis des Ablaufs
Nachbedingung	eine Bedingung, die erfüllt sein muss, um den Anwendungsfall zu beenden.
Ablauf	beschreibt den Standardablauf – was soll passieren, wenn der Anwendungsfall gestartet ist? Achtung, hier werden keine Ausnahmen behandelt!

Textschablonen für Nicht-Funktionale Anforderungen



Nummer, Name

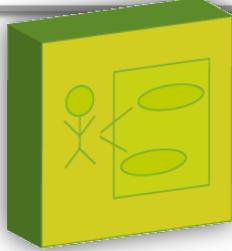
Bezeichner	Eine eindeutige Bezeichnung, Identifier
Kurzbeschreibung	ein kurzer erklärender (strukturierter) Satz zur Übersicht
Kategorie	muss, soll, oder kann die Nicht-Funktionale Anforderung realisiert werden?
Erläuterung	Motivation und Hintergründe für die Anforderung. Warum ist sie wichtig? Was passiert wenn sie nicht erfüllt wird?
Lösungsidee	Erste Überlegungen wie diese Nicht-Funktionale Anforderung umgesetzt werden könnte
Betroffene Use-Cases	Verweis zu den Use-Cases bei denen diese Nicht-Funktionale Anforderung gelten muss

Beispiel: Terminverwaltung

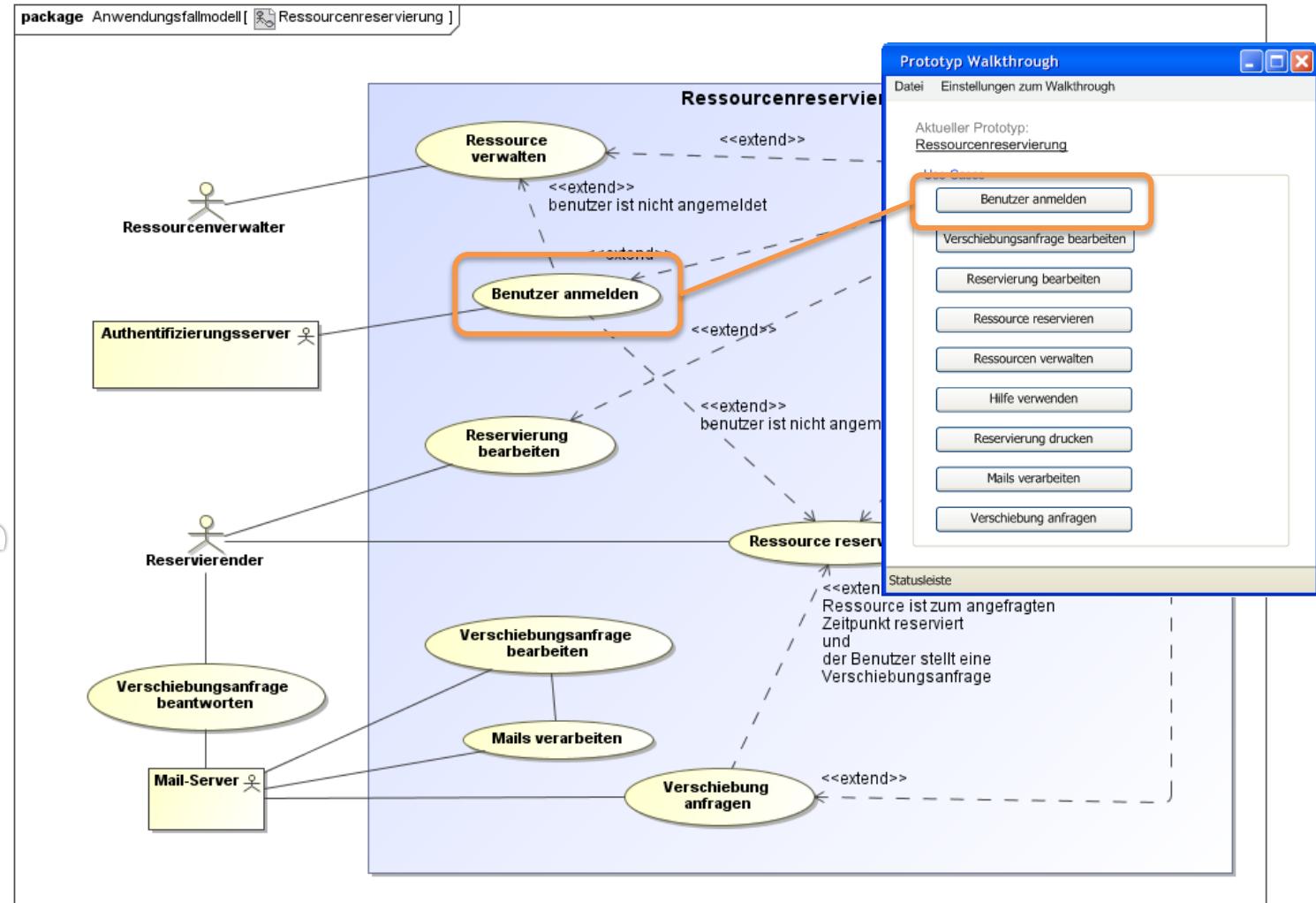
Ausgangspunkt: Funktionale Anforderungen

- ...
- **Verwendungszweck**
 - Hauptaufgabe der Ressourcenverwaltung ist es, das Reservieren von Räumen und mobilen Ressourcen wie zum Beispiel von Laptop und Beamer, zu unterstützen.
- **Geforderte Funktionen**
 - Das System muss dem Benutzer die Möglichkeit bieten, sich anzumelden
 - Das System muss dem Benutzer Verwalter die Möglichkeit bieten, Ressourcen zu verwalten
 - Das System muss dem Benutzer die Möglichkeit bieten, Ressourcen zu reservieren
 - Das System muss dem Benutzer die Möglichkeit bieten, Reservierungen zu bearbeiten
 - Das System soll überprüfen, ob eine ausgewählte Reservierung möglich ist (Ressource bisher nicht Reserviert: Start und Ende der Reservierung überschneidet sich mit keiner bestehenden Reservierung)
 - ...
- ...

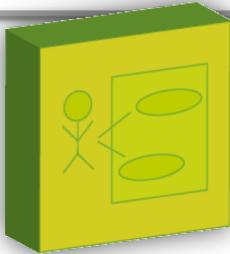
Anwendungsfallmodell



Use-Case-Diagramm



Anwendungsfallmodell



Use-Case-Tabelle

Funktionale Forderung
aus Textueller
Beschreibung

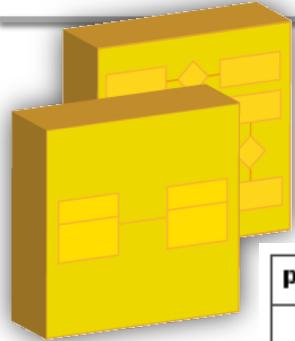
Benutzer anmelden	
Kurzbeschreibung	Das System muss dem Benutzer die Möglichkeit bieten, sich anzumelden
Akteure	Benutzer
Kategorie	Muss
Auslöser	Der Reservierende betätigt den Button „anmelden“
Vorbedingung	Der Benutzer ist nicht angemeldet. Der Benutzer hat ein gültiges Benutzerkonto (Benutzername, Passwort).
Eingabe	Benutzername, Passwort
Ausgabe	Bestätigung der Anmeldung.
Nachbedingung	Der Benutzer ist am System angemeldet.
Standardablauf	Das System präsentiert dem Benutzer ein Dialogfenster, in dem der Benutzer seinen Benutzernamen und sein Passwort eingeben kann. Der Benutzer gibt seinen Benutzernamen ein. Der Benutzer gibt sein Passwort ein. Der Benutzer bestätigt die Eingabe. Das System überprüft Benutzername und Passwort. Das System meldet den Benutzer an. Das System zeigt dem Benutzer die Information an, dass er angemeldet ist.

Anwendungsfallmodell

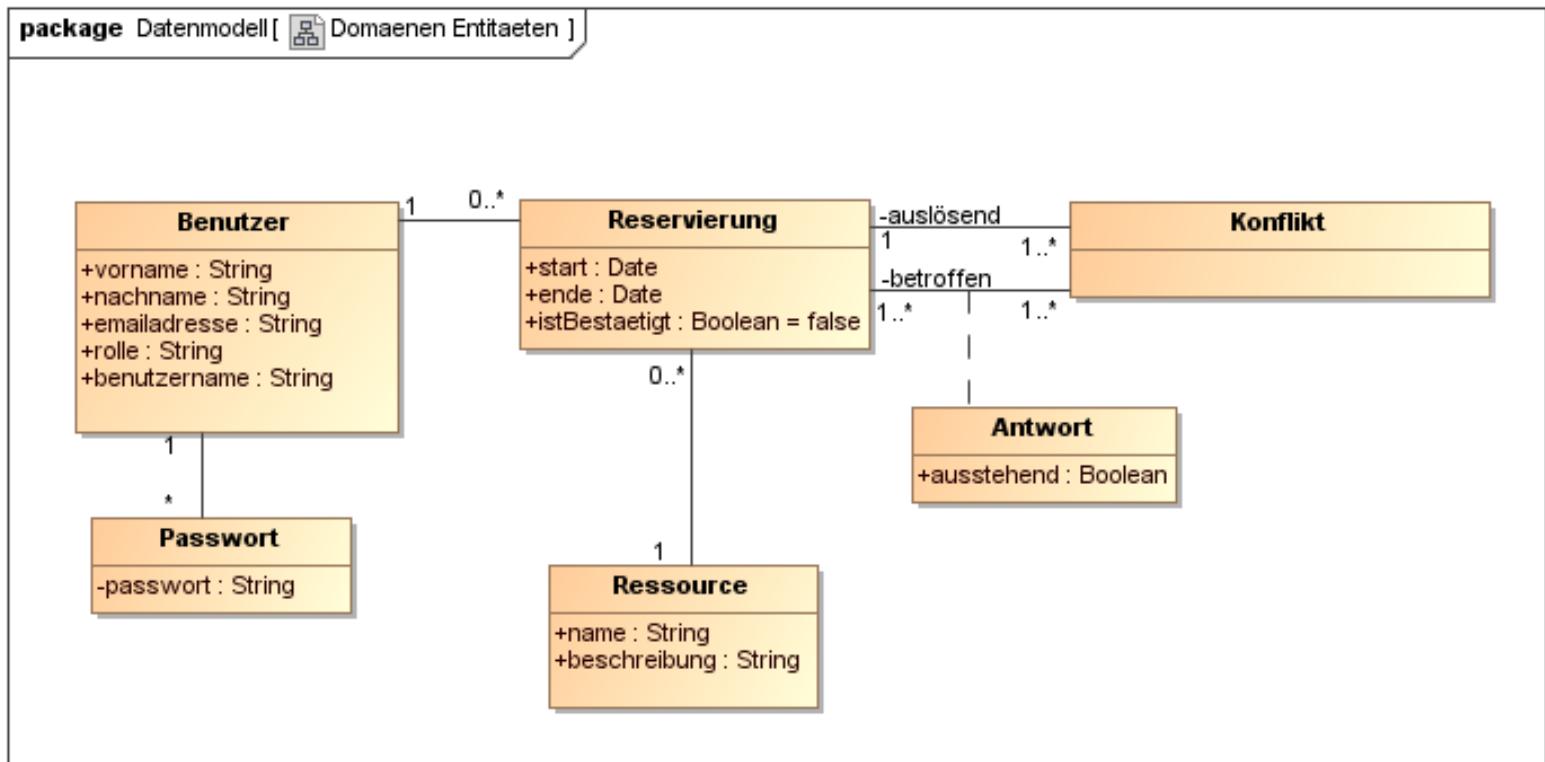


NFA1: Aktuelle Daten	
Kurzbeschreibung	Das System muss dem Benutzer innerhalb von 5 Sekunden aktuelle Daten zeigen
Kategorie	Muss
Erläuterung	Es ist notwendig aktuelle Daten anzuzeigen. Nur so können Konflikte bei der Reservierung reduziert werden.
Lösungsidee	<pre>sequenceDiagram participant User1 as 1:Benutzer participant User2 as 2:Benutzer participant System as :System User1->>System: setName() activate System System->>User1: show new data System->>User2: show new data activate User1 activate User2 User1-->>System: show new data User2-->>System: show new data deactivate User1 deactivate User2 activate System System-->>System: re-show GUI activate System note over System: { ≤ 5 sek }</pre>
Betroffene Use-Cases	UC1, UC2, UC5

Datenmodell

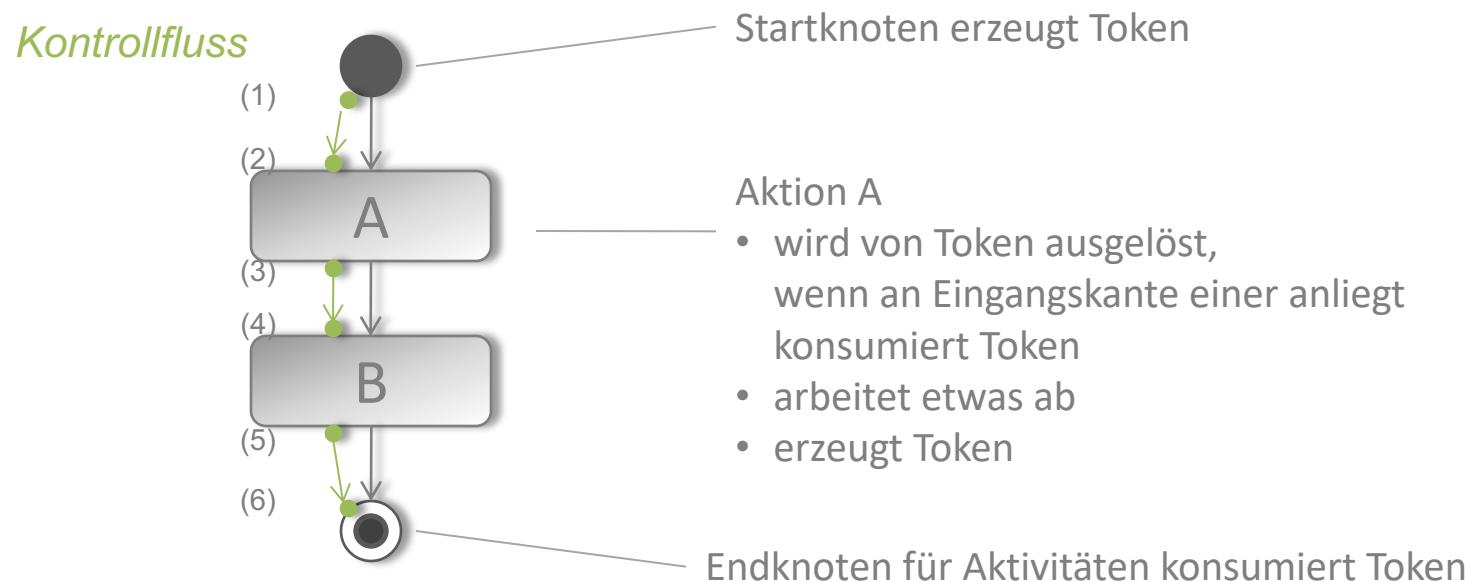


Klassendiagramm



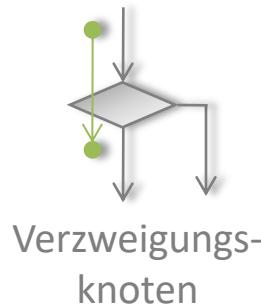
Allgemein: Token-Semantik in Aktivitätsdiagrammen (1)

- Kontrollfluss und Datenfluss mit Token-Semantik
 - Kontrollfluss über **Kontroll-Token**
 - Datenfluss über **Daten-Token**
 - Token (bildlich)
 - als Marke vorstellbar
 - wird an verschiedenen Punkten im Ablauf „weitergegeben“

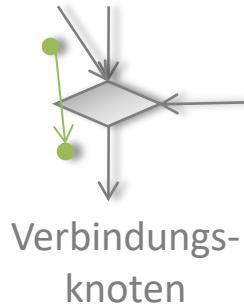


Allgemein: Token-Semantik in Aktivitätsdiagrammen (2)

- Kontrollfluss und Datenfluss über Token-Semantik
 - Kontrollflussknoten

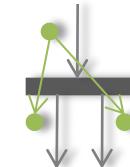


Verzweigungs-knoten



Verbindungs-knoten

ODER



Parallelisierungs-knoten

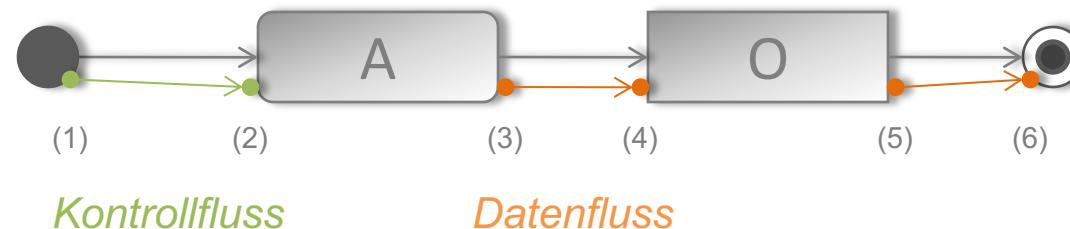


Synchronisations-knoten

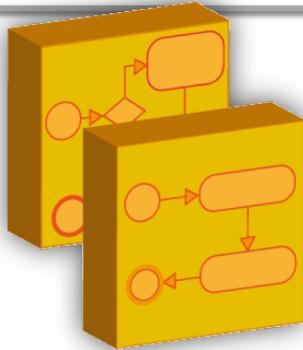
UND

Allgemein: Token-Semantik in Aktivitätsdiagrammen (3)

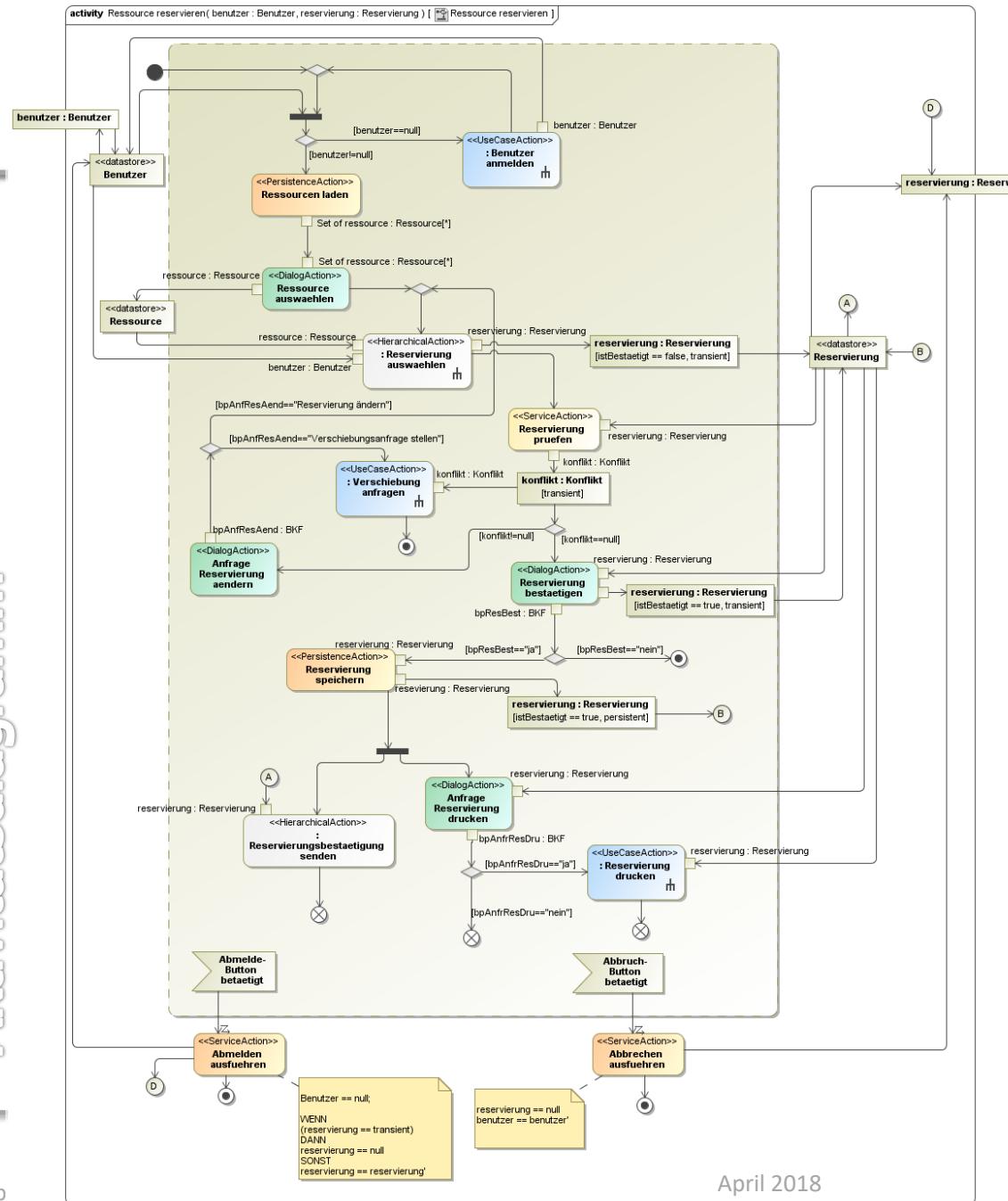
- Kontrollfluss und Datenfluss über Token-Semantik
 - Daten-Token
 - dienen als Transportmittel für Daten oder Werte
 - bewegen sich über Kanten, an denen mindestens ein Objektknoten beteiligt ist.
 - In einen Objektknoten eingehende Token repräsentieren Daten oder Werte, die in dem Objektknoten gesetzt bzw. gesammelt oder durch den Objektknoten repräsentiert werden
 - Aus einem Objektknoten herausgehende Token transportieren das Objekt selbst, bzw. Daten des Objekts in die Folgeaktion.
 - Der Objektknoten an sich stellt jedoch kein Objekt bzw. keinen Datenwert dar.



Aktivitätsmodell

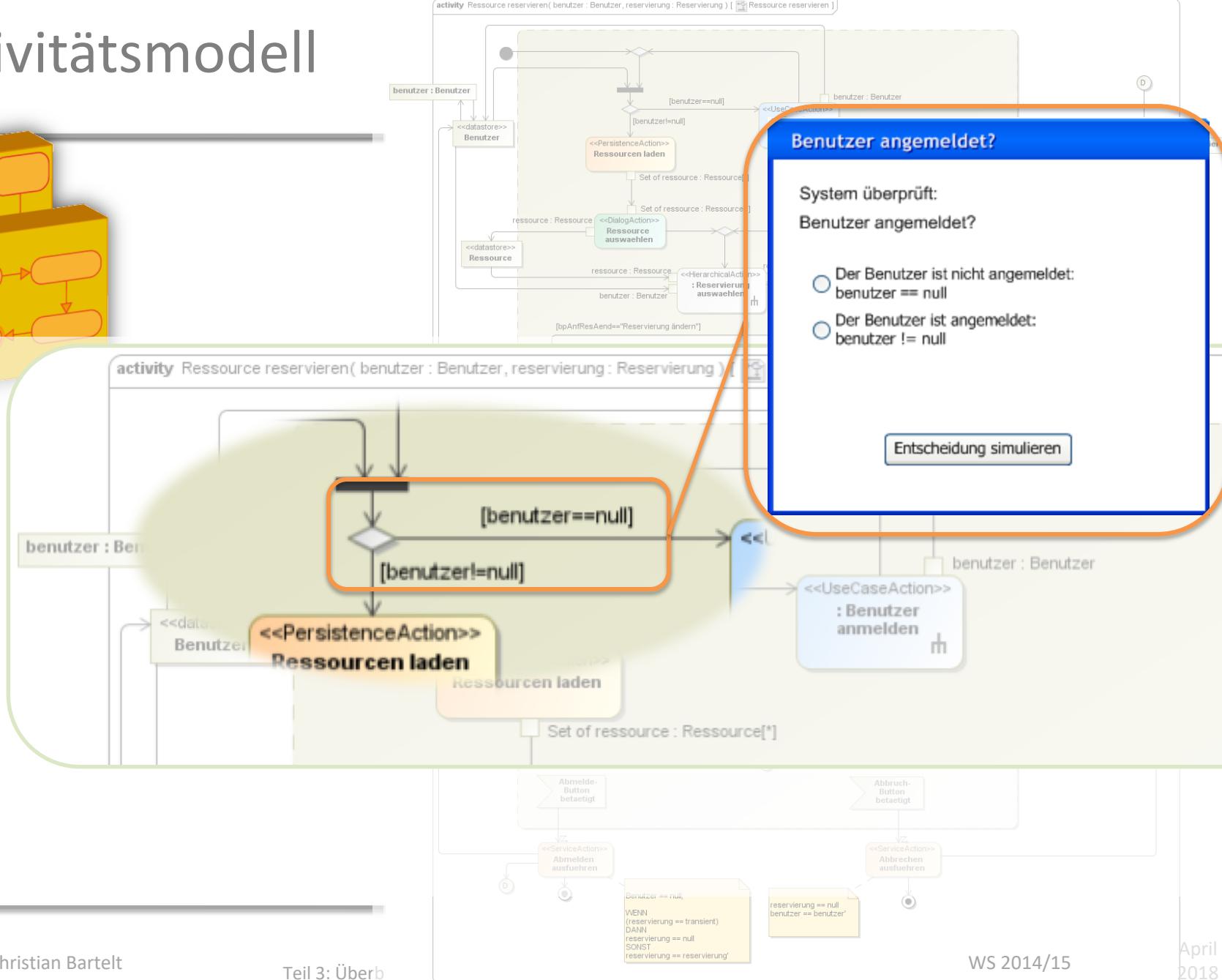
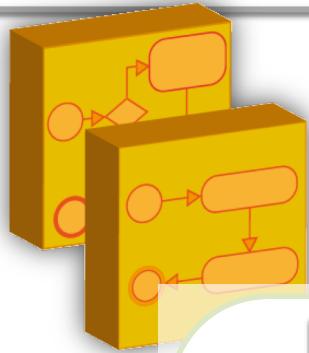


Aktivitätsdiagramm



Aktivitätsmodell

Aktivitätsdiagramm



Aktivitätsmodell

Reservierungskonflikt?

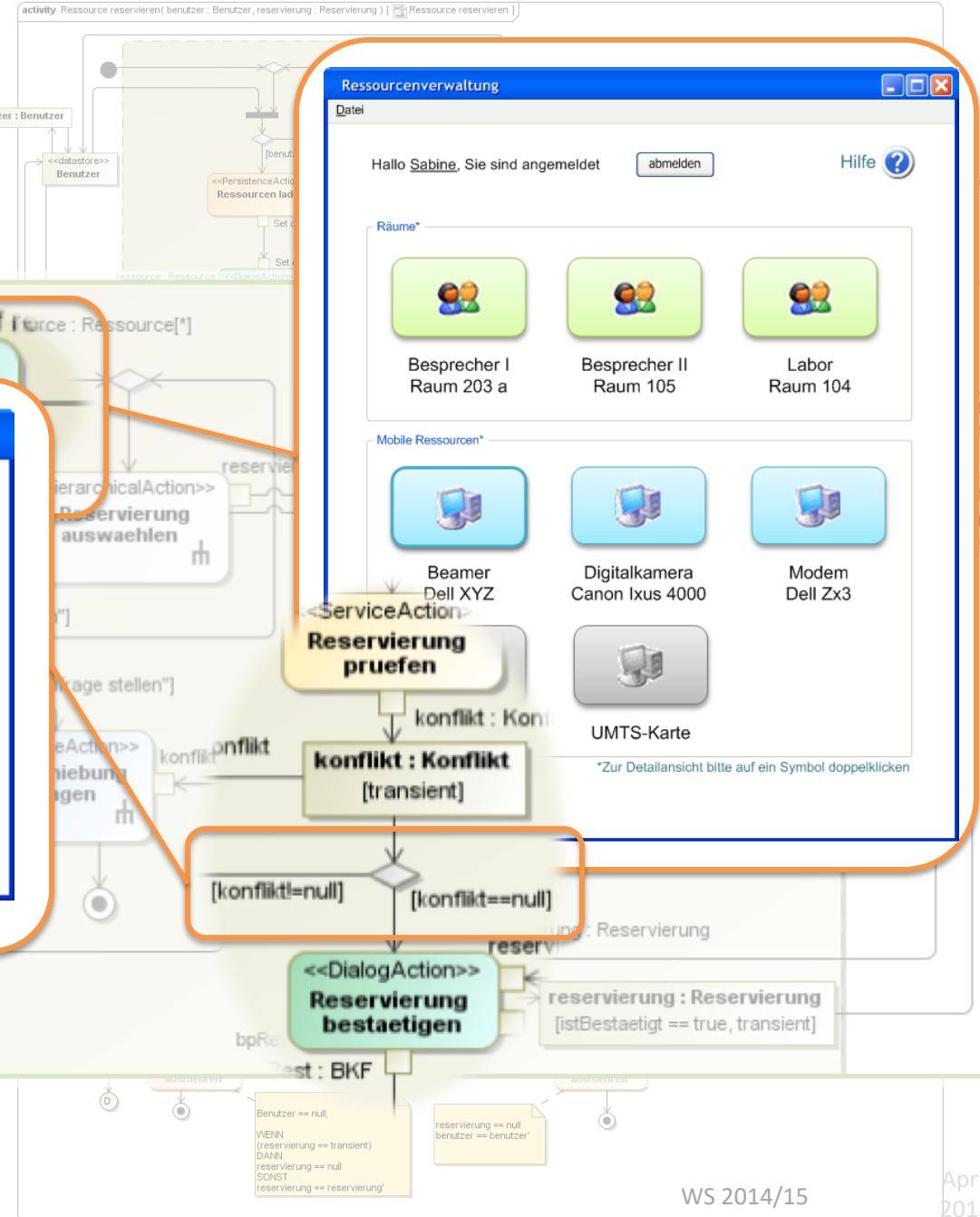
System überprüft:

Gibt es einen Reservierungskonflikt?

- Es gibt keinen Konflikt:
konflikt == null
- Es gibt einen Konflikt:
konflikt != null

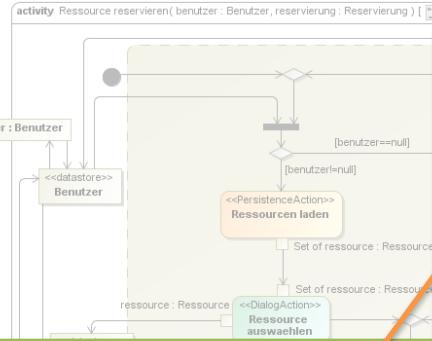
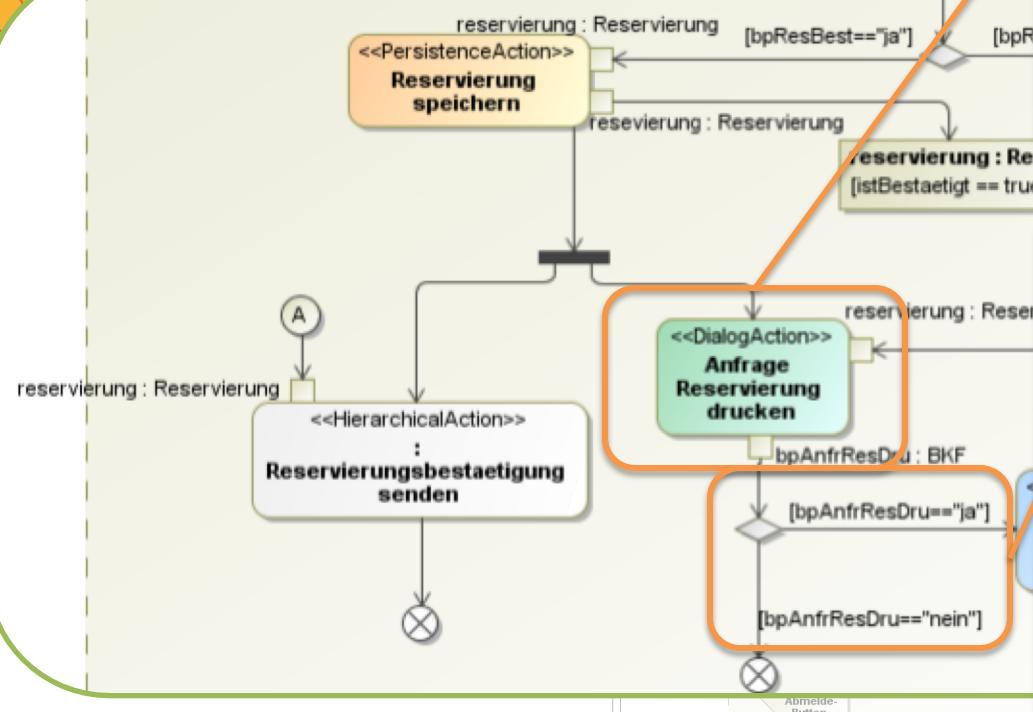
Entscheidung simulieren

Reservierung
ändern



Aktivitätsmodell

Aktivitätsdiagramm



Reservierungsbestätigung

Raum 203 a ist für Sabine

von Mo, 04.08.2008; 09:00 Uhr
bis Mo, 04.08.2008; 10:00 Uhr

reserviert.

Reservierung drucken und Fenster Schließen

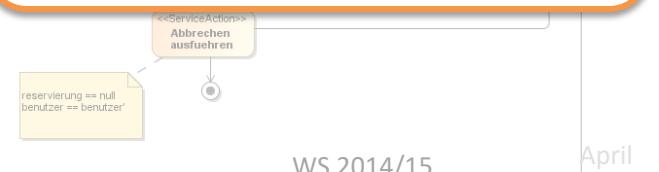
Bestätigung schließen

Reservierung drucken?

System überprüft:
Reservierung drucken?

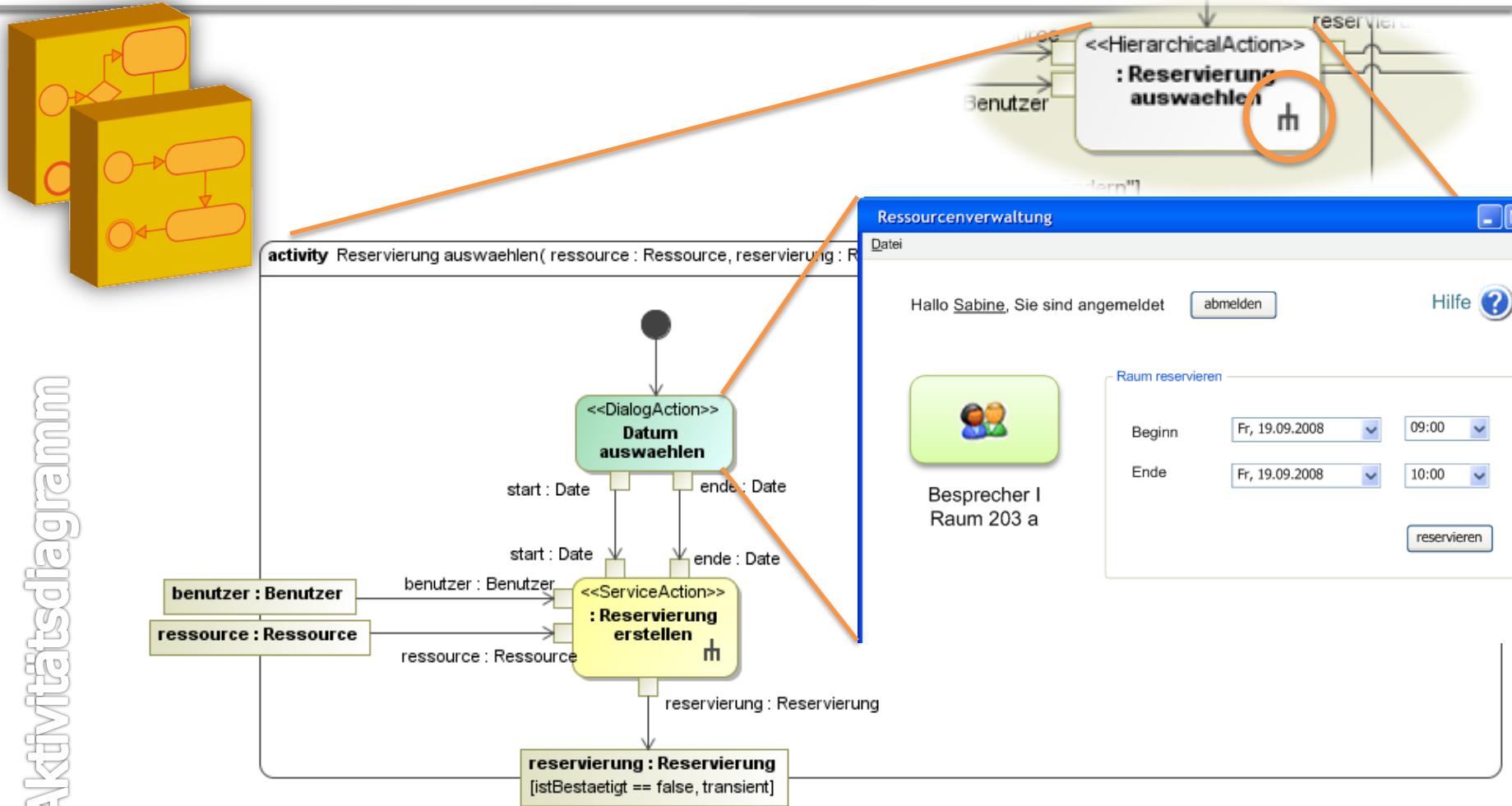
- Es soll gedruckt werden:
bpAnfrResDru == „ja“
- Es soll nicht gedruckt werden:
bpAnfrResDru == „nein“

Entscheidung simulieren

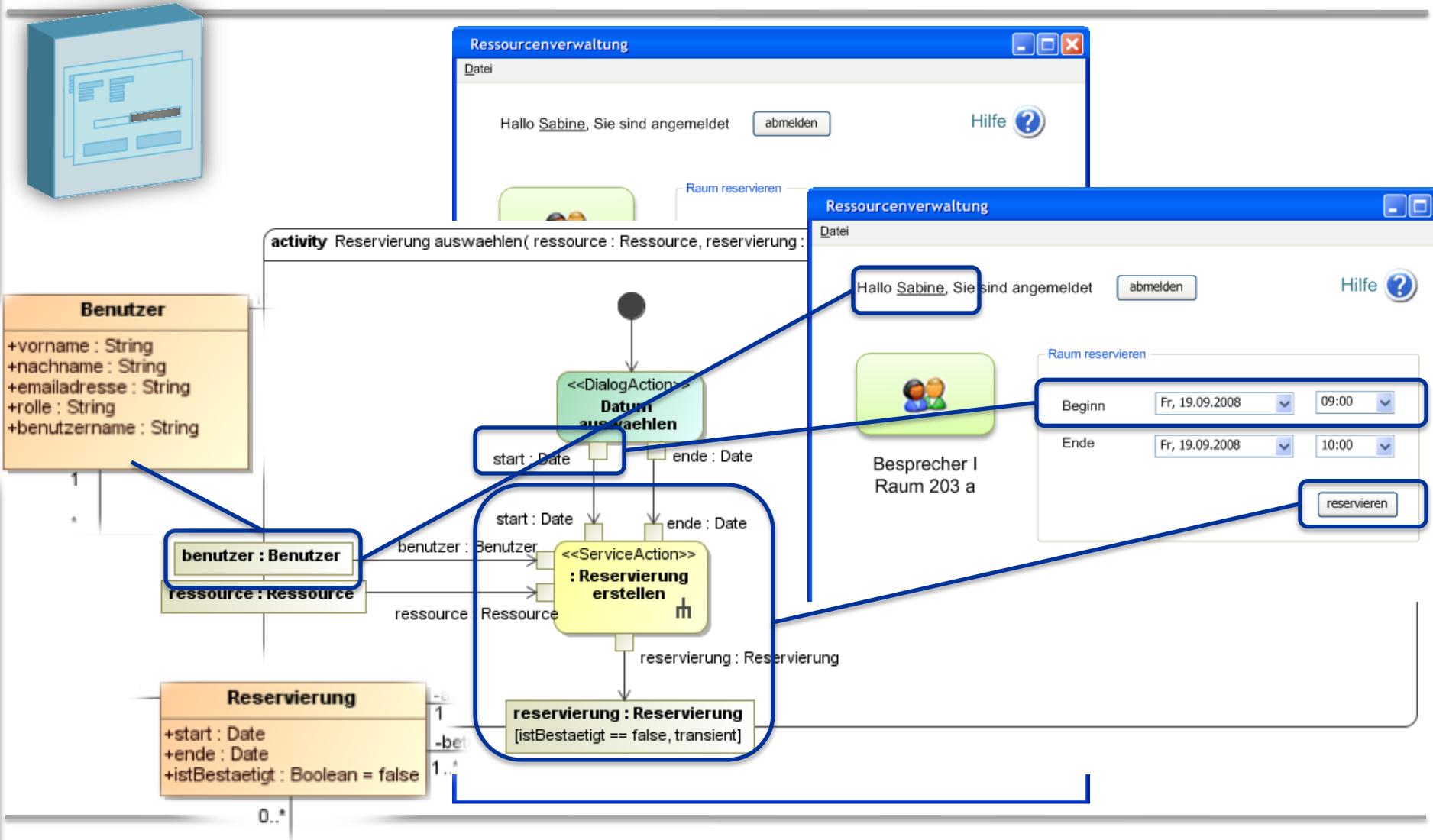


Aktivitätsmodell

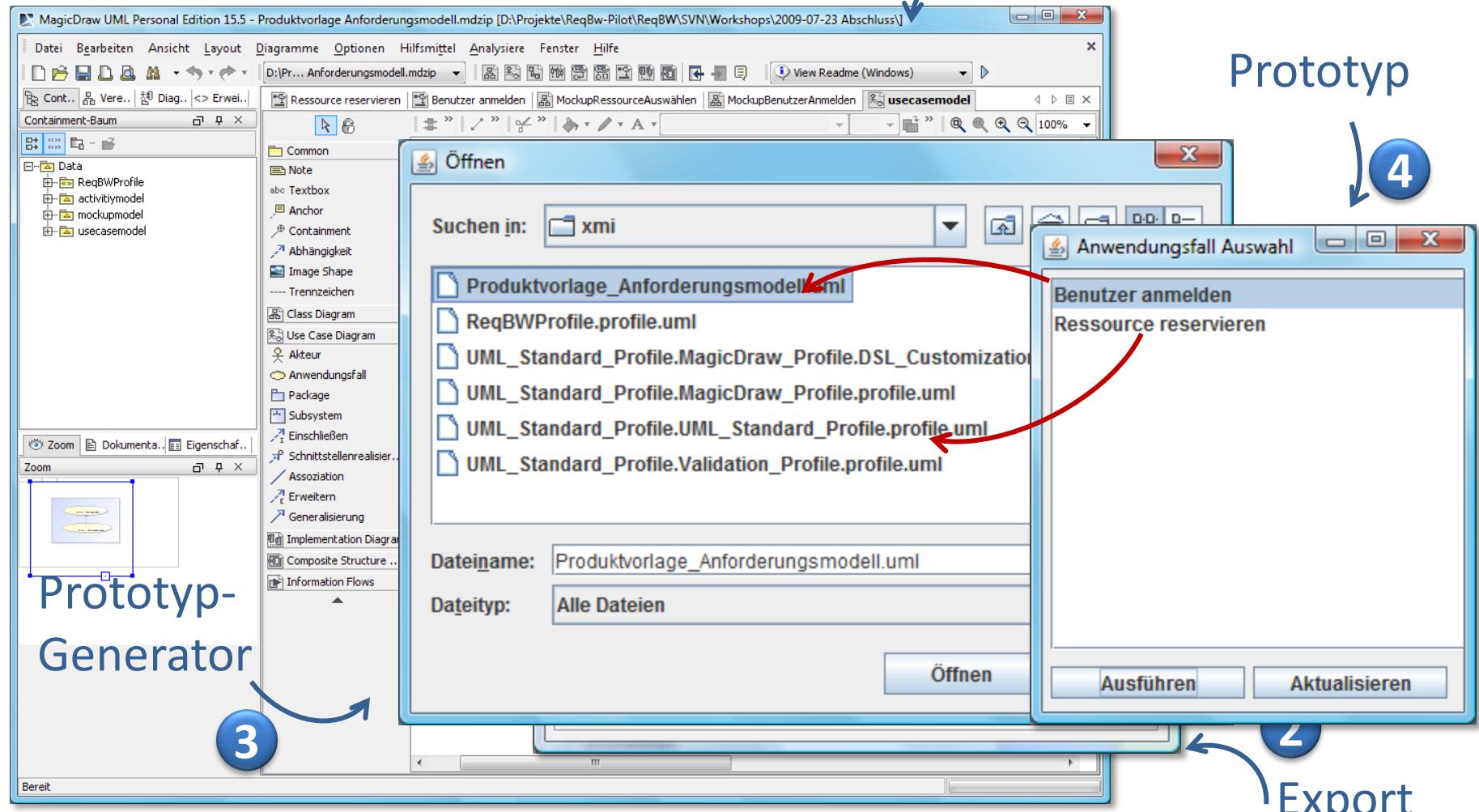
Aktivitätsdiagramm



Screen-Mock-up

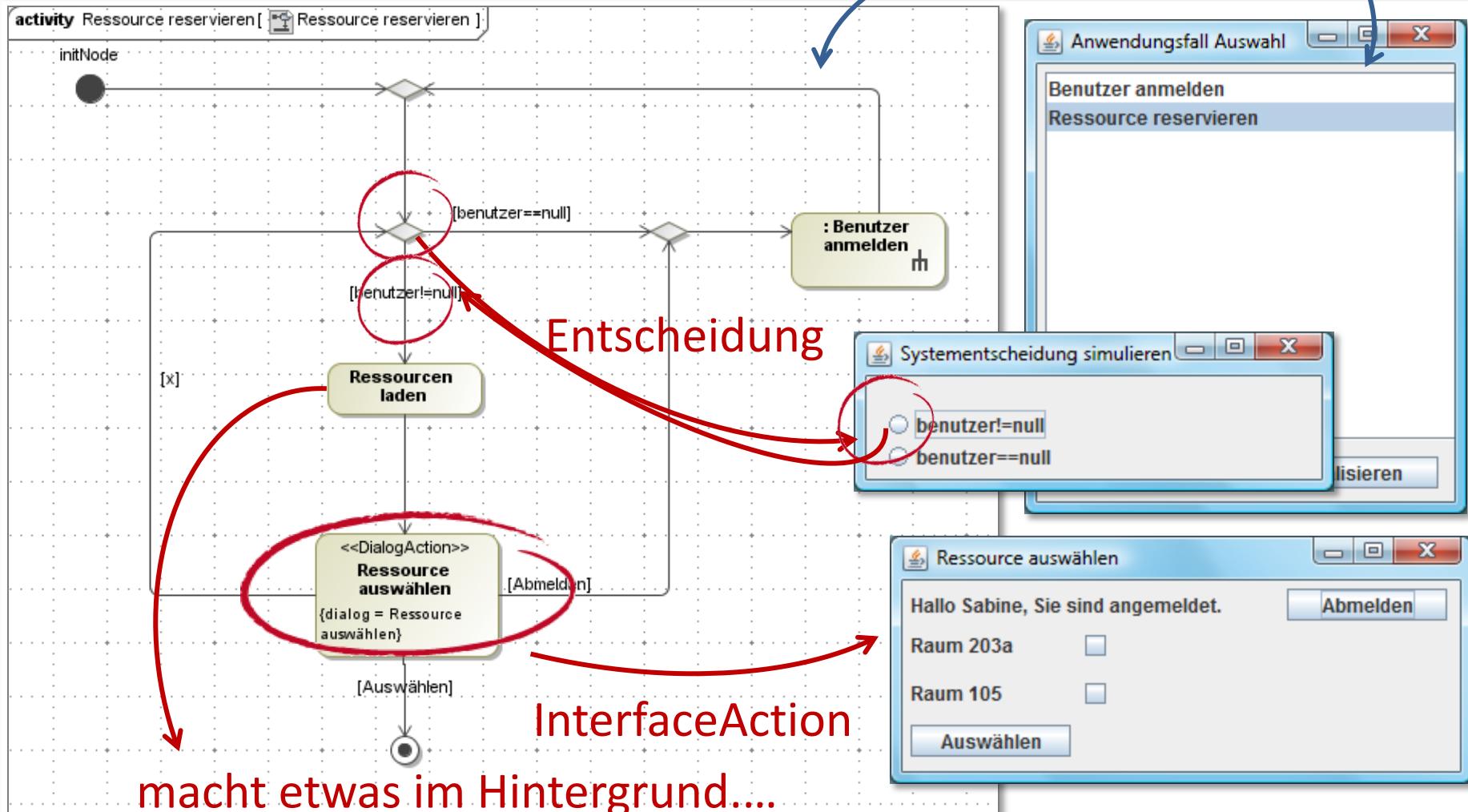


Tool Support und Demo (1)



Tool Support und Demo (2)

CASE-Tool Prototyp



Blick ins Dokument....

1

4SOFT

4Soft-GmbH
Mitterstraße 3
80336 München
www.4soft.de

-Systemsspezifikationen:Gesamtsystemsspezifikation-(Pflichtenheft).¶

PAPXL-Gesamtsystemsspezifikation¶

Version:·1.41¶

Projektbezeichnung	PAPXL – Projektassistent Planung extended&enlarged¶ (GSE Praktikum 2006)¤
Projektleiter¤	Thomas Termit¤
Verantwortlich¤	Michael Storck → Anforderungsanalytiker(AN)¤
Erstelltam¤	03.05.2006¤
Zuletzt geändert¤	17.11.2013 12:16¤
Bearbeitungszustand¤	<input checked="" type="checkbox"/> in Bearbeitung <input checked="" type="checkbox"/> vorgelegt¤ <input checked="" type="checkbox"/> fertig gestellt¤
Dokumentablage¤	\Produkte\Systemsspezifikationen\Gesamtsystemsspezifikation-(Pflichtenheft).doc¤
V-Modell-XT Version¤	Version 1.2.0¤

Seitenumbruch

thland)