MIXED MODE

1) Requirements – Management

2) Use Cases

3) Fallbeispiel (UC, OOA, OOD)

Dipl.-Ing. Andrea Asprian

Mixed Mode GmbH – Lochhamer Schlag 17 – 82166 Gräfelfing – www.mixed-mode.de

Inhalt und Vorgehensweise

- Einführung in das Requirements - Management
  - Requirements?
  - Requirements im Entwicklungsprozess
  - Arten von Requirements
- Requirements aufdecken mit Use Cases und UML
  - Use Cases?
  - Use Cases & UML
  - Erstellen von Use Cases
- Fallbeispiel
  - Requirements aufnehmen
  - OOA des Problems
  - Erstellen eines OO Designs

1. Einführung in das Requirements - Management

Requirements

Requirement = Anforderung

zur Spezifikation von Software

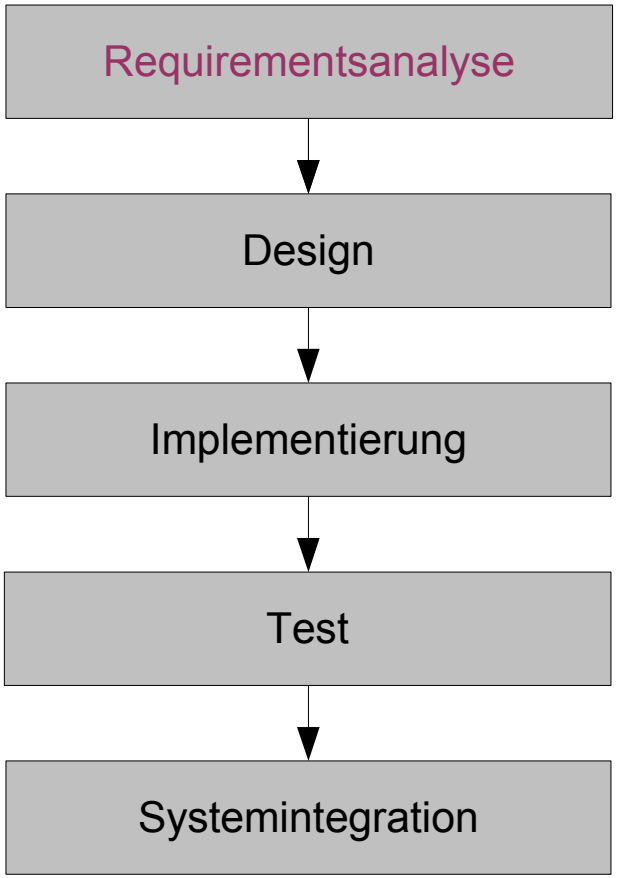
Definition eines benötigten Programms aus der Sicht des Auftraggebers

1. **Was** soll die Software leisten? (Nicht: Wie... !)
2. Unter welchen Bedingungen, in welchem Umfang,

   mit welcher Performance, ... soll die Software das leisten?

Requirementsanalyse in Entwicklungsprozessen

Phasenmodell (klassisch): Wasserfall



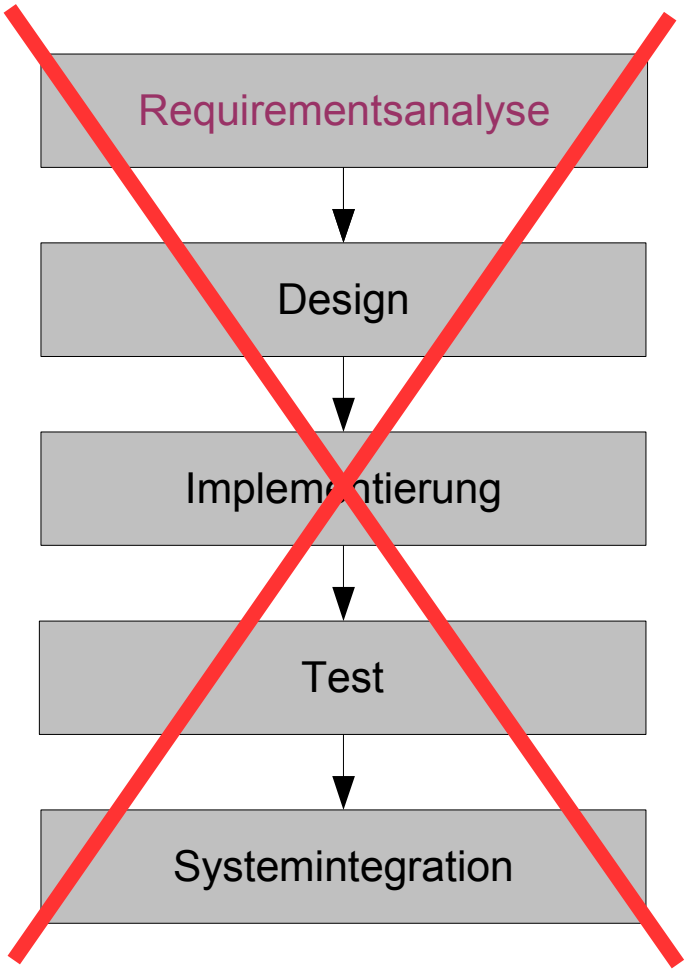
- Sequentielle Abarbeitung der Phasen
- Vollständige Beendigung

  einer Phase vor Eintritt in die folgende
- 1. Phase: Requirements

  Werden nach Abschluss nicht mehr

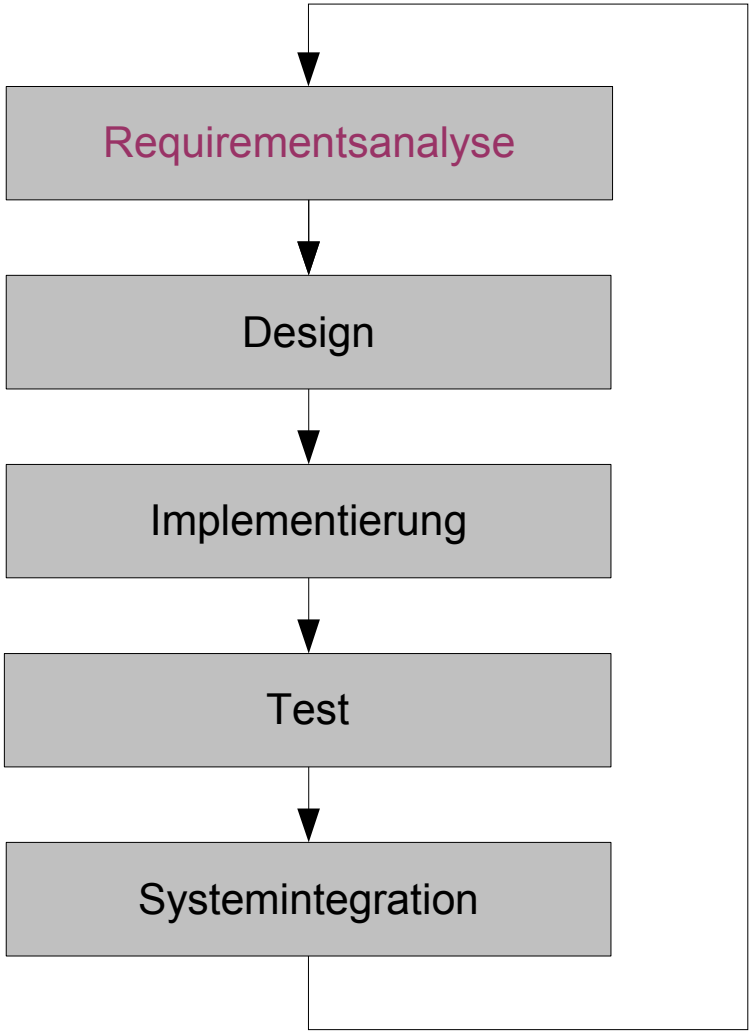
  angetastet

Phasenmodell (klassisch): Wasserfall



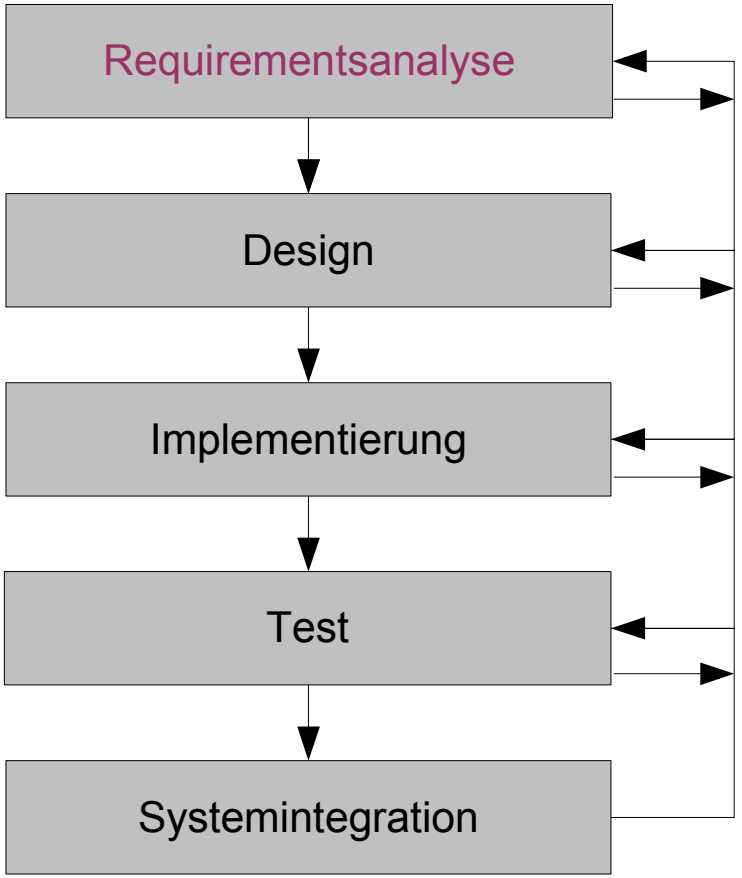
- In heutigen Softwareentwicklungsprojekten nicht gewährleistet
- Unrealistisch

Phasenmodell mit Iteration (1/2)



- Sequentielle Abarbeitung der Phasen
- Iterationen
- Beliebig viele Iterationen

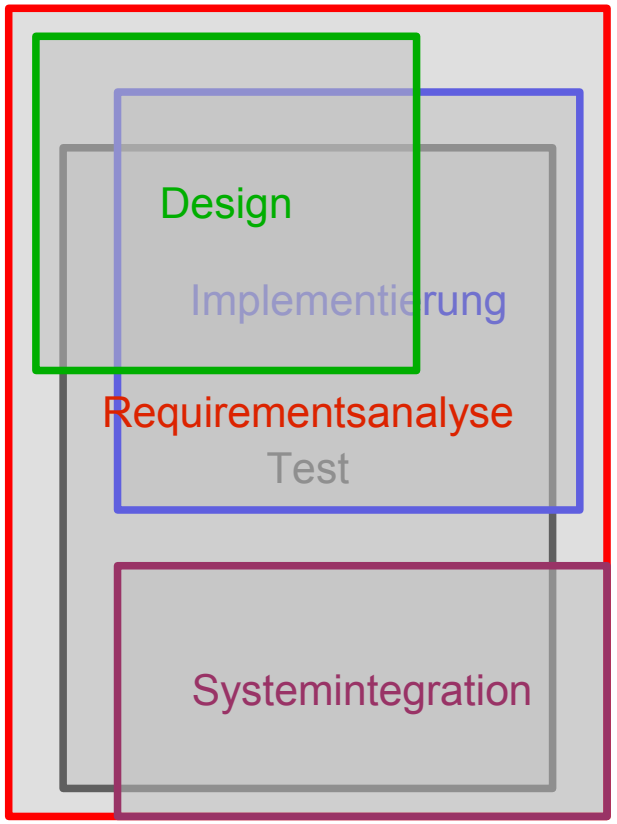
Phasenmodell mit Iteration (2/2)



- Sequentielle Abarbeitung
- Rücksprung in jede beliebige frühere Phase

  nach jeder abgeschlossenen Phase
- Flexibelste Variante des Phasenmodells

Agile Prozesse



- Sequentieller Ablauf aufgehoben
- Sprung in jede Phase zu jedem Zeitpunkt
- Parallele Bearbeitung verschiedener Phasen

"Evolutionäre" Softwareentwicklung

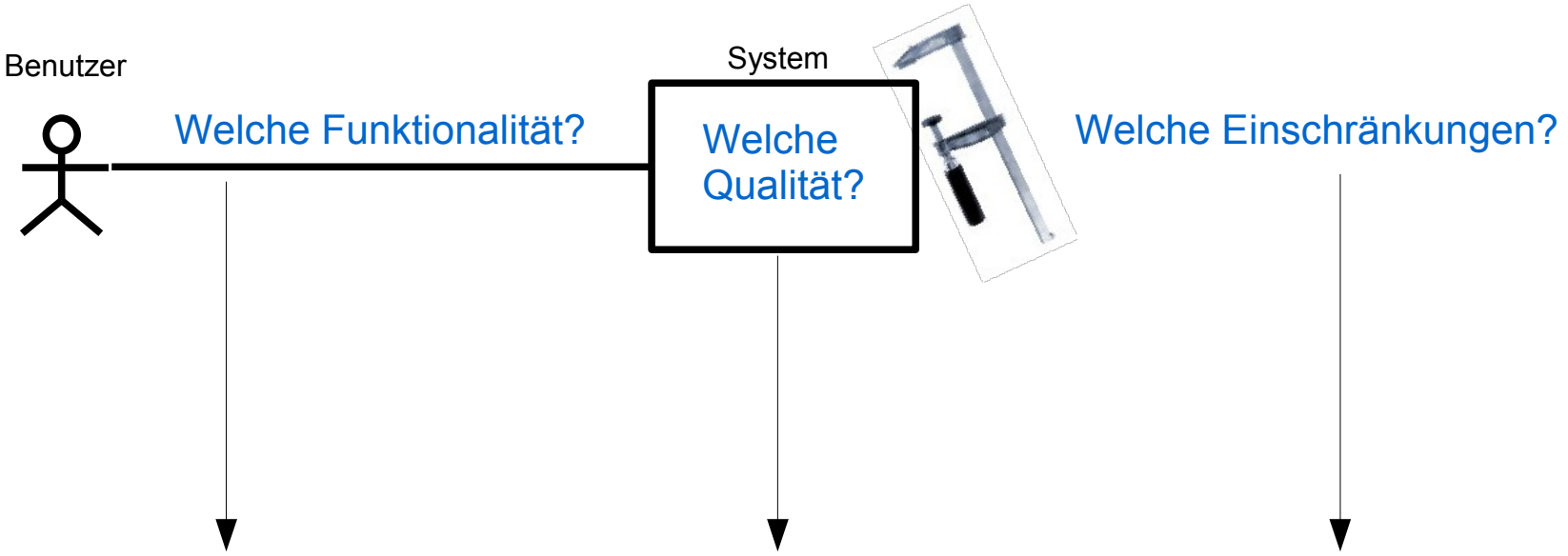
Requirements

- verändern sich
- entstehen neu
- verschwinden

während des gesamten Entwicklungsprozesses

Requirements - Typen

Typen von Requirements



Functional Requirements

vom System angebotene Funktionalität
“Zweck”

Das Navigationssystem berechnet die Route in Abhängigkeit von der aktuellen Position.

Non-functional Requirements

Eigenschaften / Qualität der Ausführung

Die Berechnung der Route darf max. 4 Sekunden dauern.

Constraints

Vorgaben durch Umgebung

Die Software für das Navigationssystem soll auf einem Linux-System laufen.

Checkliste für Requirements: FURPS+

Functional Requirements

Non-functional Requirements

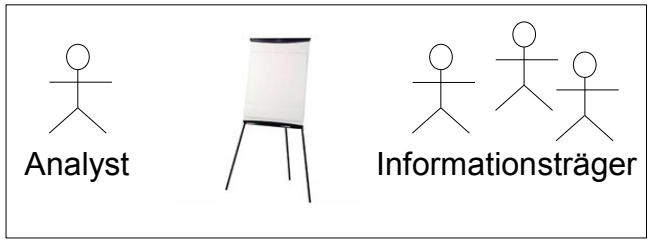
Functionality	Funktionalitäten, Fähigkeiten, Sicherheit
Usability	Verständlichkeit, Erlernbarkeit, Bedienbarkeit
Reliability	Fehlertoleranz, Wiederherstellbarkeit
Performance	Zeitverhalten, Verbrauchsverhalten
Supportability	Anpassbarkeit, Austauschbarkeit, Modifizierung
Implementation	Resourcen Limits, Sprachen, Tools, Hardware
Interface	Schnittstellen zu externen Systemen
Operations	Entwicklungsprozesse, Organisationsstrukturen
Legal	Lizenzen, Gesetzgebung

Constraints

Lebenszyklen von Requirements

Erfassung

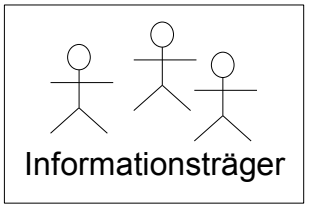
Requirements Workshop



Functional Requirements
Non-functional Requirements
Constraints

Änderungen

Change Management



Änderungswünsche

Dokumentation

Requirements Specification = Vertrag über das zu entwickelnde System

Erfassen und Darstellen von Requirements

	Erfassung		Darstellung	
	Brainstorming	Use Case Analyse	Tabellarisch	Use Cases
1. Constraints	X		X	
2. Functional Requirements		X		X
3. Non-functional Requirements	X	X	X	X

Methoden zur Ermittlung von Requirements

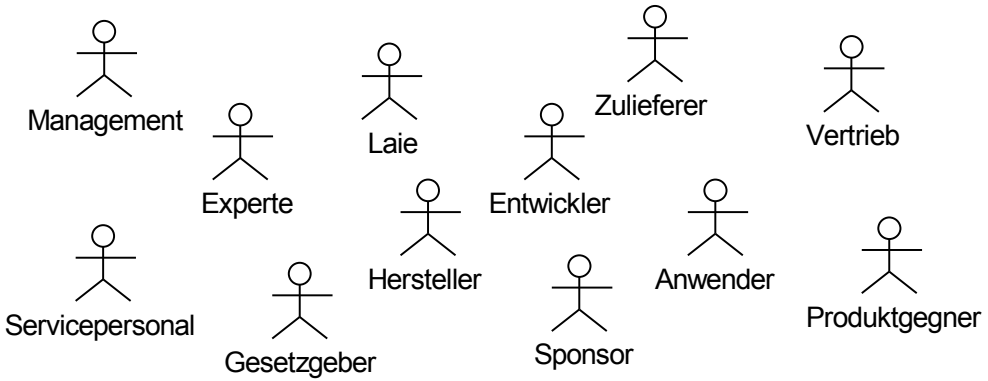
Requirementsanalyse

	Aktion		Artefakt
1	Erfassung der Informationsträger		Stakeholder List
2	Erfassung der Aktoren	+	Actor List
3	Festlegung des Kontext	+	Context Diagram
4	Benutzerziele identifizieren	+	User Goal List
5	Use Cases formulieren	+	Use Cases
6	Ergänzende Spezifikation verfassen	+	Supplementary Specification

= Specification

1. Erfassung der Stakeholder

Stakeholder sind alle Personen, Einrichtungen, ... die von der Systementwicklung sowie vom Einsatz und Betrieb des Systems betroffen sind.

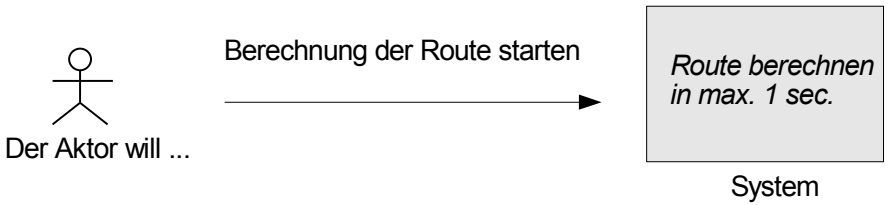


Vollständige Liste von Stakeholdern = Vollständige Liste von Informationsquellen

vergessene Stakeholder → vergessene Anforderungen

2. Erfassung der Aktoren

Aktoren sind Personen (bzw. Systeme), die ein bestimmtes Ziel erreichen wollen und dazu den Service des Systems nutzen.

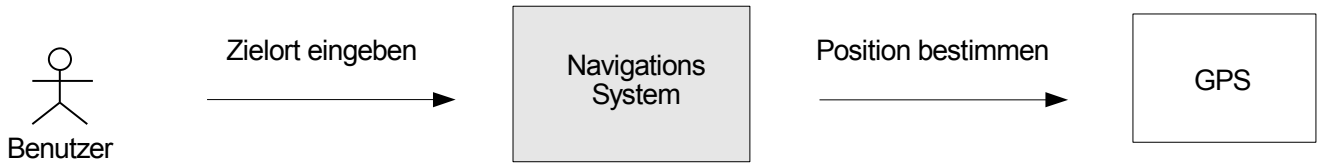


Vollständige Liste von Aktoren = Vollständige Liste von Systemfunktionalitäten

vergessene Aktoren → vergessene Systemfunktionalitäten

3. Festlegung des Systemkontext

Der Systemkontext legt die Grenzen des zu entwickelnden Systems und somit seine Schnittstellen zur Außenwelt fest.



System – Funktionalität realisieren:

- eigenständig
- mittels Fremdsystem ⇨ Schnittstelle

4. Identifizieren der Benutzerziele

Ein Benutzerziel (User Goal) ist derjenige Wunsch, den ein Aktor bei der Nutzung des Systems erfüllt haben möchte.

Überblick: Was soll das System leisten?

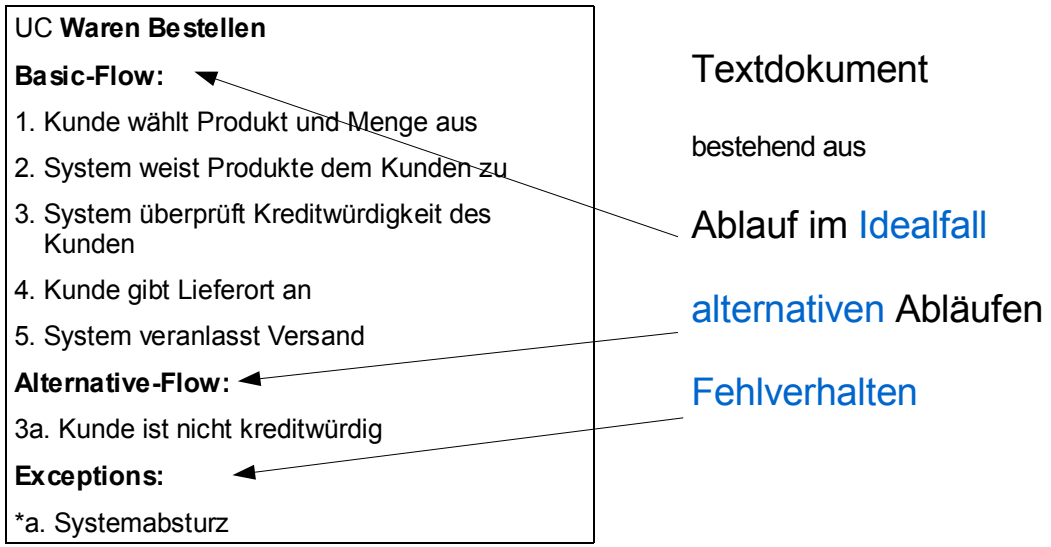
Aktor	Benutzerziel	Technische Schwierigkeit	Priorität	Use Case #	Zeitab-schätzung [md]	Release Datum
Benutzer	Eingabe des Zielorts	Mittel	1	1	10	31.12.05
Zeit	Neubestimmung der momentanen Position	Leicht	2	4	3	31.12.05

Actor – Goal – List

- Gliederung der Use Cases
- System – Funktionen und Prioritäten
- Kosten – / Zeitabschätzung

5. Formulieren der Use Cases (Functional Requirements)

Akteure nutzen den Service des Systems, indem sie Prozesse auslösen.
Ein Use Case beschreibt einen solchen Prozess.



Use Cases sind heutiger Standard um funktionale Requirements abzubilden.

6. Verfassen der ergänzenden Spezifikation (Non-functional Requirements)

Die ergänzende Spezifikation enthält Non-functional Requirements, die nicht in den Use Cases erfasst wurden.

Non-functional Requirement

im Kontext eines Use Cases

Non-functional Requirement

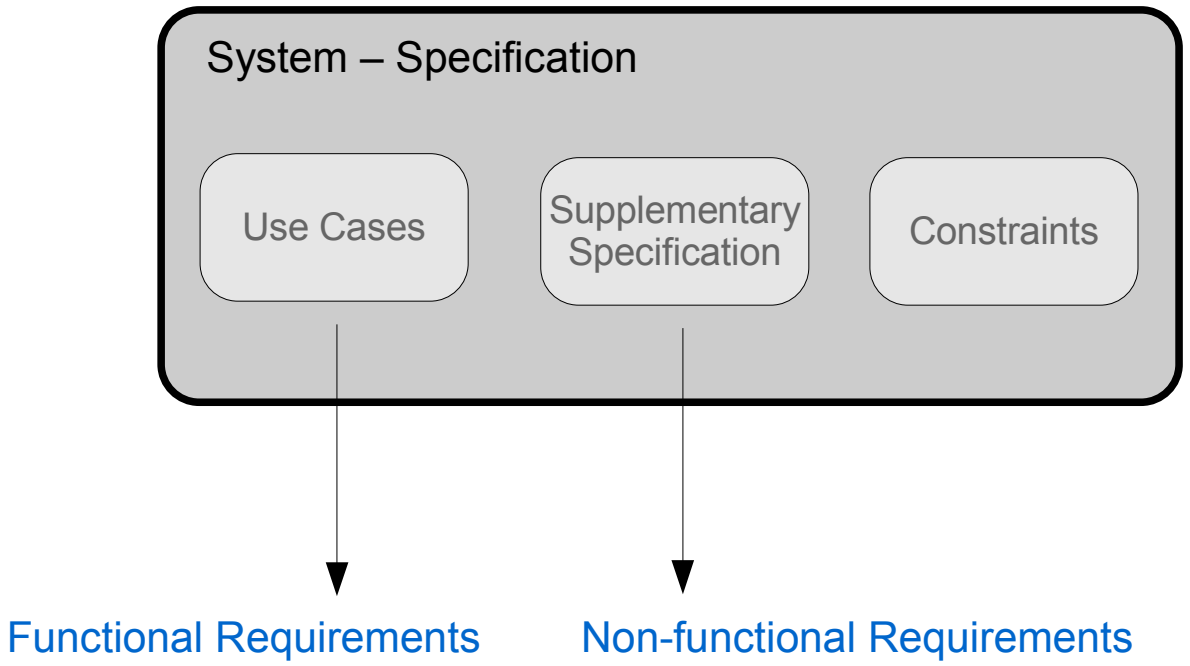
gesondert in der ergänzenden

Spezifikation

(Supplementary Specification)

Die aktuelle Position soll innerhalb von 0.5 Sekunden bestimmt werden.

Das System darf in 100 Betriebsstunden maximal einmal ausfallen und neu starten.



2. Requirements aufdecken mit Use Cases und UML

Use Cases und die Unified Modeling Language (UML)

Überblick

Was ist ein Use Case?

Texte,

die Abläufe

über die Zeit

beschreiben

Use Case: **Essen bestellen**

1. Kunde bestellt Essen bei Zustelldienst
2. Zustelldienst bereitet Essen zu und liefert Bestellung an Kunden
3. Kunde übernimmt Essen und bezahlt Rechnung

- **Business Use Case**
- **Unternehmen**
- **Black Blox**
- **kurze Fassung**
- **Benutzerebene**

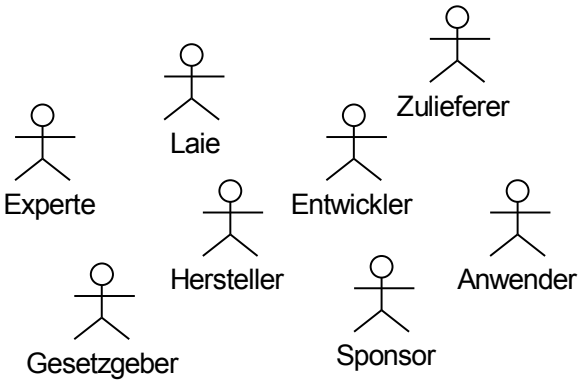
Wofür werden Use Cases verwendet?

- Workflows beschreiben
- Funktionelle (System-) Requirements finden
- Dokumentation (Abläufe, Vorgehensweise, ...)

Motivation

Einfacher Informationsaustausch zwischen Personen mit unterschiedlichem

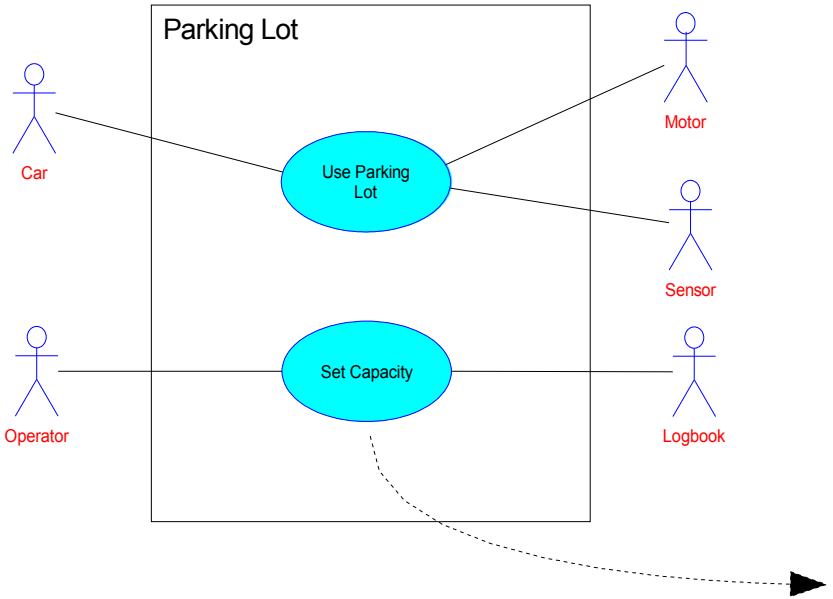
- Wissensstand und Vokabular
- Background und Verständnis
- Blickwinkel
- ...



Prinzip

Jeder Stakeholder versteht Use Cases

Use Case Diagramme



Use Case Text

Basic-Flow:
1. Benutzer wählt Maschinenparameter aus, den er neu setzen möchte
2. System zeigt momentanen Wert des Parameters an, der editiert werden kann
3. Benutzer gibt neuen Wert für den Parameter ein
4. System überprüft Wert auf Gültigkeit
5. System übergibt neuen Wert an Maschine
6. Maschine meldet korrekte Übernahme des neuen Wertes
7. System zeigt Benutzer die korrekte Übernahme des Wertes an und aktualisiert die Anzeige des Parameters

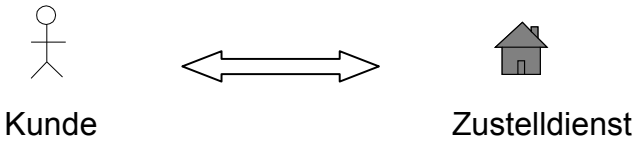
Alternative-Flows:
3a. eingegebener Wert ist ungültig
   1. System teilt dem Benutzer mit, dass der eingegebene Wert ungültig ist
   2. Benutzer bestätigt die Kenntnisnahme der Mitteilung
   3. System setzt den Wert auf den vorherigen zurück
   Szenario endet

Exceptions:
6a. Maschine liefert keine Rückmeldung
   1. ....
   2. ...

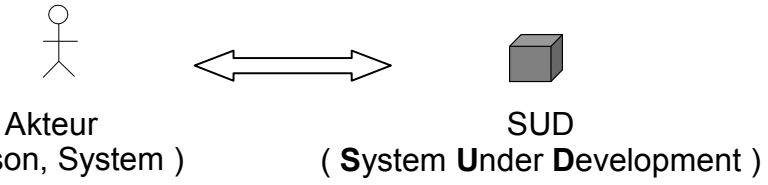
Typ eines Beispiel Use Cases

Use Case: **Essen bestellen**

1. Kunde bestellt Essen bei Zustelldienst
2. Zustelldienst bereitet Essen zu und liefert Bestellung an Kunden
3. Kunde übernimmt Essen und bezahlt Rechnung



- Business Use Case
- Unternehmen
- Black Blox
- kurze Fassung
- Benutzerebene



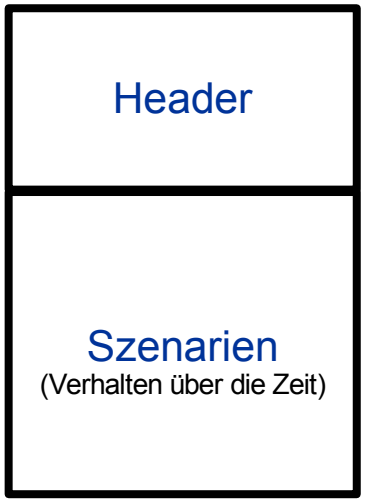
- System Use Case
- System
- Black Blox

Use Cases erstellen

Use Case Templates

Sind nicht genormt →
- beliebige Templates verwendbar (z.B. aus Prozessen)
- individuelle Erweiterungen

Beispiel für ein Use Case Template (vollständige Erfassung):



Beispiel für den Header eines Use Case Templates (vollständige Erfassung):

Feld	*Beschreibung*
Geltungsbereich	Unternehmen, Abteilung, System, Subsystem
Use Case Name	aussagekräftige Benennung
Primärer Akteur	möchte sein Ziel erfüllt haben
Vorbedingungen	werden nur vor Beginn der Ausführung geprüft
Nachbedingungen	sind nach erfolgreichem Abschluss erfüllt
Minimale Zusicherungen	sind nach jedem Abschluss erfüllt z.B. Logging
Erweiterungspunkte	optionale Erweiterung duch einen anderen UC
Inkludierungspunkte	Nutzung eines anderen UC
Stakeholder	Interessen müssen in jedem Schritt gewahrt werden
Offene Punkte	die noch zu klären sind
Nichtfunktionale Anforderungen	die sich nicht in einer ergänzenden Spezifikation finden (Perfomance, Business Rules, Lieferdatum)
...	...

Beispiel für Szenarien in einem Use Case Template (vollständige Erfassung):

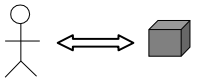
Basic-Flow:

1. Benutzer wählt Maschinenparameter aus, den er neu setzen möchte
2. System zeigt momentanen Wert des Parameters an, der editiert werden kann
3. Benutzer gibt neuen Wert für den Parameter ein

4'. Benutzer gibt Grund für die Änderung des Parameters ein

5. System überprüft Wert auf Gültigkeit
6. System übergibt neuen Wert an Maschine
7. Maschine meldet korrekte Übernahme des neuen Wertes
8. System zeigt Benutzer die korrekte Übernahme des Wertes an und aktualisiert die Anzeige des Parameters

- Ein möglicher Weg (keine Verzweigungen)
- Interaktion Akteur – System



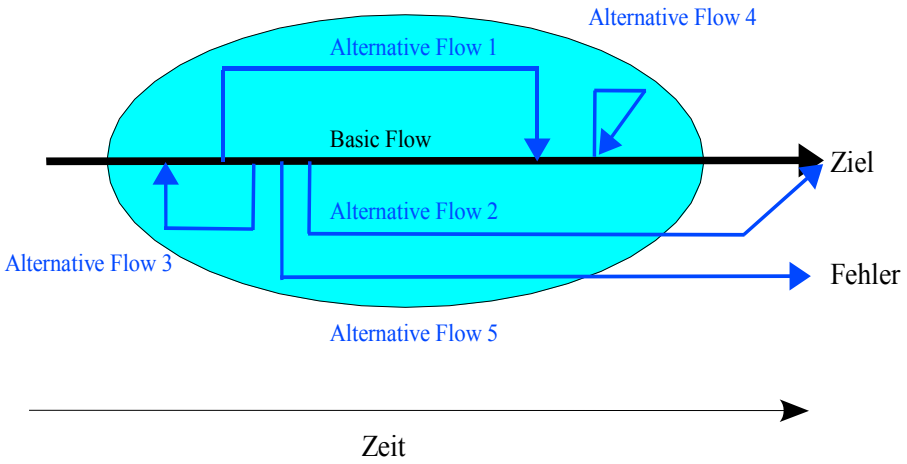
- Startet / endet mit stabilen Systemzustand
- Liefert ein messbares Ergebnis

Alternative-Flows:

5a. eingegebener Wert ist ungültig:
1. System teilt dem Benutzer mit, dass der eingegebene Wert ungültig ist
2. Benutzer bestätigt die Kenntnisnahme der Mitteilung
3. System setzt den Wert auf den vorherigen zurück

Szenario endet

5-6a. Maschine meldet Fehlverhalten
1. ?



Exceptions:

*a. Energieversorgung fällt aus:

7a. Maschine liefert keine Rückmeldung:
1. ....

- Fehler, die das System detektieren kann
- Gehören nicht zum regulären Programmablauf

Use Cases Formulieren (1/2)

- Gute Use Cases sind
  - **eindeutig**
  - **einfach**
  - **kurz**

- Schritt-Schema

  Wer macht **Was**?

Wert wird validiert	System validiert Wert
Messung starten wird ausgewählt	Benutzer startet Messung

- „Abbruch wählen“, „Beenden drücken“, … beschreiben GUI Design

  → beschreibt WIE

  → GUI Änderung bewirkt ungültigen Use Case

  } **keine GUI Beschreibung in Use Cases**

Use Cases Formulieren (2/2)

Was? sind die Ziele des Anwenders

~~**Wie?**~~

Der Anwender will …	sich am System anmelden
Der Anwender sagt …	„Benutzername eingeben“, „Passwort eingeben“, …
Der Systemanalyst formuliert …	System identifiziert und autorisiert Benutzer

„essentieller Stil“

verantwortlichkeits - bezogen

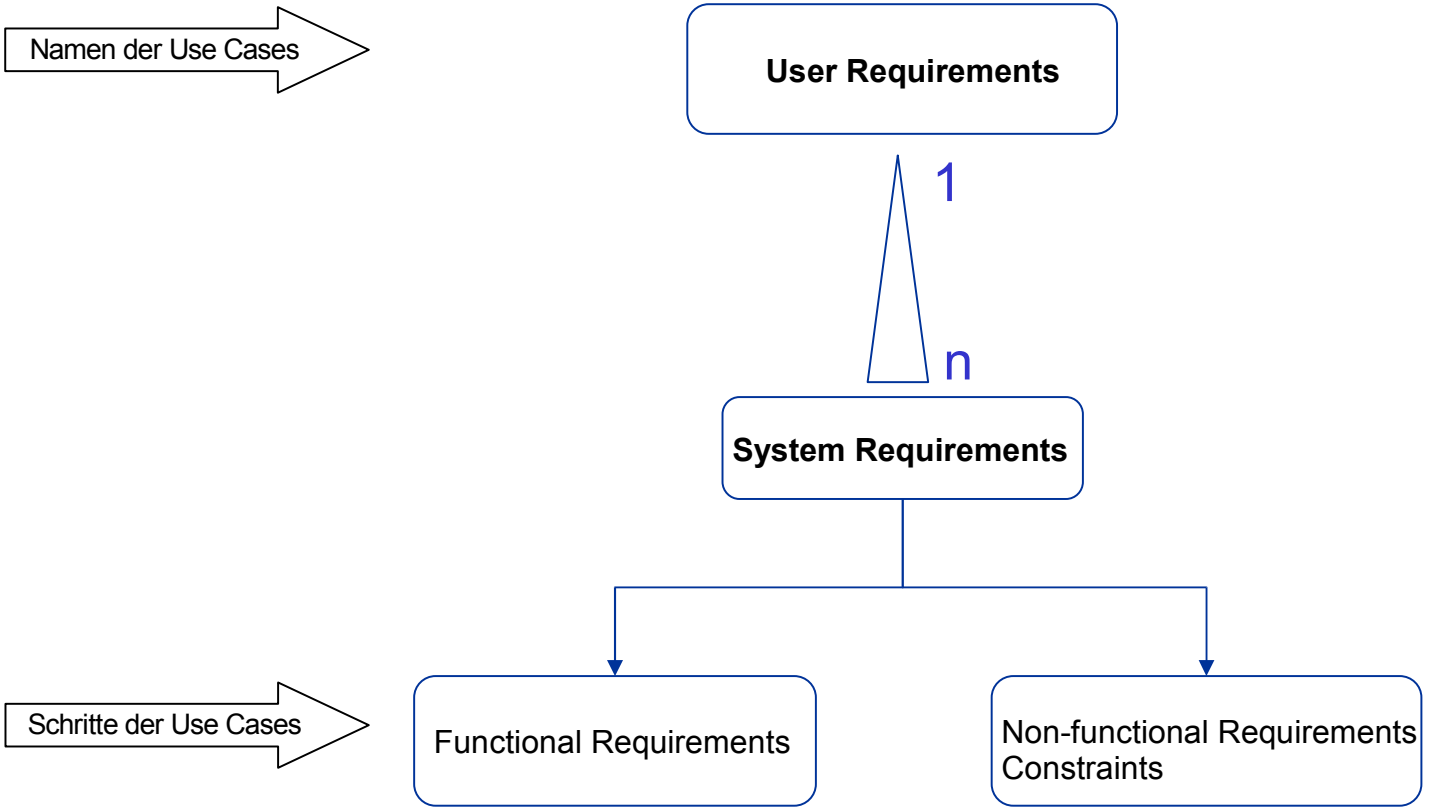
Wer?

mechanismus - unabhängig

Was?

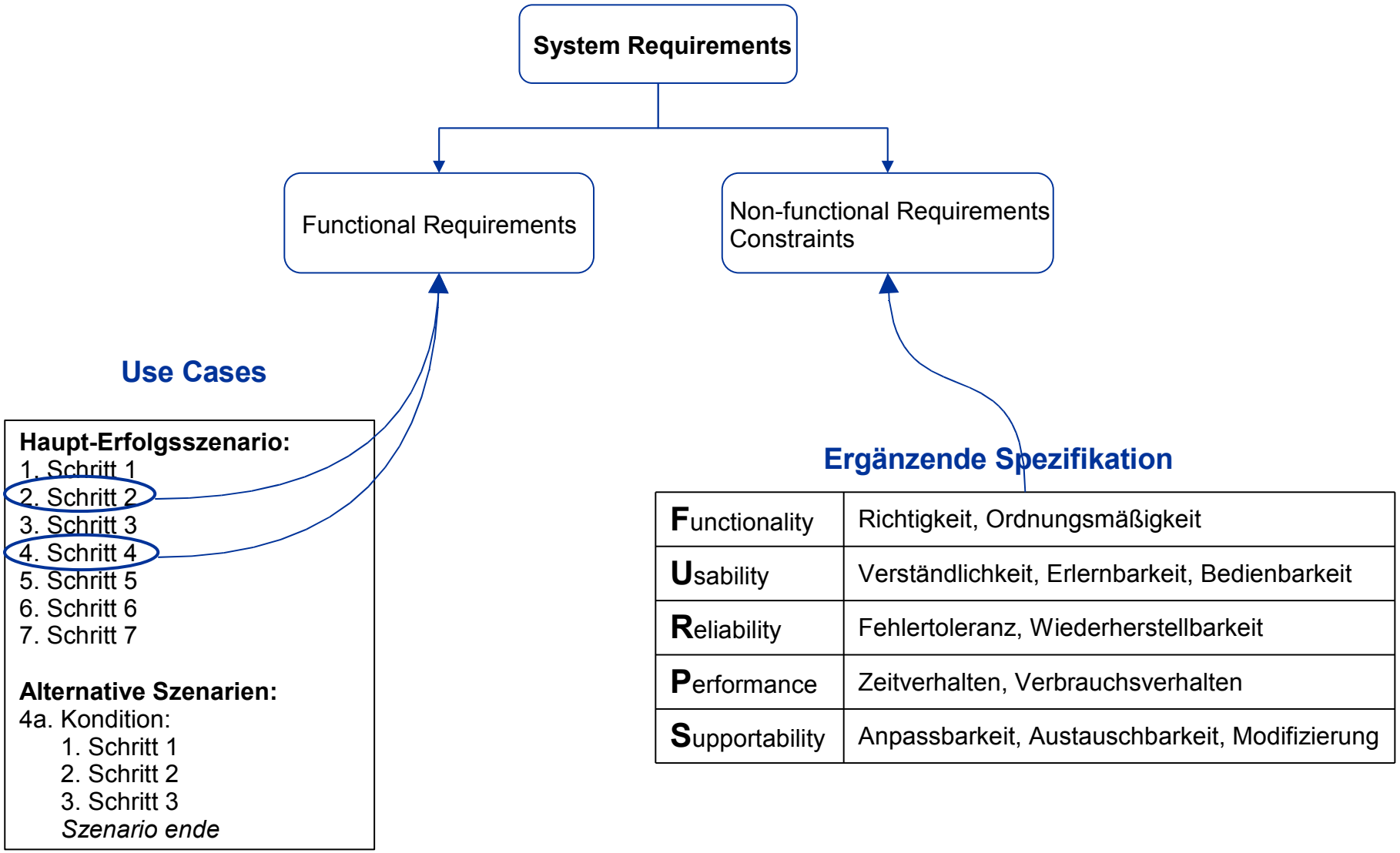
Use Cases und Requirements

User Requirements und System Requirements



Use Cases beinhalten	**alle Verhaltens-Requirements**	(funktionale)
und damit	**1/3 aller Requirements**	eines Systems

Use Cases und System Requirements



3. Fallbeispiel

zu

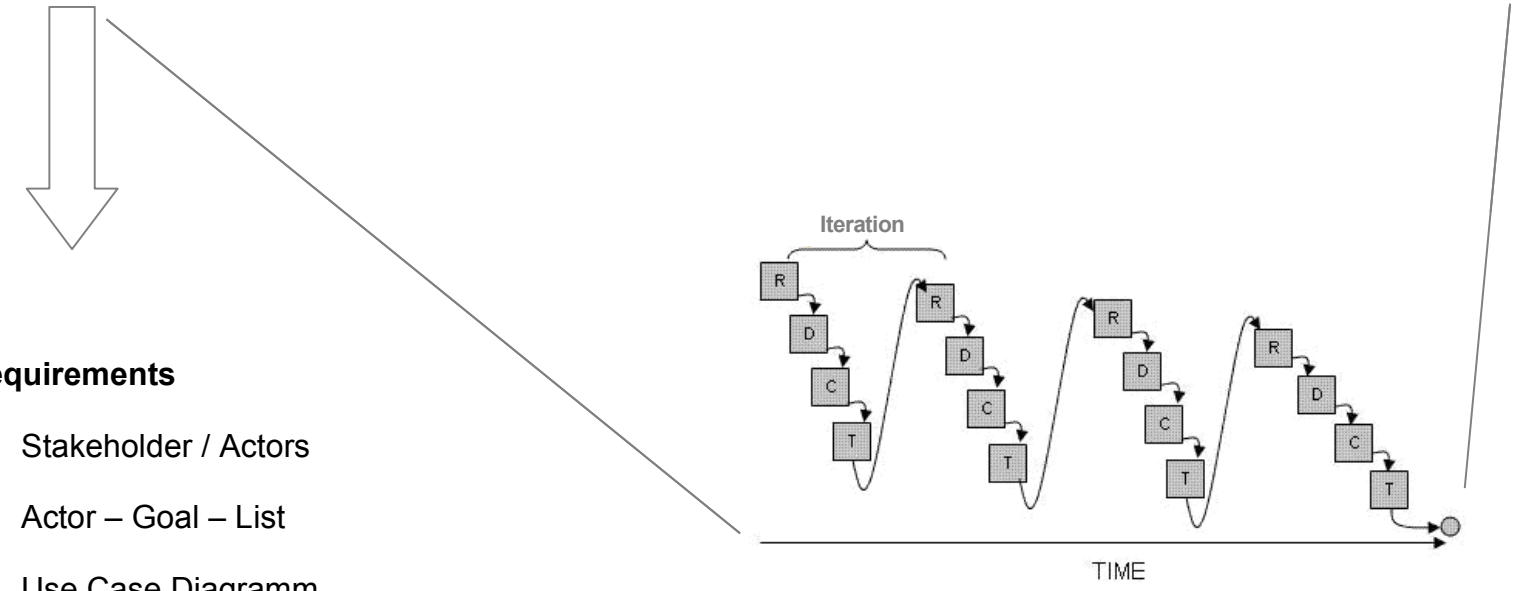
Requirements

Stakeholder / Actor	User Goals	System – Context	Requirements

OOA / OOD

OOA	OOD

Entwicklungszyklus



1) Requirements

- Stakeholder / Actors
- Actor – Goal – List
- Use Case Diagramm
- Use Case(s) (Funktionale Requirements)
- Ergänzende Spezifikation (Nichtfunktionale Requirements)

2) OOA / OOD

Beispiel: Transport – System

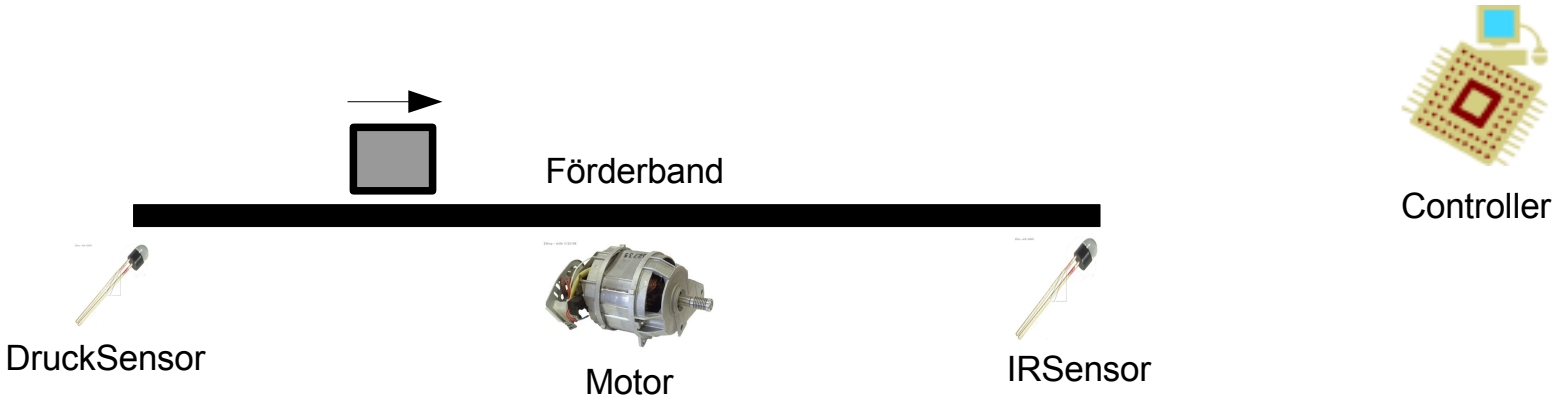
Das Förderband wird an dessen Anfang mit Transportgut beladen.

Das Gut soll bis ans Ende des Bandes transportiert werden, wo es von einem Greifer übernommen wird.

Wenn dieser nicht bereit ist, muss das Band gestoppt werden.

Ein Controller und mechanische Elemente sind bereits vorhanden.

Die Software für den Controller soll obigen Vorgang automatisieren, der bis jetzt von Hand ausgeführt wurde.



Iteration 1

Requirements

- Stakeholder / Actors
- Actor – Goal – List
- Use Case Diagramm
- Use Case(s) (Funktionale Requirements)
- Ergänzende Spezifikation (Nichtfunktionale Requirements)

Stakeholder / Actor

Wer ist am Erfolg des Projekts beteiligt?

Wer besitzt essentielle Informationen?

Wessen Interessen müssen gewahrt werden?

User Requirements

Was sind die Interessen der Benutzer?

Use Case Diagramm

Use Cases

(1/2) Functional Requirements

Ergänzende Spezifikation

(2/2) Non-functional Requirements

Objektorientierte Analyse

Objektorientiertes Design

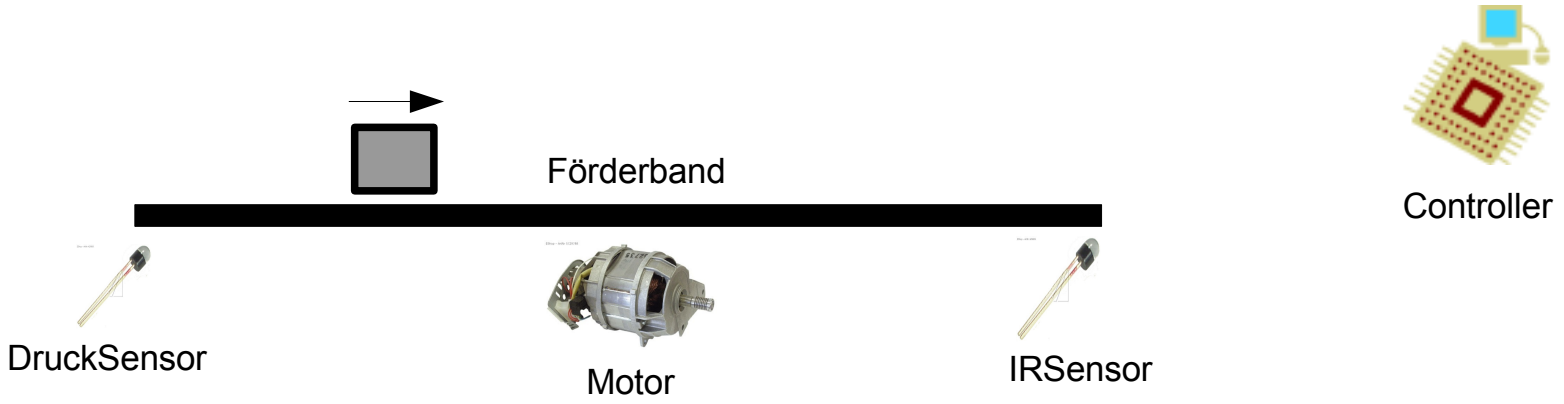
Objektorientierte Analyse

a) objektorientierte Abbildung des Systems

b) Beschreibung der Vorgänge im System

a) objektorientierte Abbildung des Systems

➔ graphisches Wörterbuch der Domäne



b) Beschreibung der Vorgänge im System

➔ Was tut das System?

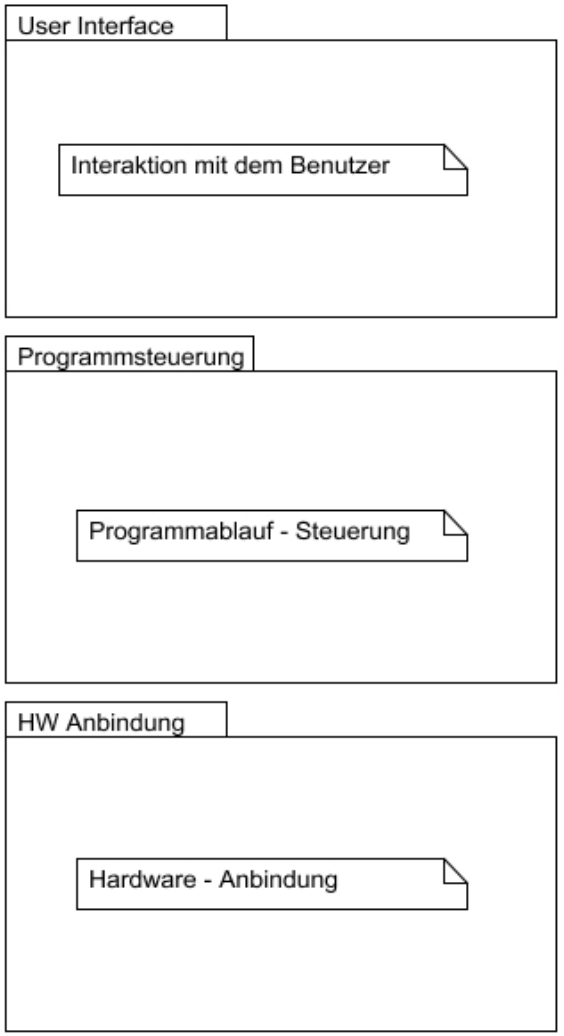
Objektorientiertes Design

a) Systemarchitektur

b) Klassendesign

a) Systemarchitektur

➔ Layer - Architektur



Ziele

- logische Strukturierung des Gesamtsystems (Layer)
- Aufgaben der Layer
- Klassen zur Erfüllung der Aufgaben

b) Klassen Design

"Gute" Klassen haben

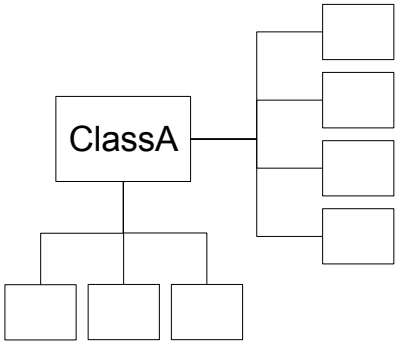
➔ größe Kohäsion

"A class should do one thing

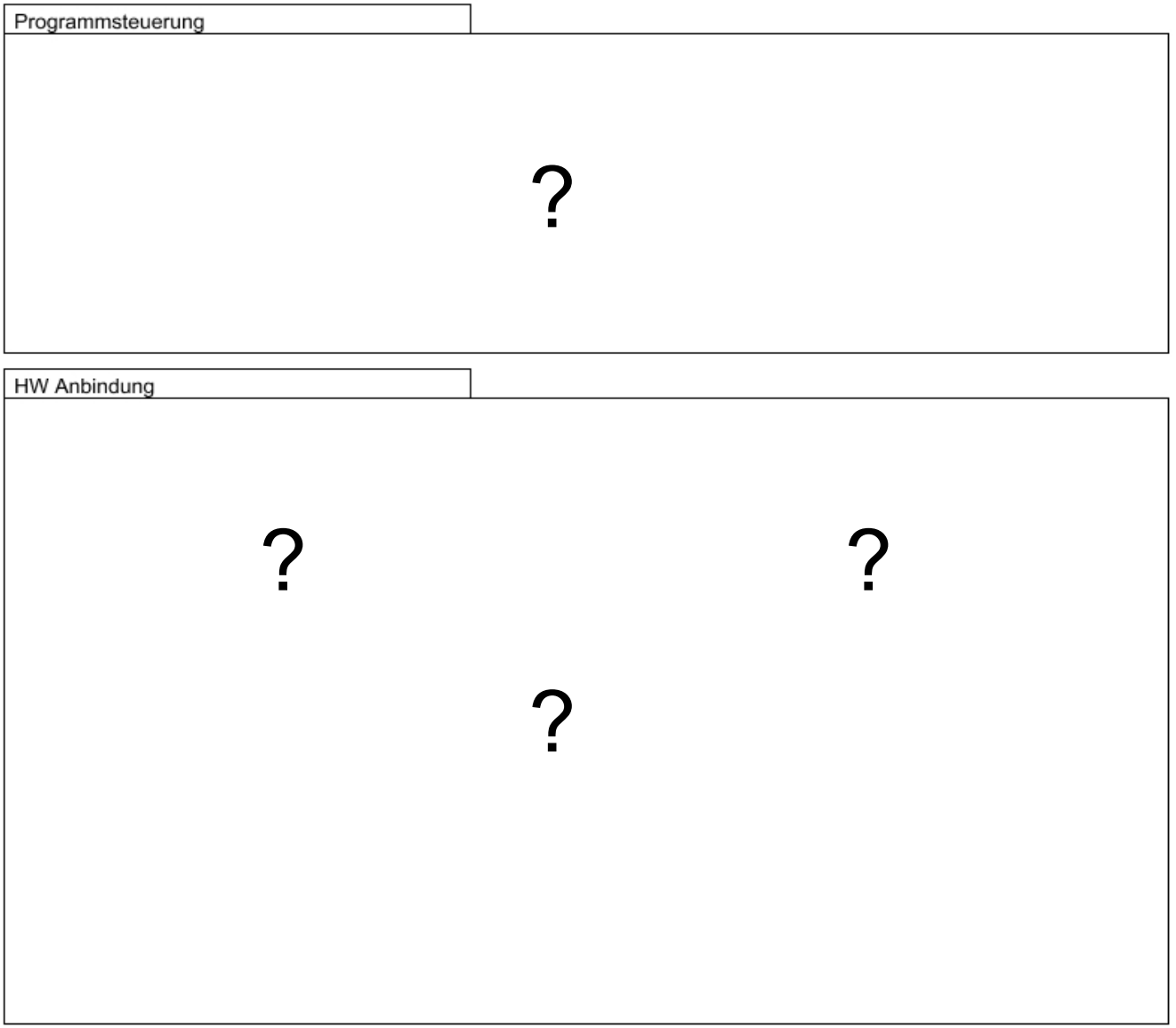
– and this should be done well!"

➔ geringe Abhängigkeiten

(vgl. Sensor – Steureung)



Analyse Klassen	≠	Design Klassen
1	:	n



Requirements	✔
Design	✔
Implementierung	✔
Test	✔

Iteration 1 Ende

Status Quo

(Teil der) Requirements: funktionale, nicht funktionale, SystemKontext, Akteure, ...

(Teil des) Design: Systemarchitektur → erweiterbar für fongende Use Cases zB GUI

Kernklassen → risikoreiche Kernfunktionalität

Implementierung: der Kernfunktionaltität

Test: risikoreicher Kern auf Realisierbarkeit getestet

Literatur

A. Kockburn, Writing effektive Use Cases

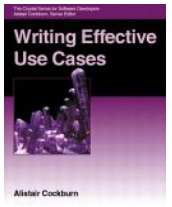
A. Kockburn, Agile Software Development

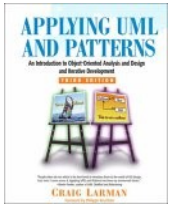
R. Wirfs-Brock, Designing object-oriented Software

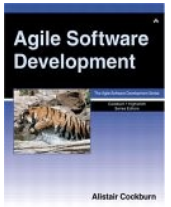
Craig Larmann, Applying UML and Patterns

Mario Jeckle, Chris Rupp, UML 2 glasklar

Gamma, Design Patterns







Links

http://www.iese.fhg.de/gi_fgrq/

http://www.uml.org

http://www.objectmentor.com/home

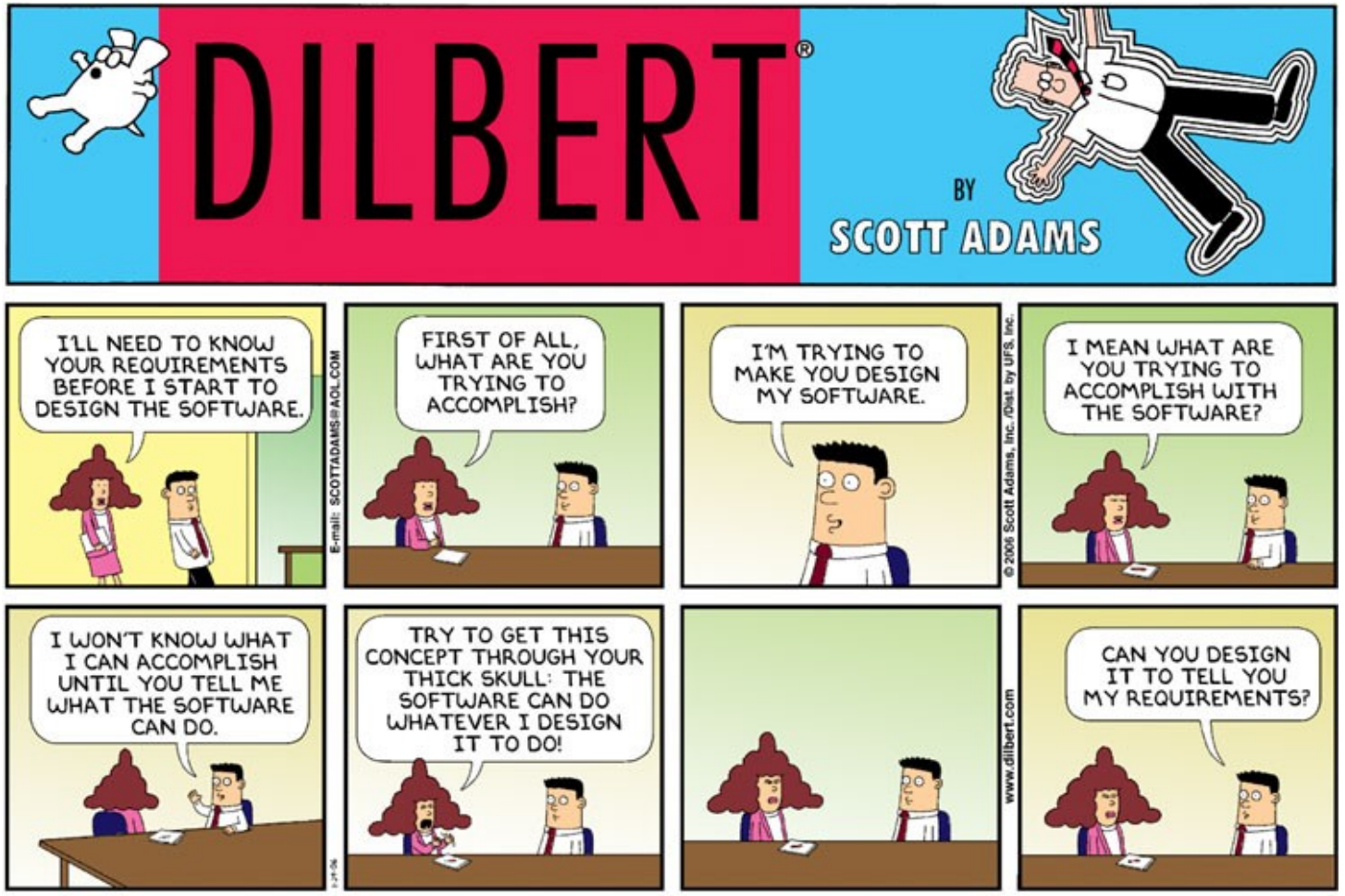
http://hillside.net/patterns/

Danke für die Aufmerksamkeit!

Fragen?

Feedback!

Diskussion



© Scott Adams, Inc./Dist. by UFS, Inc.

Kontakt:

andrea.asprian@mixed-mode.de

www.mixed-mode.de