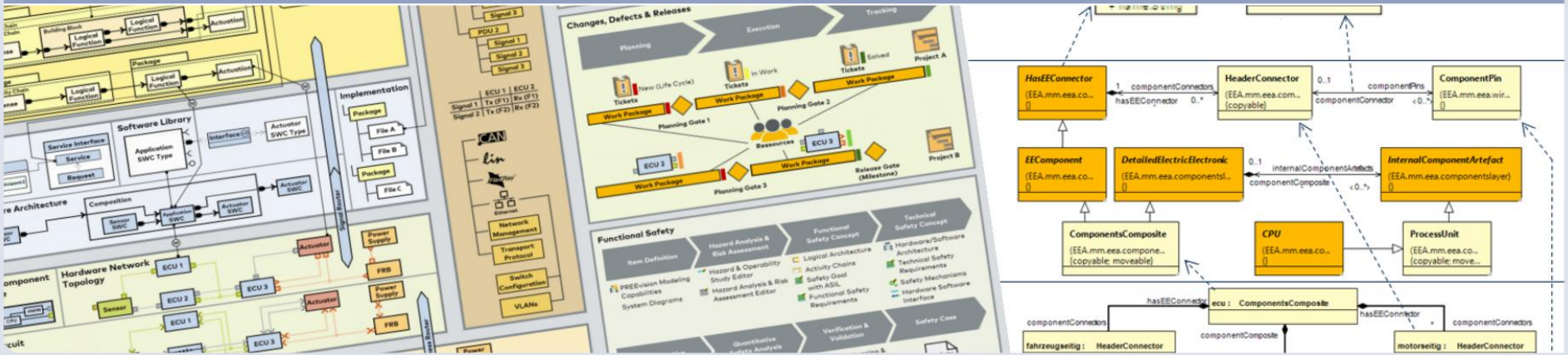
Karlsruher Institut für Technologie

Vorlesung Software Engineering (SE)

Wintersemester 2017/2018

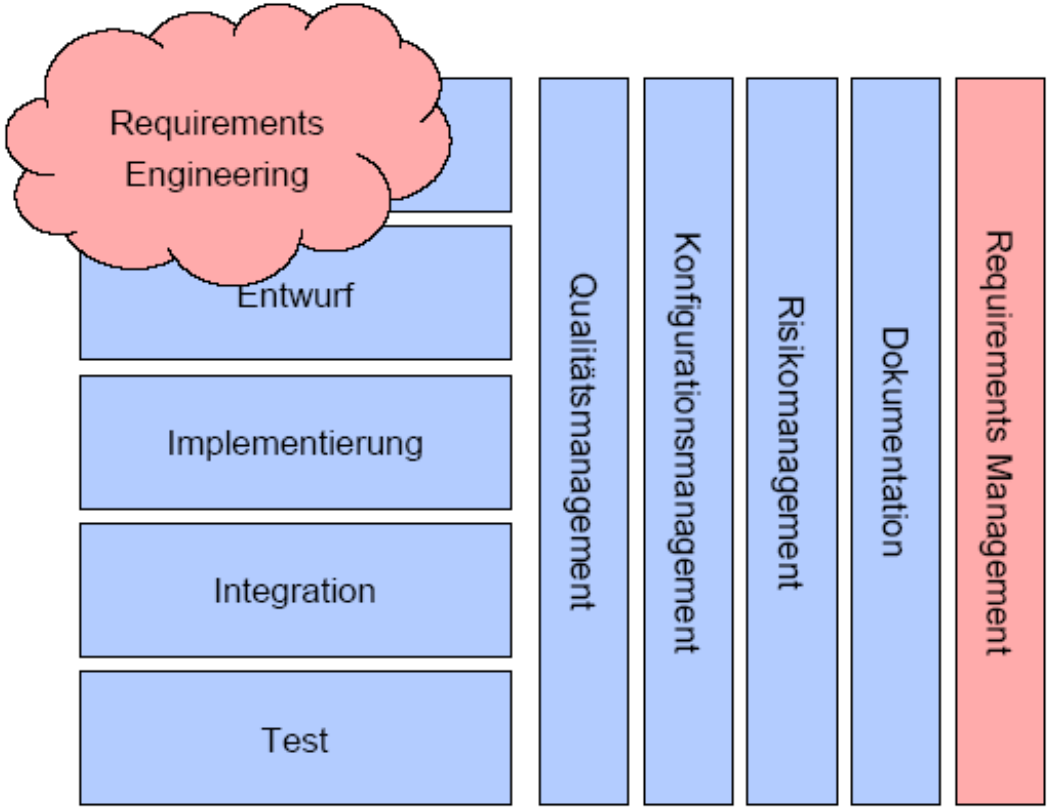
Kapitel 3 – Anforderungsmanagement



3. Anforderungsmanagement

Inhalt

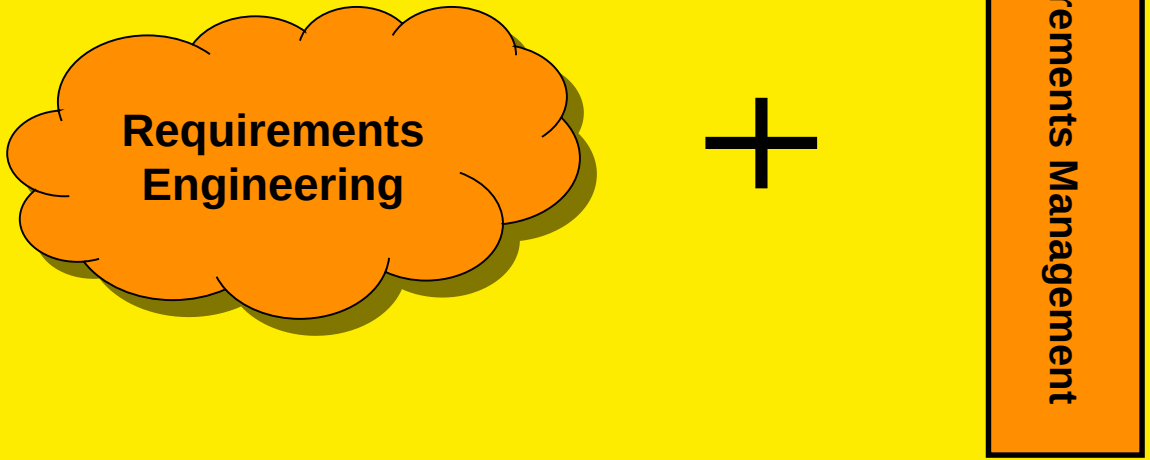
Requirements Engineering vs. Management



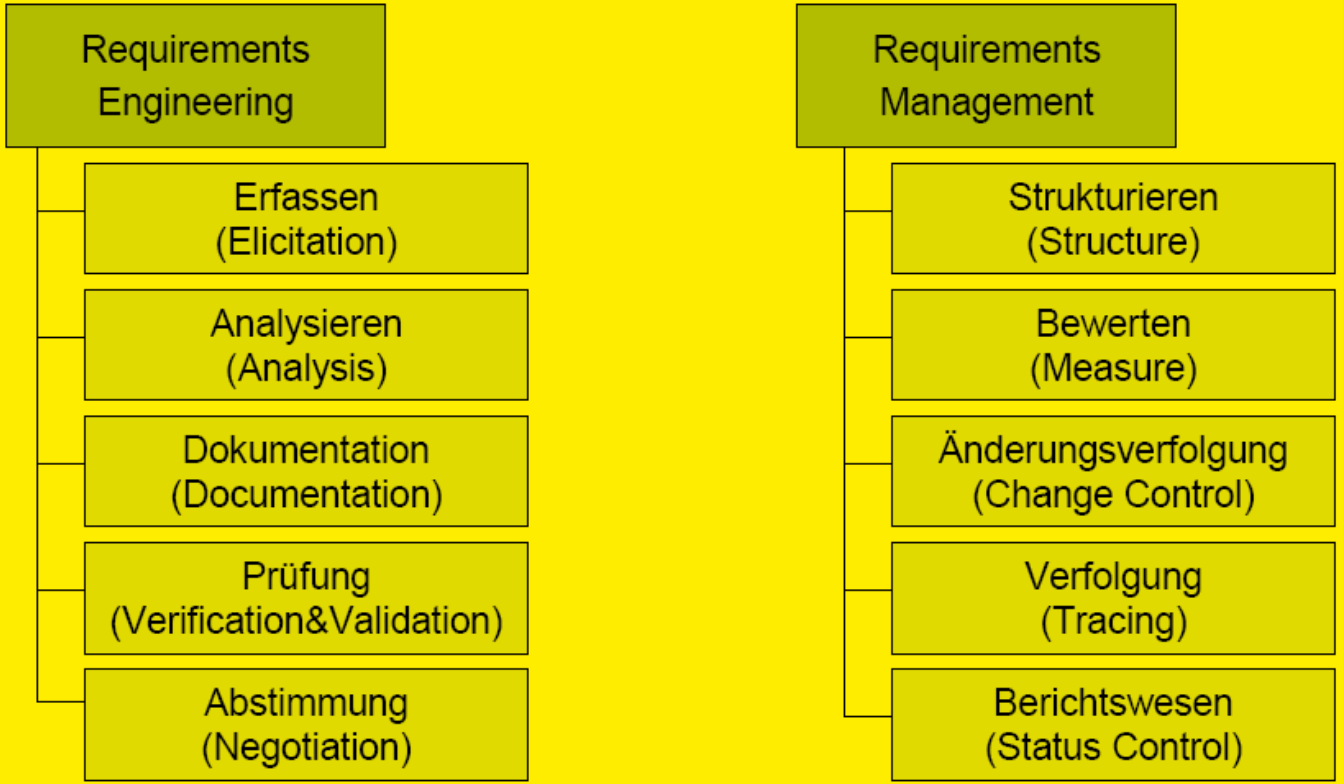
Als Oberbegriff

Anforderungsmanagement als Oberbegriff (Schienmann) für

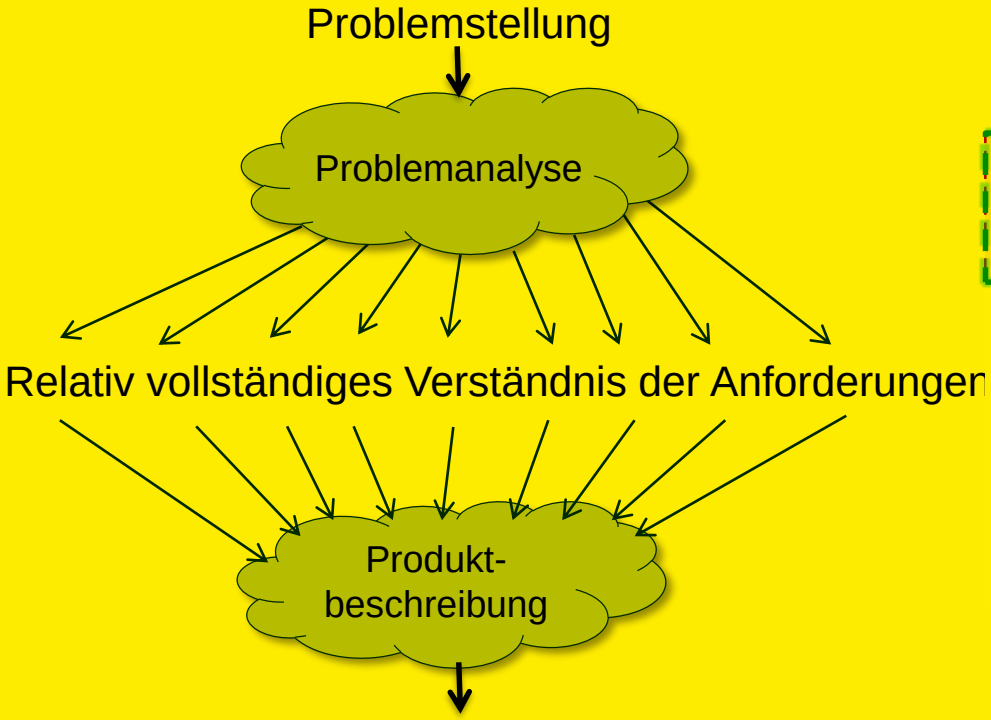
- Requirements Engineering (**RE**)
- Requirements Management (**RM**)

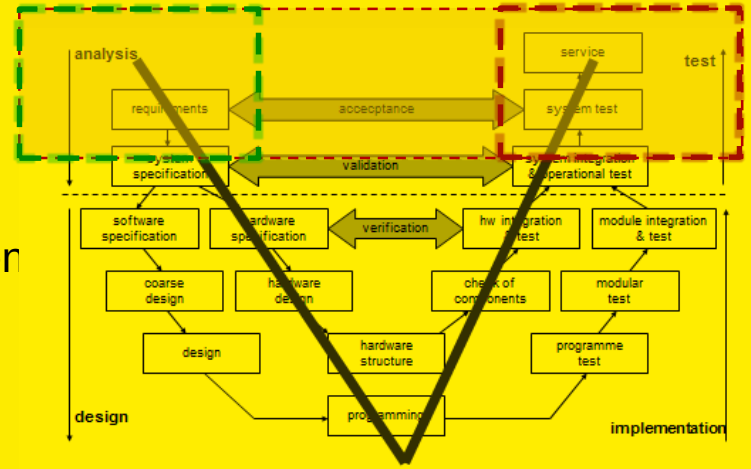


Wesentliche Bestandteile



Umfeld

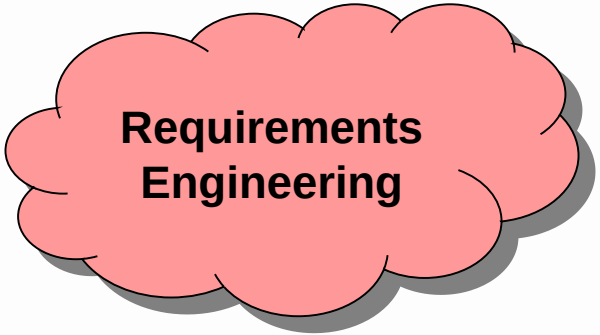




Konsistente und vollständige Anforderungsdefinition, aus der Testfälle für Abnahmetests abgeleitet werden können.

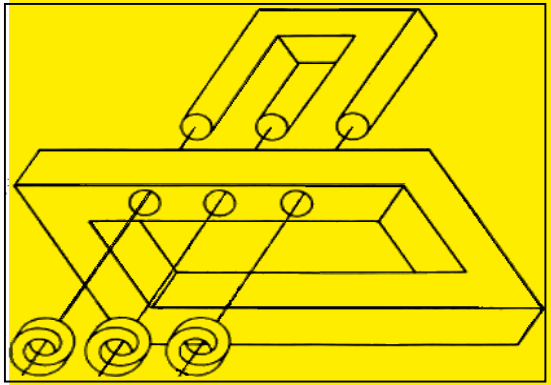
Anforderungsmanagement

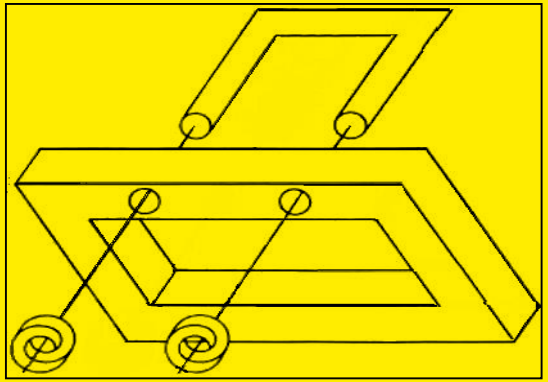
→ Requirements Engineering

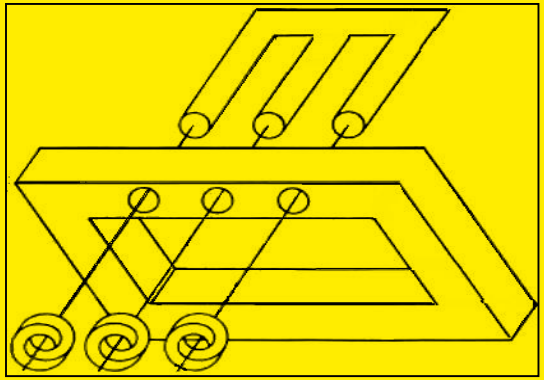


Begriff: Anforderung

Eine Beschreibung der Dienste und Einschränkungen des Systems sind ***Anforderungen***.







Qualitäts-Eigenschaften an Anforderungen (Characteristics)

EIGENSCHAFTEN

„Die Anforderungen an Anforderungen.“

- An einzelne Anforderungen (9)
- An Gruppen von Anforderungen (4)

(A) Notwendig (Necessary)

NOTWENDIG

Die Anforderung definiert eine essentielle Fähigkeit oder Eigenschaft:

Ohne diese wäre ein Mangel vorhanden, welcher nicht anderweitig gedeckt werden könnte.

- Die Anforderung ist aktuell anwendbar, und wurde noch nicht durch Weiterentwicklung obsolet.
- Besitzt die Anforderung eine geplante Lebensspanne, so ist deren Einsatz- und Auslaufdatum klar definiert.

(B) Unabhängig von Lösung (Implementation-Free) (I)

UNABHÄNGIG VON LÖSUNG

Die Anforderung bleibt *unabhängig* von einer Implementierung.

Sie deckt die notwendigen Aspekte ausreichend ab, ohne das Design in seinem Aufbau einzuschränken. (Etwa durch unnötige Konkretisierung der Architektur.)

Beschrieben werden soll das Ziel selbst (**was**), *nicht* dessen Weg (wie).

Sonst werden implizit Aussagen über Toleranzen oder Einzelheiten im konzeptuellen System getroffen, was zu *Einschränkungen* führen kann, welche allgemein nicht Teil der Anforderung wären.

(B) Unabhängig von Lösung (Implementation-Free) (II)

UNABHÄNGIG VON LÖSUNG

Anforderungen beinhalten also „was“-Informationen. [Problemraum]
Dennoch sind aber auch „wie“-Informationen wichtig. [Lösungsraum]

Design-Lösungen (direkt) in der Anforderung anzugeben, kann aber das Entstehungspotential neuer Ansätze gefährden.

- Bsp: Erwähnung konkreter kommerzieller Systeme, oder allgemein Systeme die gekauft anstatt geschaffen werden können.

Daher sollten sie *an anderer Stelle* dokumentiert und in anderer Form kommuniziert werden. *(vgl. Kapitel 5: Software-Entwurf)*

- Bsp: Anforderungs-Attribute können bei Design und Implementierung helfen.

(C) Unmissverständlich (Unambiguous)

UNMISSVERSTÄNDLICH

Die Anforderung ist *präzise* formuliert.

Ihre Absicht ist klar und leicht nachvollziehbar. Sie ist *eindeutig* und lässt keinen Interpretationsspielraum.

Mehrdeutigkeit kann durch Definition von Fachausdrücken und deren Verwendung vermieden werden.

Bei Abkürzungen empfiehlt es sich, dessen Bedeutung zunächst auszuschreiben.

(C) Unmissverständlich (Unambiguous)

UNMISSVERSTÄNDLICH

Terminologie

muss	must	Constraint by law
soll	shall	Necessary requirement
sollte	should	Recommended requirement
wird	will	Future requirement

Beispiel

»Die Soft[illegible]re **hat** Mittel zur Darstellu[illegible]xterner Dateien.«

unklar!

korrekt:

»Die Software **soll** über Mittel zur Darstellung externer Dateien verfügen.«

(D) Simpel (Concise)

SIMPEL (PRÄGNANT)

Die Anforderung ist kurz und bündig.

Sie enthält keine überflüssigen Informationen, stilistische Ausschmückungen, oder vermeidbare Fachsprache.

(**KISS-Prinzip**)

(E) Vollständig (Complete)

VOLLSTÄNDIG

Die Anforderung benötigt keine weiteren Zusätze, denn sie ist *messbar* und beschreibt *hinreichend* die Fähigkeiten und Eigenschaften, welche den **Bedürfnissen** des Stakeholders genügen.

(F) Atomar (Atomic)

ATOMAR (SINGULÄR)

Die Anforderung lässt sich nicht weiter aufteilen.

Ihre Beschreibung beinhaltet nur eine Anforderung, **ohne Konjunktionen** *(und, aber, …)* zu nutzen.

Beispiel

Anf1: »Das System soll die Möglichkeit bieten, einen Flug zu Buchen, ein Ticket zu kaufen, **und** einen Hotelraum zu reservieren.«

schlecht verfolgbar

Anf1.1: »Das System soll die Möglichkeit bieten, einen Flug zu Buchen.«

Anf1.2: »Das System soll die Möglichkeit bieten, ein Ticket zu kaufen.«

Anf1.3: »Das System soll die Möglichkeit bieten, einen Hotelraum zu reservieren.«

separat

(G) Machbar (Feasible)

MACHBAR (REALISTISCH)

Die Anforderung ist technisch *umsetzbar*. Sie fordert keine größeren technologischen Fortschritte, und berücksichtigt die **Systemeinschränkungen** mit akzeptablem Risiko.

(Dies beinhaltet Kosten, Zeitplan, Technik, Vorschriften …)

(H) Rückverfolgbar (Traceable)

RÜCKVERFOLGBAR

Die Anforderung ist *in beide Richtungen* rückverfolgbar.

Vorwärts: Wurde Sie vollständig erfüllt?
Dafür müssen untergeordnete Anforderungen auf tieferer Ebene einsehbar sein.

Rückwärts: Warum wurde sie erstellt?
Es existiert Dokumentation zu ihrem *Ursprung* (Stakeholder, Bedarfs-Erklärung, Anforderung höheren Ranges, oder vergleichbare Quellen).

Kurz: Für sie können alle Eltern-Kind Beziehungen bei Rückverfolgung identifiziert werden, sodass die Anforderung sowohl zu Quelle als auch zur Implementierung führt.

(I) Verifizierbar (Verfifiable)

VERIFIZIERBAR

Die Anforderung bietet Möglichkeiten um zu prüfen, ob das System die genannten Spezifikationen erfüllt.

Es können Belege gesammelt werden, die beweisen, dass das System die Anforderung erfüllt.

Verifizierbarkeit wird zudem von **Messbarkeit** gesteigert.

[SET] (I) Vollständig (Complete) (I)

VOLLSTÄNDIG

Eine Anforderungsmenge erfordert keine zusätzlichen Details; sie beinhaltet alles, welches zur Definition des zu spezifizierenden Systems erforderlich ist.

Es erhält keine Klauseln folgender Form:

TBD (To be Defined) **TBS** (To be specified) **TBR** (To be resolved)

- Auflösung der ‚TBx-Bezeichner' kann iterativ erfolgen.
  Ihre Items beinhalten ein akzeptables Zeitfenster, welches von Risiken und Abhängigkeiten bestimmt wurde.

[SET] (I) Vollständig (Complete) (II)

VOLLSTÄNDIG

Eine Anforderungsmenge erfordert keine zusätzlichen Details; sie beinhaltet alles, welches zur Definition des zu spezifizierenden Systems erforderlich ist.

Best Practices zur Verbesserung der Vollständigkeit:

- ✓ *Sämtliche* Anforderungstypen miteinschließen
- ✓ *Sämtliche* Stadien des Lebenszyklus bei jeder Anforderung beachten
- ✓ *Sämtliche* Stakeholder bei Aktivitäten zur Anforderungserhebung teilhaben lassen

[SET] (II) Konsistent (Consistent)

KONSISTENT

Zwischen Anforderungen bestehen keine Konflikte.

- Keine Widersprüche: Das Einhalten einer Anforderung darf nicht durch das Einhalten anderer unmöglich werden.
  - Bsp: verschiedene Vorgaben für Datumsangaben.
- Begriffe werden überall einheitlich verwendet. Synonyme werden vermieden.
- Es existieren keine Duplikate.

[SET] (III) Erschwinglich (Affordable)

ERSCHWINGLICH

Die Anforderungsmenge ist erfüllbar mit einer Lösung, die machbar ist innerhalb der Einschränkungen seines Lebenszyklus (etwa Kosten, Zeitplan, Technik, Vorschriften, …)

[SET] (IV) Begrenzt (Bounded)

BEGRENZT (STABIL)

Die Anforderungsmenge soll den identifizierten Gültigkeitsbereich beibehalten, der für die beabsichtigte Lösung im Rahmen des Notwendigen festgelegt wurde.

Ziel ist die Vermeidung nachträglicher Änderungen und Erweiterung der Anforderungen im Laufe des Entwicklungszyklus (‚Requirements-Creep‘):

Dies hat Einwirkungen auf Kosten, Zeitplan, und Qualität.

Von maßgebender Wichtigkeit sind hier sorgfältige Überprüfung der Anforderungsmenge und Rückverfolgbarkeit im Aufbau des Designs.

Fragen?

Anforderungen

Eigenschaften

Kategorisierung von Anforderungen

Typen

- Funktionale Anforderungen *(Functional Requirement)*
- Nichtfunktionale Anforderungen *(Non-Functional Requirement)*
- Problembereichs-Anforderungen *(Problem-Range-Requirement)*
  - ➢ Domäne, Fachlichkeit

Problem

künstliche Unterscheidung

➢ Typen nicht klar trennbar!
(nichtfunktionale Anforderungen können „funktionalisiert“ werden)

Funktionale Anforderungen

Funktionale Anforderungen

Aussagen über Dienste, die das System leisten soll

- Reaktion des Systems auf Eingaben
- Verhalten in bestimmten Situationen
- Was das System [nicht] tun soll

vgl. **EVA** -Prinzip (Datenverarbeitung)

Funktionale Anforderungen (II)

Funktionale *Benutzer*anforderung

- Häufig sehr allgemein

Funktionale *System*anforderung

- Detaillierte Spezifikation
- Häufige Fehlerquelle:
  Mehrdeutigkeit führt zu Implementierung, die Kunde nicht will
- Ausführbare Spezifikation

  z.B. MATLAB/Simulink:
  Blockdiagramme formalisieren grafisch die Anforderung

Nichtfunktionale Anforderungen (Non-functional requirement, NFR)

Nichtfunktionale Anforderungen

Qualitätseigenschaften und Randbedingungen

Beschränkungen

- Sollte qualitativ sein (Metriken)
- Sollte testbar sein (oft schwer überprüfbar)

Globale Sichtweise

Wichtiger als funktionale Anforderungen,
da bei Nichteinhaltung System unbrauchbar ist

Nichtfunktionale Anforderungen (Non-functional requirement, NFR)

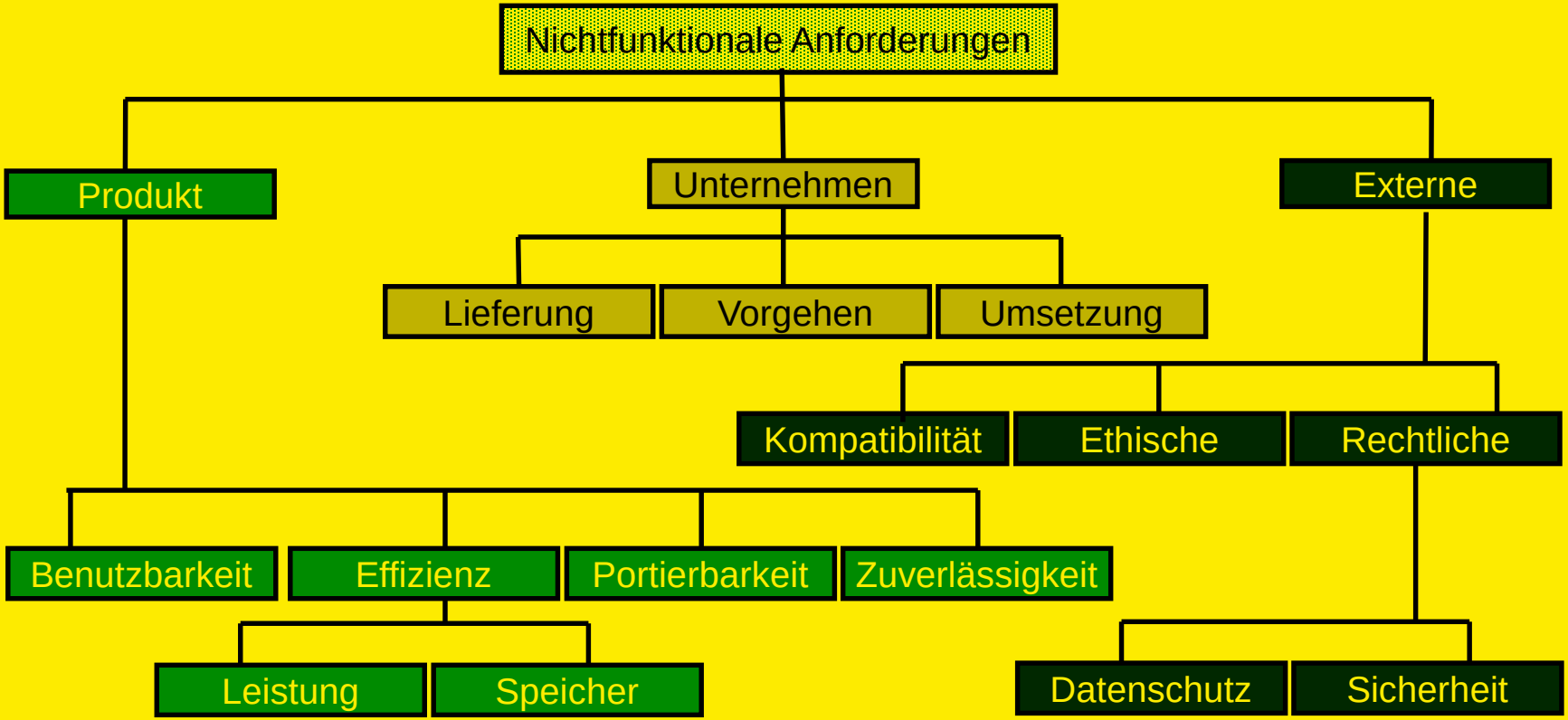
Nichtfunktionale Anforderungen

Qualitätseigenschaften und Randbedingungen

Beispiele

- Zeitbeschränkungen
- Entwicklungsprozess
- einzuhaltende Standards
- Zuverlässigkeit
- Reaktionszeit
- Speicherbedarf
- Leistungsfähigkeit von E/A Geräten
- Datendarstellung an Systemschnittstellen

Nichtfunktionale Anforderungen – Arten



Problembereichsanforderungen

Problembereichsanforderungen

- Problembereich des Systems, gibt Charakteristik wieder
- Sind funktionale/nichtfunktionale Anforderungen

Einschränkung einer funktionalen Anforderung

Beispiel (Bibliotheksystem)

1. Es sollte eine Standardbenutzerschnittstelle zu allen Datenbanken geben, die auf dem Z39.50 Standard basiert
2. Aus urheberrechtlichen Gründen müssen einige Dokumente direkt bei der Ankunft gelöscht werden. Abhängig von den Anforderungen des Benutzers sollen diese Dokumente entweder lokal auf dem Systemserver ausgedruckt, um manuell vom Benutzer verschickt zu werden, oder sie sollen an einen Netzwerkdrucker weitergeleitet werden.

Anforderungen: Beschreibungsformen

Natürliche Sprache

- Mangel an Genauigkeit
- Verwirrung bei Anforderungen (Typen nicht unterscheidbar)
- Verschmelzung von Anforderungen
- Gleicher Sachverhalt kann unterschiedlich ausgedrückt werden (überflexibel)
- Nicht einfach modularisierbar
  - ➢ Gruppierung

Sprachgebrauch

- Verbindliche Anforderungen: soll (shall)
- Wünschenswerte Anforderungen: sollte (should)
- Rechtliche/Ethnische Vorgaben: muss (must)
- Zukunftsvisionen: wird (may)

Verwendung von CASE Werkzeugen

Standardisierung des Formats (RIF, ReqIF)

Requirements Interchange Format

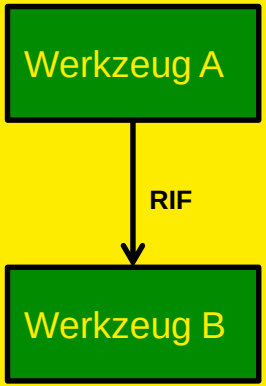
Strukturierung

Informationen in einer Anforderung

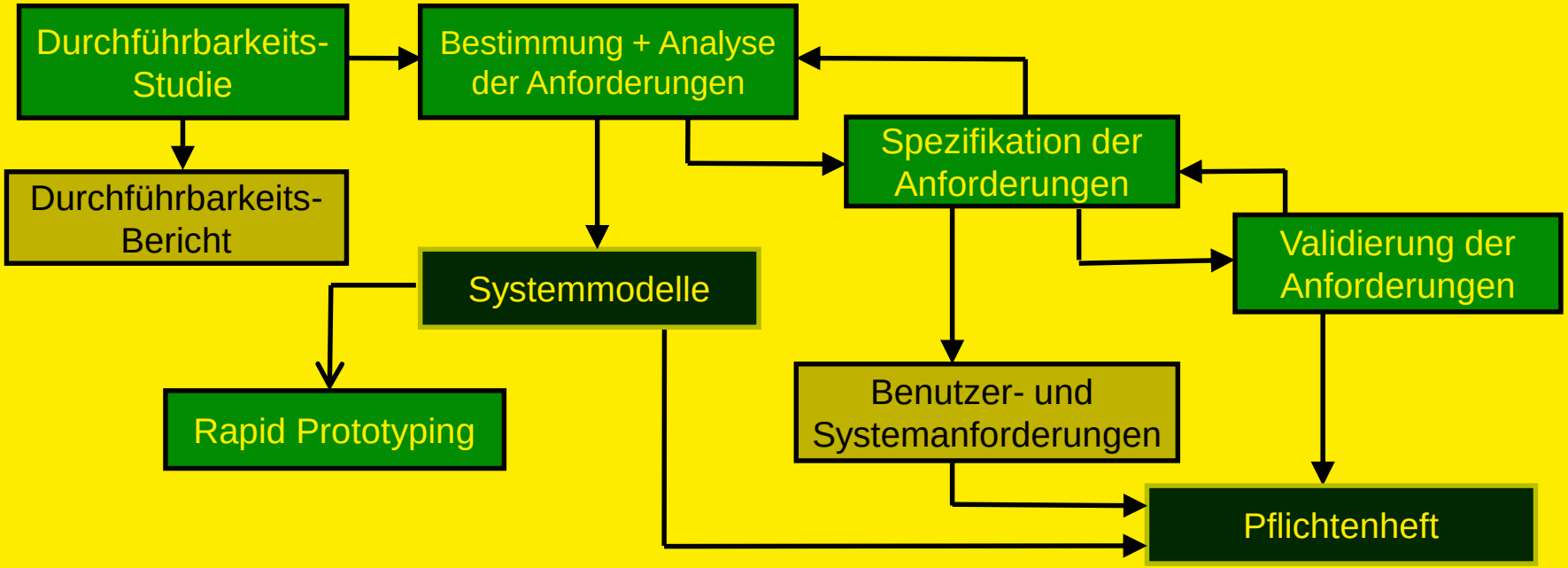
- Beschreibung mit Hervorhebungen (fett, kursiv)
- Attribute
- Beziehungen (benötigt)
- Versionierung, Benutzermanagement
- Eingaben
- Ausgaben
- Vorbedingung
- Nachbedingung
- Invarianten

Glossar

- Begriffsdefinition
  z.B. Hervorhebung durch Farbe



3.2.3 Prozess der Anforderungsanalyse



Das **Pflichtenheft** beschreibt in konkreterer Form,
wie der Auftragnehmer die Anforderungen im Lastenheft zu lösen gedenkt –
das sogenannte **wie** und **womit**.

Durchführbarkeitsstudie (Feasibility-Studie)

Durchführbarkeitsstudie

Prüfung

Bedürfnisse erfüllbar mit…

- Software Technologie?
- Hardware Technologie?

Kosten im Rahmen?

Ziel

Entscheidung, ob detaillierte Analyse sinnvoll ist

Analyse

Bestimmung und Analyse der Anforderungen

Ableiten aus…

- Beobachtung bestehender Systeme
- Diskussion mit potentiellen Benutzern und Käufern
- Aufgabenanalyse

Systemmodelle

CASE Tools

Prototypen

Tools für funktionale Prototypen

- MATLAB/Simulink, MATLAB/Stateflow (The Mathworks)
- Statemate/Rhapsody (IBM)
- ASCET (ETAS)
- …

Ziel: System verstehen

Analysetechniken

Viewpoint (VP)

- Blickwinkel auf das System
- Rollen der Benutzer

Szenarien

- UML **M**essage **S**equence **C**harts (MSC)
- Strukturierung in Anwendungsfälle (UML Use Case)

Ethnografie

- Gesellschaftliches Umfeld
- Wirtschaftliches Umfeld
- Eintauchen des Analytikers in Umfeld des zu erstellenden Systems

Analysetechniken: Viewpoint Attributes

Viewpoint Attribute		Attribute Description
Name		Identifies the viewpoint (VP)
Focus		Shows the definition of the perspective taken by the VP
Sources		Shows a list of soruces of the viewpoint‘s requirements
Actors		Shows a list of actors associated with the VP
NFRs		Shows a list of the NFRs applied to the VP
Use cases		Shows a list of functionalities relevant to the VP
Aggregation		Shows a list of aggregations where the VP is involved; illustrated in the viewpoint diagram and a specific template
Association		Shows a list of associations where the VP is involved; illustrated in the viewpoint diagram and a specific template
Gen/ Spec	**Sub VP**	Shows a list of sub VPs, illustrated in the viewpoint diagram
	Super VP	Shows a list of super VPs, illustrated in the viewpoint diagram
History		Keeps the alterations of the viewpoint information through time.

3.2.3.1 Spezifikation der Anforderungen (Requirements Document) (I)

Anforderungsspezifikation

Dokument, das aus Informationen der Analyse Anforderungen spezifiziert und strukturiert.

Spezifikation der Anforderungen (Requirements Document) (II)

Pflichtenheft

Produktdefinition, Produktspezifikation

- Offizielle Aufstellung: was wird vom SW-Entwickler erwartet?

Pflichtenheft enthält zwei Typen von Anforderungen

1. **Benutzeranforderungen** (BENUTZER / USER)
   - Abstrakte Beschreibung der Systemanforderungen
   - Kunde braucht grobe Aufstellung
2. **Systemanforderungen** (WAS / WHAT)
   - Detaillierte Auflistung der Funktionen
   - Systementwickler braucht detaillierte Systemspezifikation

Systemanforderungen

Systemanforderungen

- Anforderungen als Ganzes erkennen
- Beinhaltet Beratung mit dem Kunden und Benutzer
- Gesamtziele des Systems
- Abstrakte Funktionale Anforderungen
- Systemeigenschaften
  - Nicht funktionale, sichtbare Eigenschaften des Systems (Verfügbarkeit, Leistungsfähigkeit, Sicherheit, etc.)
  - Globale Auswirkungen
- Eigenschaften, die das System *nicht* aufweisen darf

Schwierigkeiten

- Anforderungen für komplexe Probleme lassen sich vor dem Auftreten des Problems nicht endgültig spezifizieren
- Implizite Anforderungen werden häufig bei der Spezifikation vergessen

Beispiel: Benutzer- und Systemanforderung

Benutzeranforderung (im Lastenheft)

1. Die Software **soll** über Mittel zur Darstellung externer Dateien verfügen, die von anderen Werkzeugen erzeugt wurden, und auf diese Dateien zugreifen können.

Systemanforderungen (im Pflichtenheft)

1.1 Der Benutzer **sollte** über Möglichkeiten zur Definition externer Dateitypen verfügen

1.2 Jeder externe Dateityp **kann** eine damit verknüpfte Anwendung besitzen, mit der die Datei bearbeitet wird

1.3 Jeder externe Dateityp **kann** als bestimmtes Symol auf dem Bildschirm des Anwenders dargestellt werden

1.4 Es **sollten** Möglichkeiten für die Definition des Symbols für externe Dateitypen durch den Benutzer bereitgestellt werden

1.5 Wenn ein Benutzer ein Symbol auswählt, das eine externe Datei repräsentiert, so **soll** die durch dieses Symbol dargestellte Datei mit der Anwendung geöffnet werden, die mit dem entsprechenden externen Dateityp verknüpft ist.

Struktur eines Pflichtenheftes nach IEEE 830-1993 (I)

Vorwort

- Leserschaft
- Versionsgeschichte
- Warum neue Version, Zusammenfassung der Veränderungen

Einleitung

- Notwendigkeit des Systems
- Zusammenfassung der Funktion
- Wie Zusammenarbeit mit Umwelt
- Zusammenhang zu wirtschaftlichen und strategischen Zielen des Unternehmens des Auftraggebers

Begriffe

Benutzeranforderungen

- Überblick über erwartete Systemarchitektur
- Verteilung der Funktionen
- Wiederverwendete Komponenten

Struktur eines Pflichtenheftes nach IEEE 830-1993 (II)

Systemarchitektur

Systemanforderungen

Systemmodelle

Systementwicklung

- Grundsätzliche Voraussetzungen
- Erwartete Veränderungen wegen…
  - HW Entwicklung
  - Benutzeranforderungs-Änderungen

Anhänge

- Minimale/maximale Hardwarekonfiguration
- Datenbank-Anforderungen

Index

3.2.3.2 Validierung

Validierung der Anforderungen

- Sind Anforderungen realistisch?
- Sind Anforderungen konsistent?
- Sind Anforderungen vollständig?

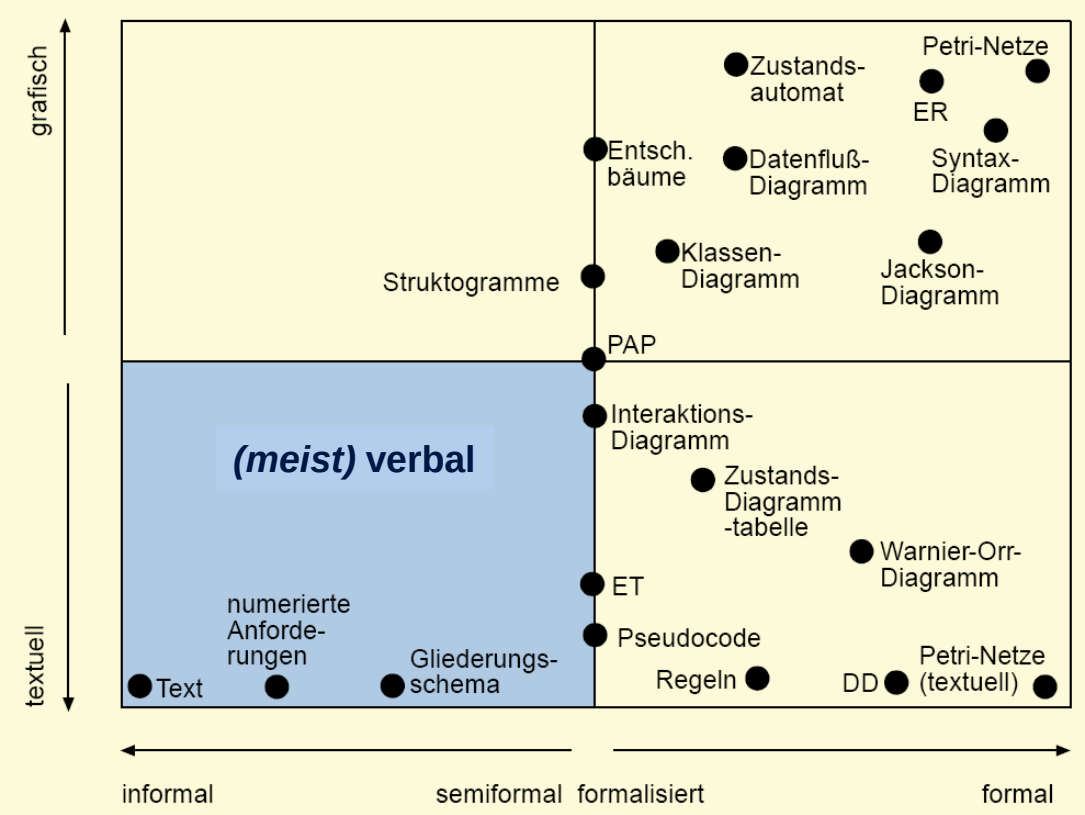
Ziel

Erkennen von Fehlern im Anforderungsdokument

Vorgehen

- Durchführung von Reviews
- Validierung mit Rapid Prototyping
  - Ausgangspunkt: Modelle
  - Automatische Codegenerierung
  - Rapidprotyping Hardware

Beschreibungsmittel



(siehe Vorlesung SSE)

3.2.3.3 Anforderungsmodelle und ihre Diagramme

Kontextmodell
(SART, UML Klassendiagramm,…)

Anwendungsfalldiagramm
(UML use case, …)

Verhaltensmodelle

- Datenfluss z.B. MATLAB/Simulink®, ASCET® (ETAS)
- Zustand z.B. MATLAB/Stateflow®, Statemate® (Telelogic / IBM)

Datenmodelle

Objektmodelle (UML z.B. ARTiSAN®, Enterprise Architect)

- Vererbung
- Objektaggregation
- Objektverhalten

Dynamische Diagrammarten (I)

Bei der Problemanalyse beschreibt man in einem ***Use-Case Diagramm*** (deutsch: *Anwendungsfalldiagramm*) die wichtigsten Leistungen des zu erstellenden Systems.

Darstellung

Für die Darstellung der Abläufe bei der Durchführung eines Use Case gibt es eine Reihe komplementärer Diagrammarten mit unterschiedlichen Darstellungsschwerpunkten…

Dynamische Diagrammarten (II)

Darstellungsdiagramme für den Systemablauf

Sequenzdiagramme

betonen die zeitliche Abfolge

Kollaborationsdiagramme

betonen die an Interaktionen beteiligten Objekte, ihr Zusammenspiel, und den Datenfluss

Aktivitätsdiagramme

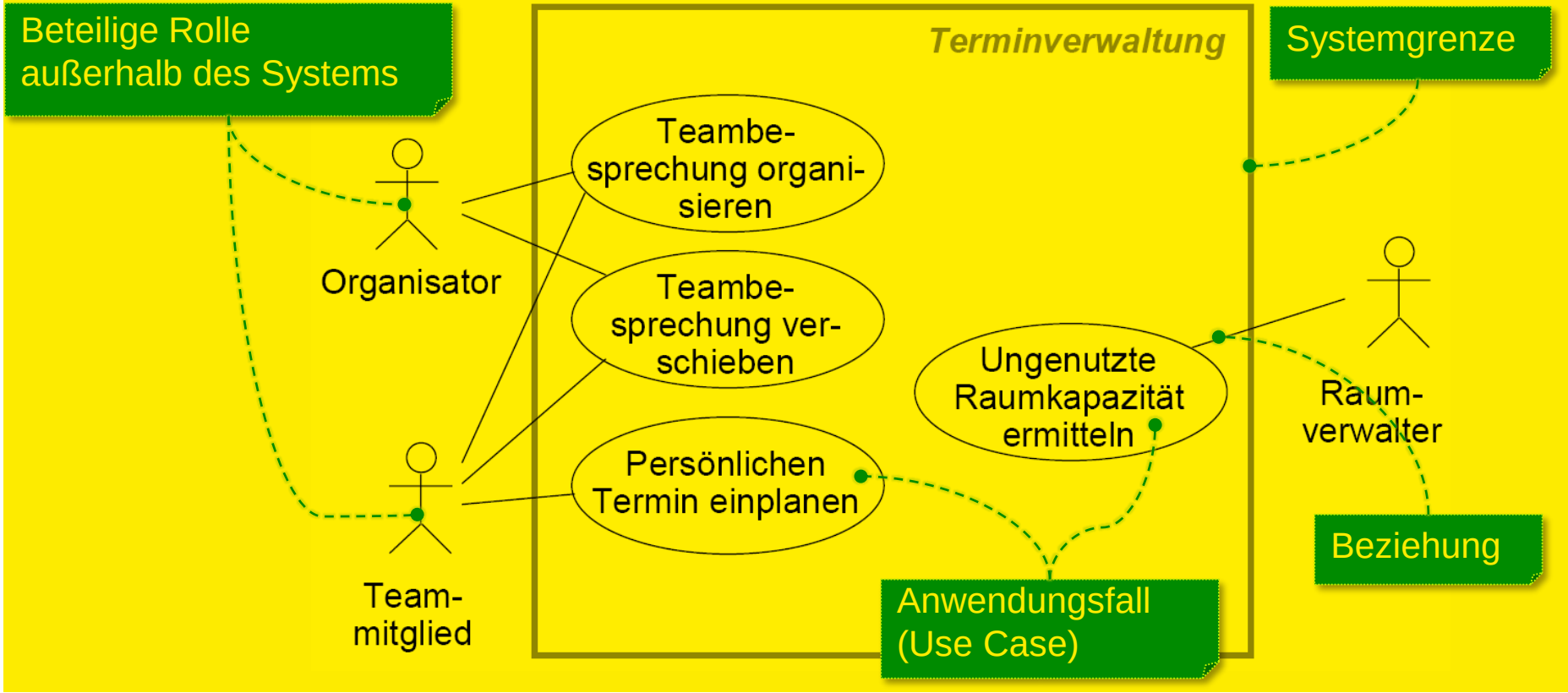
stellen Kontrollflüsse dar

Zustandsdiagramme

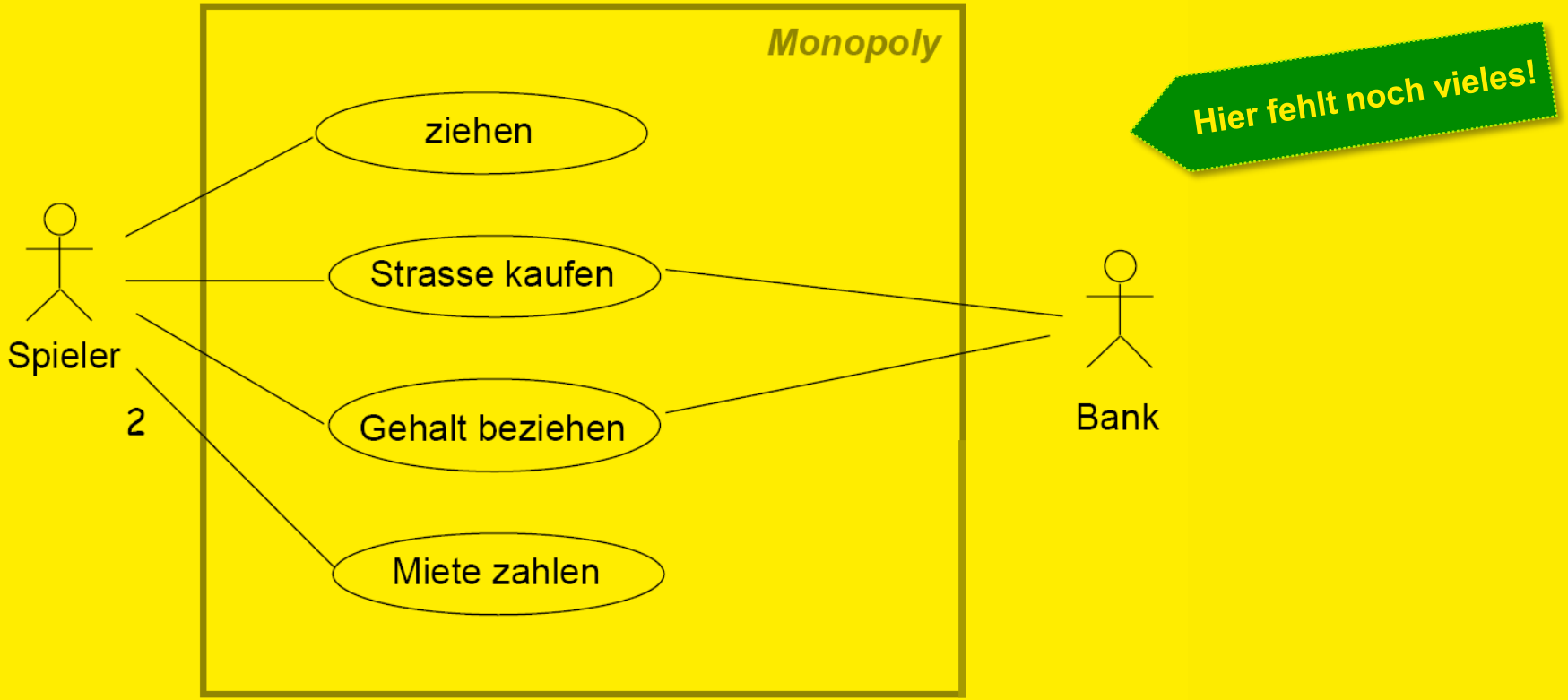
betonen unterschiedliche Verhaltensweisen in unterschiedlichen Zuständen im Objekt-Lebenszyklus

OCL (**O**bject **C**onstraint **L**anguage)
dient der Spezifikation von Zuständen & Operationen.

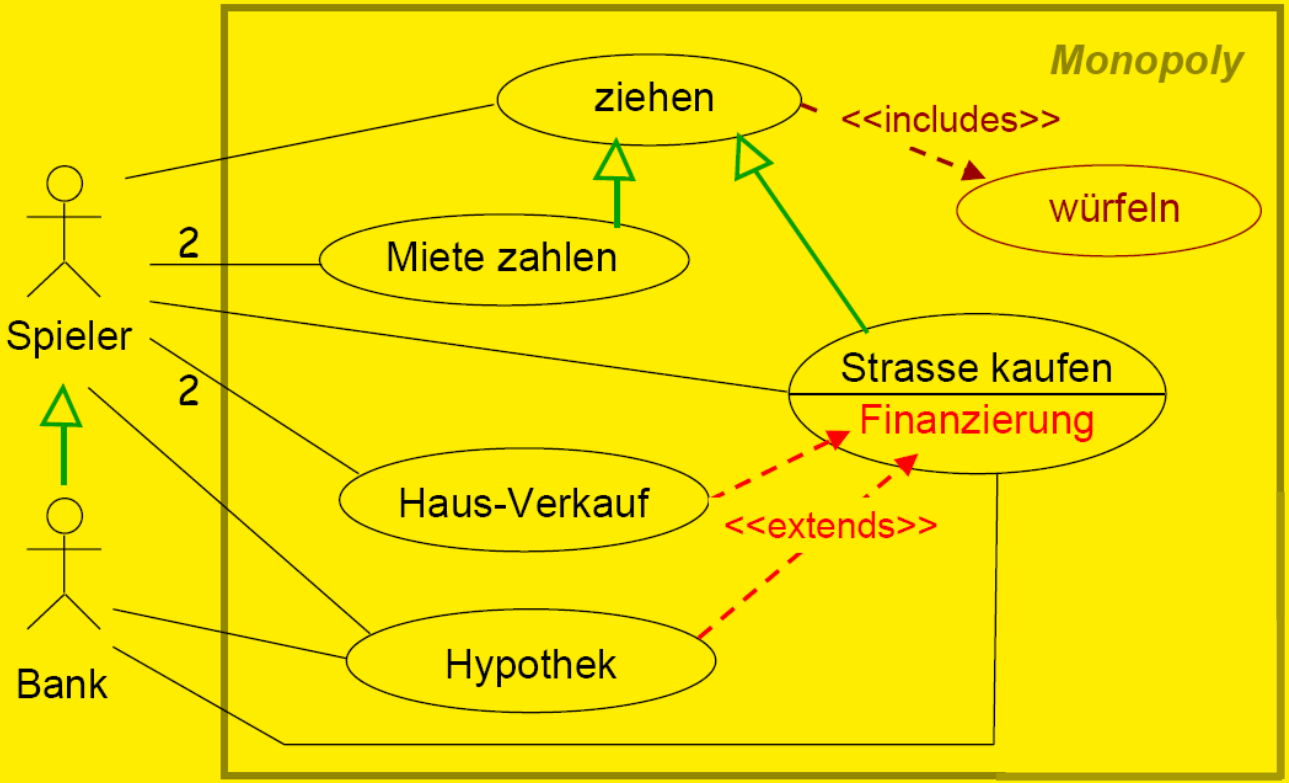
Use-Case-Diagramm: Terminverwaltung



Use-Case-Diagramm: Monopoly-Spiel

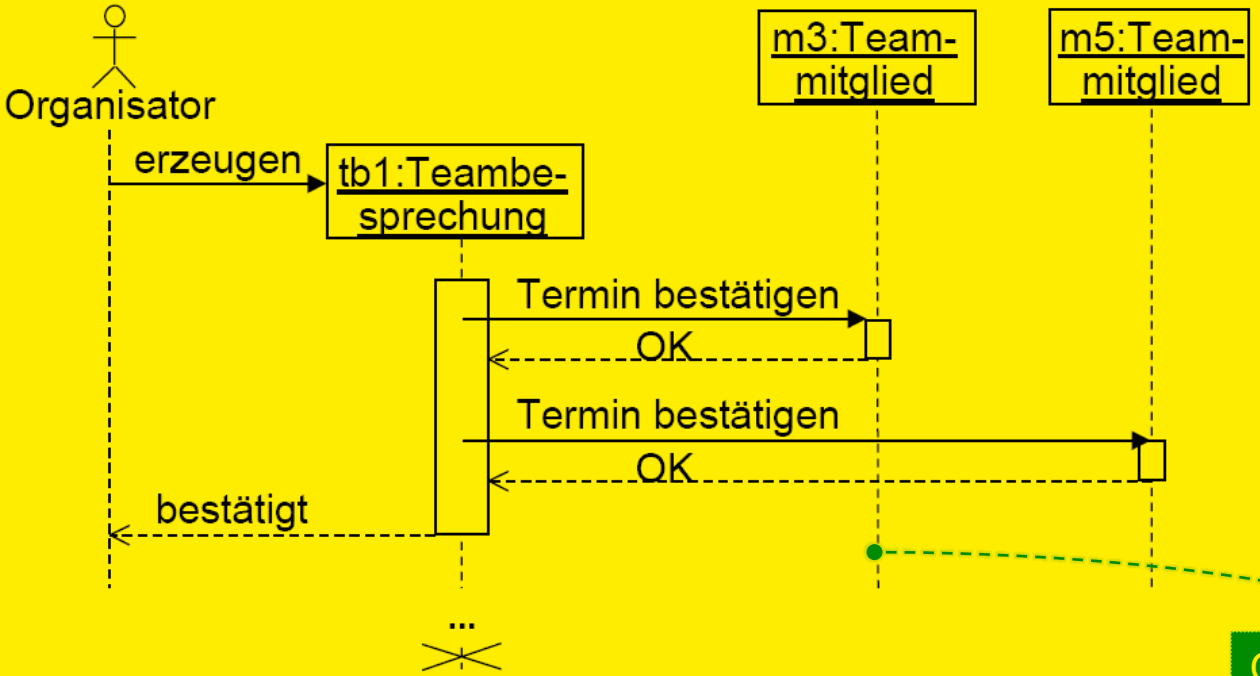


Use-Case-Diagramm: Monopoly-Spiel (etwas detaillierter)



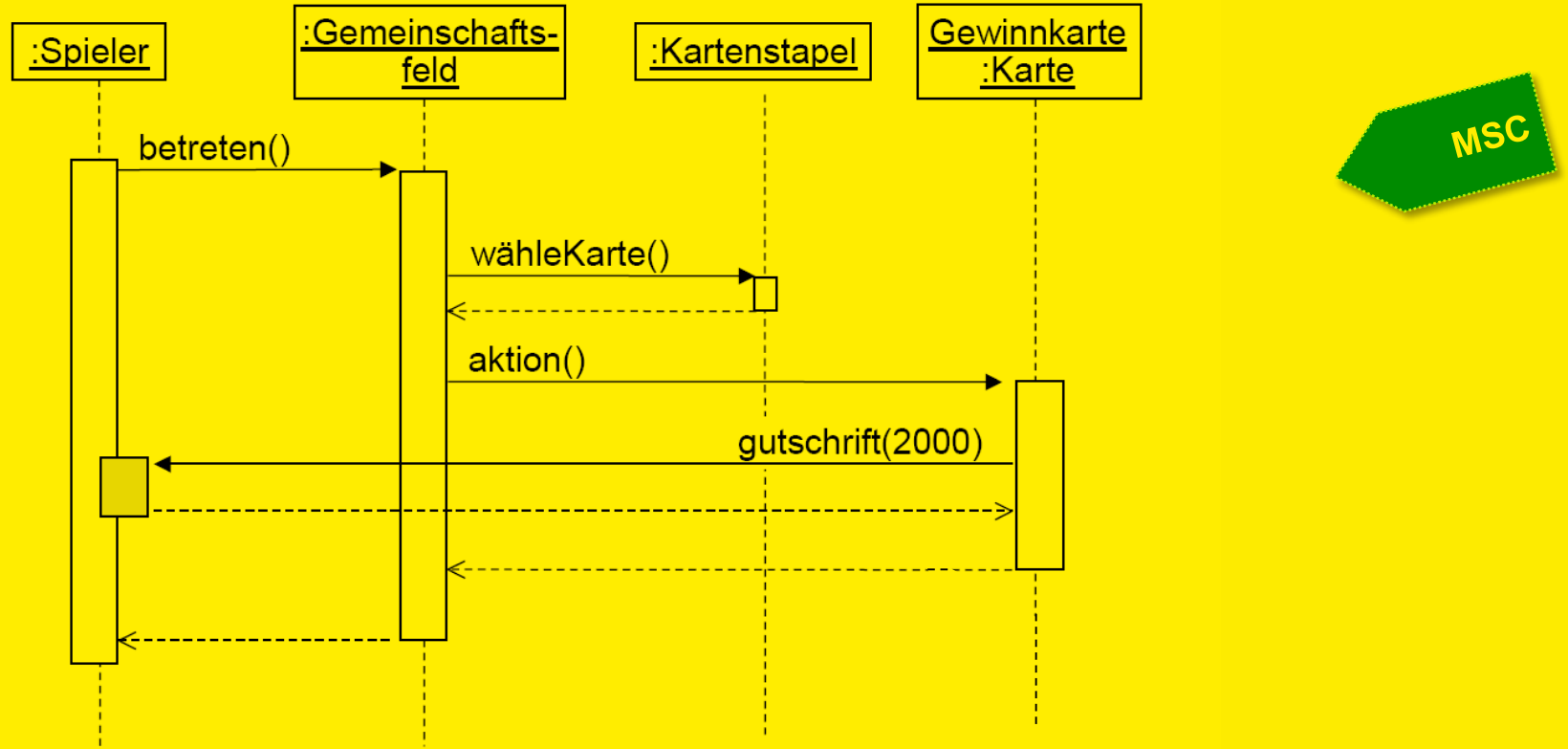
(Siehe Literatur für Details)

Sequenzdiagramme für Szenarien

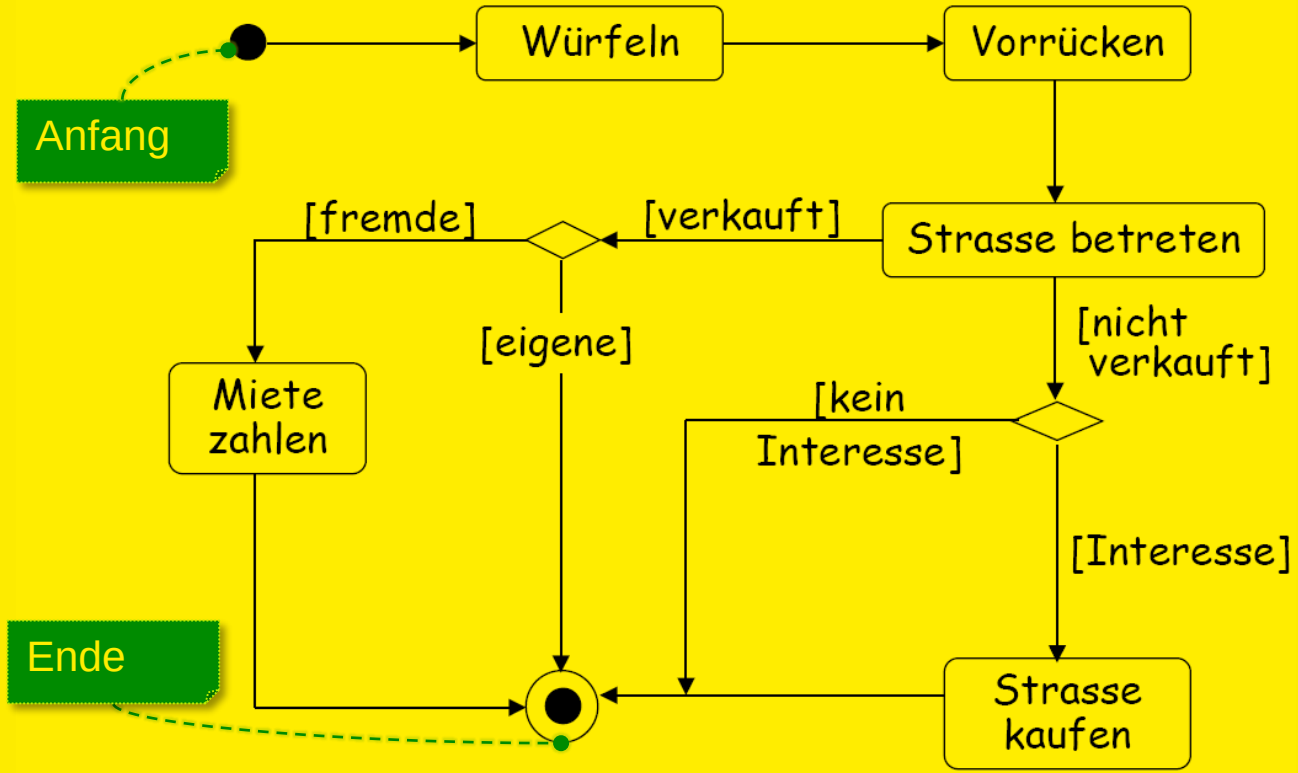


Objekt-bezogen

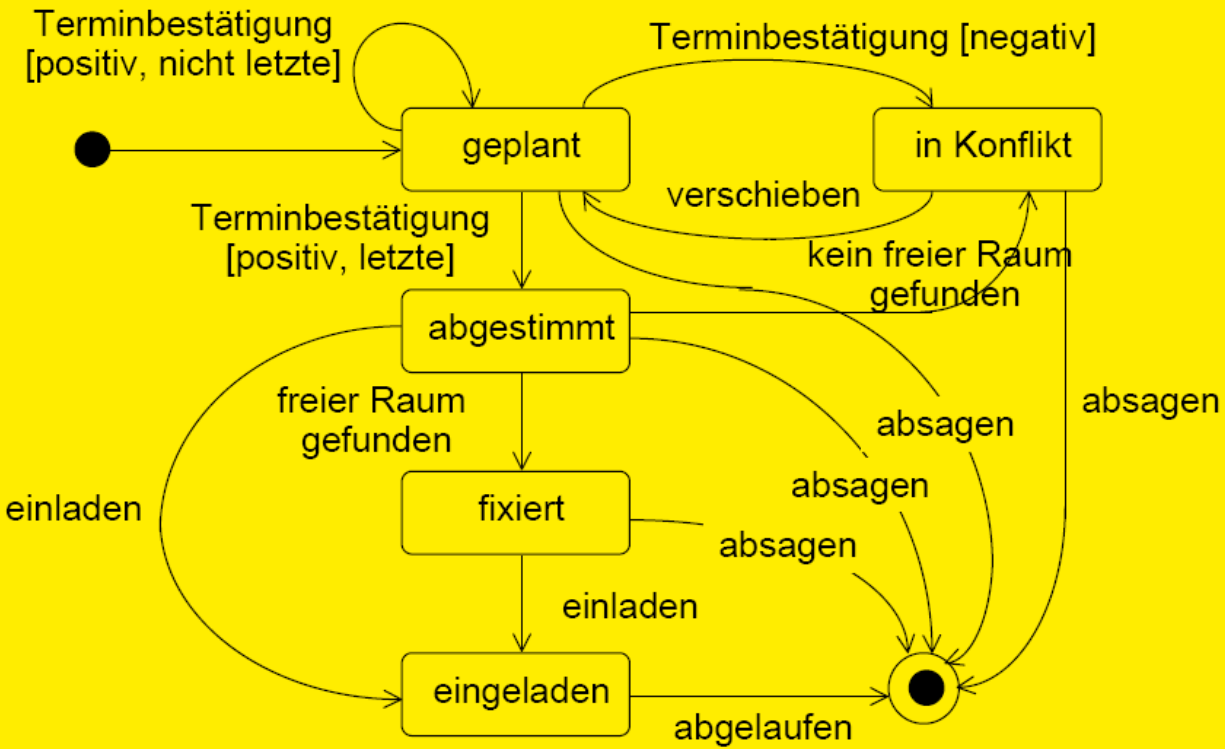
Sequenzdiagramm: Monopoly-Zug



Aktivitätsdiagramm: Monopoly-Zug

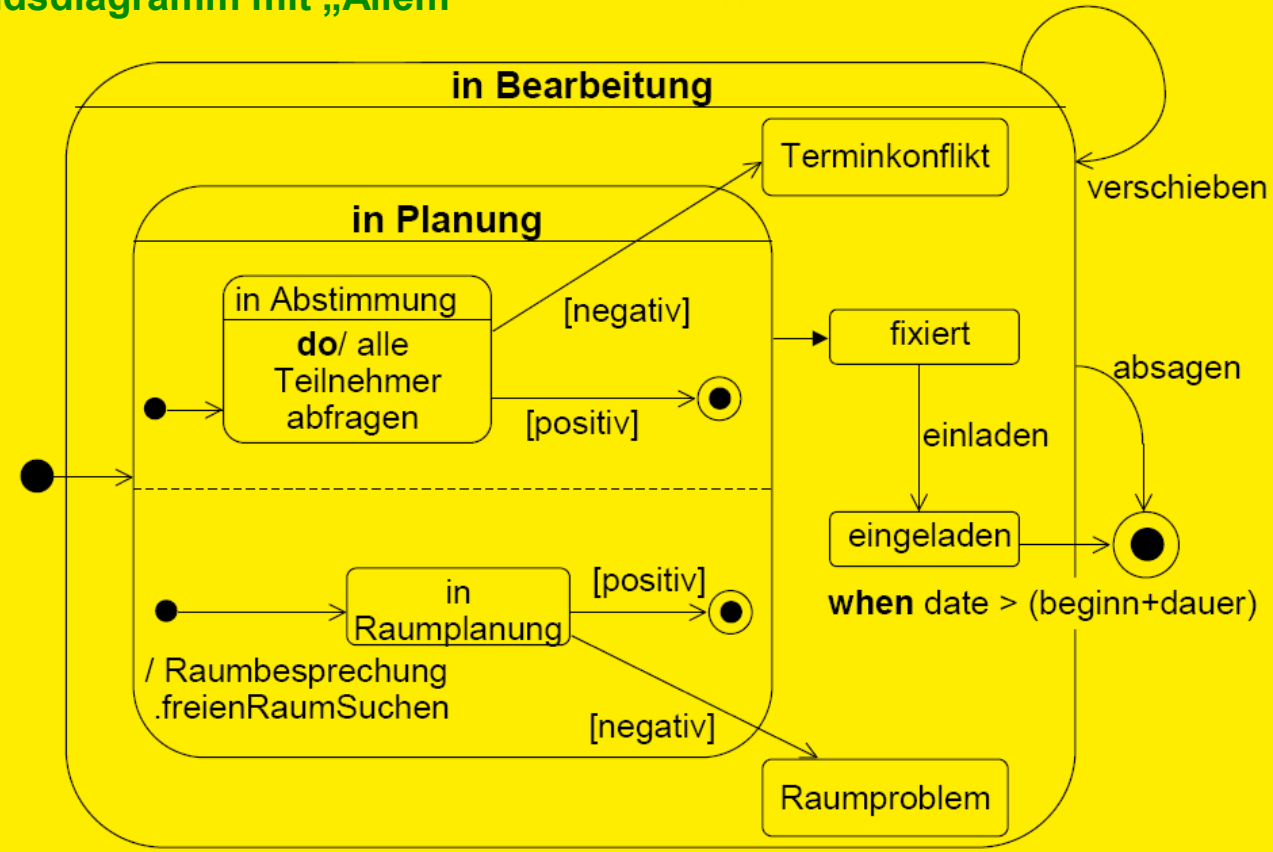


Zustandsdiagramm mit Bedingungen



Zustände von Objekten der Klasse *Teambesprechung*

Zustandsdiagramm mit „Allem“



Fragen?

Anforderungen

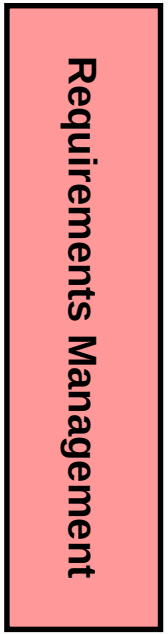
Anforderungs-analyse

Diagramme

Requirements Management

Anforderungsmanagement [AM]

→ Requirements Management



Was ist Anforderungsmanagement?

Anforderungsmanagement (**AM**) umfasst Prozesse, die notwendig sind, um…

- Anforderungen und dazugehörige Informationen für verschiedene Rollen aufzubereiten *und*
- diese konsistent zu ändern

Anforderungsmanagement muss **während der gesamten Entwicklung** berücksichtigt werden!

AM beantwortet Fragen verschiedener Rollen (I)

Projektmanager

- Was ist der aktuelle Entwicklungsstand des Projekts?
- Welchen Aufwand hat eine Anforderungsänderung?
- Wer ist zuständig für eine Anforderung?

Siehe Kapitel 4 **Projektmanagement**

Kunde

- Was ist der Ursprung einer Anforderung?
- Welche Testfälle zeigen, dass das zu entwickelnde System die Anforderungen erfüllt?
- Werden alle Anforderungen vom entwickelten System erfüllt?
- Welche Kosten hat eine gewünschte Anforderungsänderung?

AM beantwortet Fragen verschiedener Rollen (II)

Anforderungs-Ingenieur

- Sind die Anforderungen konsistent zu Vorgängerdokumenten?
- Welche Anforderungen sind von einer Änderung betroffen?

Entwerfer (Designer, SW Architekt)

- Sind alle Anforderungen im Entwurf umgesetzt?
- Auf welches Anforderungsdokument kann ich mich beziehen?
- Warum wurde eine Anforderung wie umgesetzt?

Tester

- Erfüllt das System die Anforderungen?
- Welche Testfälle müssen bei einer Änderung der Anforderungen erneut durchgeführt werden?

Wichtigkeit von Anforderungsmanagement

AM ist umso wichtiger, je…

… größer die Zahl der Anforderungen

… länger die geschätzte Lebensdauer der Software

… wahrscheinlicher Anforderungsänderungen notwendig sind

… größer die Zahl der Beteiligten

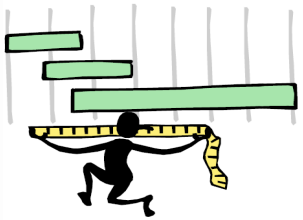
… wichtiger die Qualität der Software

Aktivitäten des Anforderungsmanagements

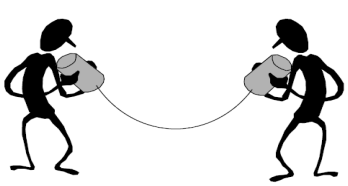
3.3.2 **Aktivitäten**

3.3.2.1 **Strukturieren**

3.3.2.2 **Bewerten**



3.3.2.3 **Verfolgen**



3.3.2.1 Strukturieren

Klassifikation

Gruppieren ähnlicher Anforderungen

Identifizierbar machen

Anforderungen ‚griffig' machen

- Kürzel

Ziele der Klassifikation von Anforderungen

- Zusammengehörigkeit und Überlappungen von Anforderungen in einer großen Menge (geschrieben von unterschiedlichen Autoren) erkennen
- Fehlende Anforderungen erkennen
- Eine Organisation von Anforderungen aufbauen, die orthogonal zur sequentiellen Dokumentenstruktur ist
- Ausschnitte aus dem Anforderungsdokument für bestimmte Rollen generieren (**Sichten**)

Beispiele für mögliche Klassifikationen

Gegenstand der Anforderung

Systemfunktionalität Benutzungsschnittstelle Datenbank
Kommunikation Beschränkungen (z.B. Performanz)

Betroffene Benutzergruppe

System-Administrator Verwaltungsangestellter Lagerarbeiter

Priorität der Anforderung

essentiell notwendig wünschenswert

Kosten/Aufwand einer Anforderung

0-5 PM* 5-10 PM >10 PM

*PM: Personenmonat

Vorgehen zur Definition von Klassen

1. *Wähle* für Projekt passende Struktur aus einem Standard aus
   z.B. unternehmensinterne Vorgaben
2. *Identifiziere* die (wesentlichen) Rollen im konkreten Projekt
3. *Befrage* die Rollen, welche Fragen sie beantworten möchten
   Welche Sichten auf die Gesamtheit der Anforderungen werden benötigt?
4. *Überlege*, mit welchen Informationen man diese Fragen beantworten kann
5. *Definiere*, welche Informationen zu jeder Anforderung erfasst werden
6. *Definiere*, wer die Informationen wann und wie erfasst
   Achtung – je mehr Informationen, desto höher der Aufwand bei der Erfassung!

Anforderungen identifizierbar machen (Kürzel)

Anforderungen sollten identifizierbar sein durch ein eindeutiges und sich nicht mehr änderndes ***Kürzel***.

Dieses Kürzel dient der Referenzierung, und sollte zur Vermeidung von Missverständnissen folgendes preisgeben:

- den Typ der Anforderung
- das Herkunftsdokument

Beispiele

LH-FA11	Funktionale Anforderung 11 im Lastenheft
PH-FA23	Funktionale Anforderung 23 im Pflichtenheft
PA17	Performanz-Anforderung 17
LH-FA-Temp19	Funktionale Anforderung 19 des Teilsystems „Temperatur“ im Lastenheft

3.3.2.2 Bewerten

Priorisierung

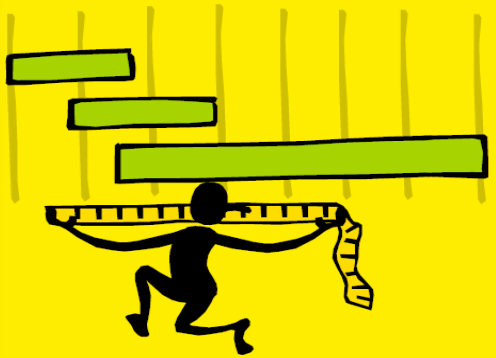
Gewichtung der Anforderungen

Beispiel für eine 3-stufige Priorisierung

- Essentiell (PrioA)
- Notwendig (PrioB)
- Wünschenswert (PrioC)

Bewerten von Risiken

Planung des mit einer Anforderung verbundenen Implementierungsrisikos



Realisierungsaufwand abschätzen

- Planung der Kosten einer Anforderung

Ziele der Priorisierung von Anforderungen

Ziel: Analyse von Anforderungen steuern

- Akzeptable Kompromisse finden zwischen in Konflikt stehenden Zielen
- Releases der Software planen
  (zuerst die wichtigen Anforderungen realisieren, dann die unwichtigen)

Beispiel

Essentiell	Prio A	die Anforderung muss implementiert werden, sonst ist das System *nutzlos*
Notwendig	Prio B	das System ist ohne Umsetzung dieser Anforderung weniger *effizient/effektiv*
Wünschenswert	Prio C	das System wäre mit dieser Anforderung *attraktiver*

➢ Die Zuordnung von Prioritäten zu Anforderungen hängt von dem zu entwickelnden System ab.

Prüfen von Prioritäten

Werden *alle* Anforderungen als essentiell eingestuft,
dann ist die Priorisierung sinnlos.

➢ Prüfung der Prioritäten notwendig:

Wäre die Anforderung noch immer essentiell, wenn es…

… doppelt soviel kosten würde, diese Anforderung zu realisieren?

… dafür die Anforderung X des Beteiligten Y nicht realisiert werden kann?

Anforderungen können auch relativ zueinander priorisiert werden:

Beispiel: {REQ7, REQ8, REQ9} **>** REQ1 **>** REQ2

Wenn sich Beteiligte nicht über die Priorität einer Anforderung einigen können, dann liegt ein Konflikt vor.

Mögliches Vorgehen zur Priorisierung

Schritt 1
Kunde bestimmt Zufriedenheit auf Skala von 1-5,
wenn Anforderung im System umgesetzt ist

Schritt 2
Kunde bestimmt Unzufriedenheit auf einer Skala von 1-5,
wenn Anforderung im System nicht umgesetzt ist

Schritt 3
Entwickler priorisiert die Anforderung,
basierend auf den zwei Zahlen

Ziele des Bewertens von Risiken

Ziele

- Unvollständige Informationen *identifizieren*
- Notwendige Korrekturen an den Anforderungen frühzeitig *erkennen*
- Projektmanagement *unterstützen* (Kostenabschätzung und Risikomanagement)

➢ Risikoanalyse ist insbesondere wichtig bei neuen Softwaresystemen

Beispiele für Arten von Risiken

Performanz

Fraglich, ob gewählte Hardware Performanz-Anforderungen genügt

Sicherheit

Fraglich, ob Sicherheitsanforderungen realisiert werden können

Aufwand

Realisierung erscheint technisch schwierig und im Aufwand unvorhersagbar

- z.B. müssen verschiedene Implementierungs-Mechanismen erst evaluiert werden

Stabilität

Anforderung ist nicht stabil und wird sich in absehbarer Zeit ändern

Abhängigkeit

Die Änderung zieht weitere Änderungen nach sich, und bedeutet extremen Änderungsaufwand

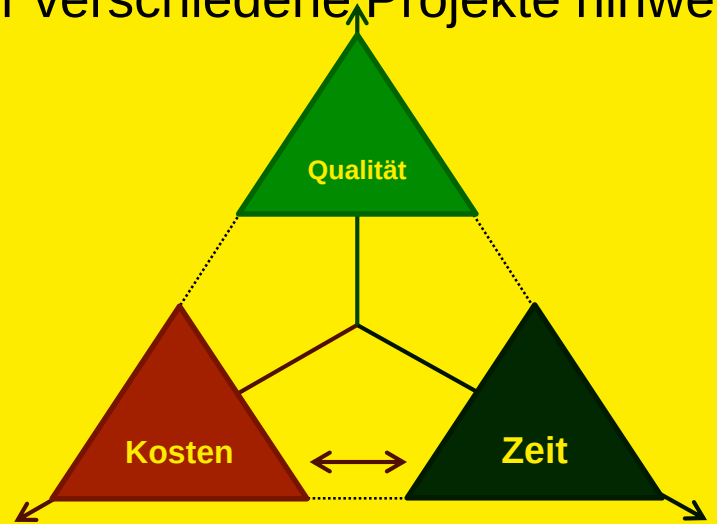
Ziele der Abschätzung des Aufwands

Ziele

- Genaue Abschätzung der Kosten des gewünschten Systems
- Genaue Planung der Systementwicklung
- Verbesserung der Messung über verschiedene Projekte hinweg

Beispiele für Maße:

- Function Points
- Boehm‘s COCOMO



Beispiel: Anforderungsspezifikation (eine Anforderung)

1.1.2.8.2 Diagramm Selektion (PA) ac158cdf1070cd71b97f

Details

Owner:	dagebauer	**Version:**	12	**Status:**	accepted
Type:	WHAT	**Priority:**	prioB		

Description: Das aktive Diagramm folgt der Selektion in der Projektansicht.

Dieses Feature ist an und abschaltbar in der Actionbar (Doppelpfeil).

Comment:

Attributes

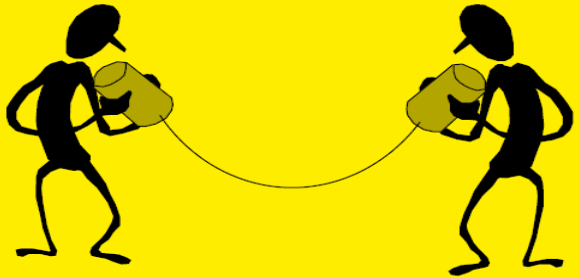
Attribute	Value
Preconditions	
Postconditions	
Reviewed By	clreichm; meberw
Revised Description	
Review Comments	
Estimate	1
Number of Ambiguities Found	0
Target Version	V1.0.1
Part of Test Suite @ Dev Team	false
System Test Needed	true
System Test Priority	A
System Test Frequency	always
TraceTo	1.1.2 Ansichten 1.1.6 Editoren
TraceFrom	
ReceiveFroms	
ReceiveTos	

3.3.2.3 Verfolgbarkeit

Anforderungsverfolgbarkeit umfasst die Dokumentation von Beziehungen einer Anforderung:

Beziehungsarten

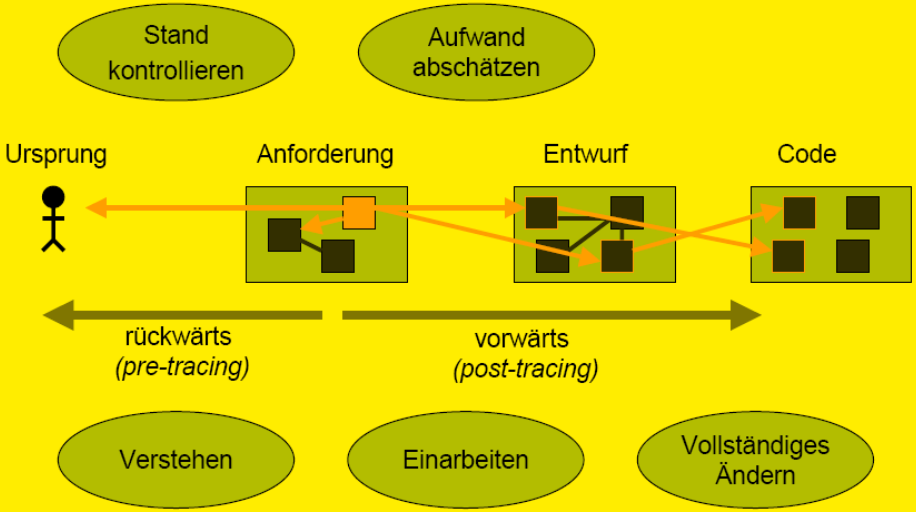
Element	Typ
andere Anforderung	horizontal
Dokumentations-Element	vertikal
andere Version	evolutionär



Ziele von Verfolgbarkeit

» *Anforderungsverfolgbarkeit* (Requirements Traceability) ist die „Fähigkeit, das Leben einer Anforderung zu beschreiben und verfolgen zu können; und zwar sowohl vorwärts als auch rückwärts ...“

– **Gotel et. al.**



Typische Verknüpfungspunkte der Anforderungsverfolgbarkeit

Quellen
Verweise auf Personen/Stakeholder, die Anforderungen einbringen

Begründungen
Verweise auf Begründungen
z.B. Gesetze, Normen, Geschäftsziele

Anforderungen
Verweise auf andere Anforderungen
z.B. übergeordnete Anforderungen

SW–Architektur
Verweise auf Teilsysteme, die Anforderungen umsetzen

Testfälle
Verweise auf Testfälle, die Anforderungen abprüfen

Schnittstellen von Fremdsystemen

Wer dokumentiert wann und wie Beziehungen?

Wer?	Ersteller eines Dokuments	Wissen
	Verantwortlicher für Verfolgbarkeit	Kein Entwicklungsaufwand Zugriffsrechte einfacher
Wann?	Während Entwicklung	Wissen
	Nach Fertigstellung	Aufwand
Wie?	Explizite Links ➢ *z.B. Querverweise, Matrizen*	Analyse einfach
	Implizite Links ➢ *z.B. Beziehungen durch Namen oder Dokumentstruktur*	Aufwand geringer
Tool?	Mit einem RM-Werkzeug	Verwaltung einfach
	Ohne spezielles Werkzeug	Anschaffungskosten

Darstellung von Verfolgbarkeits-Information

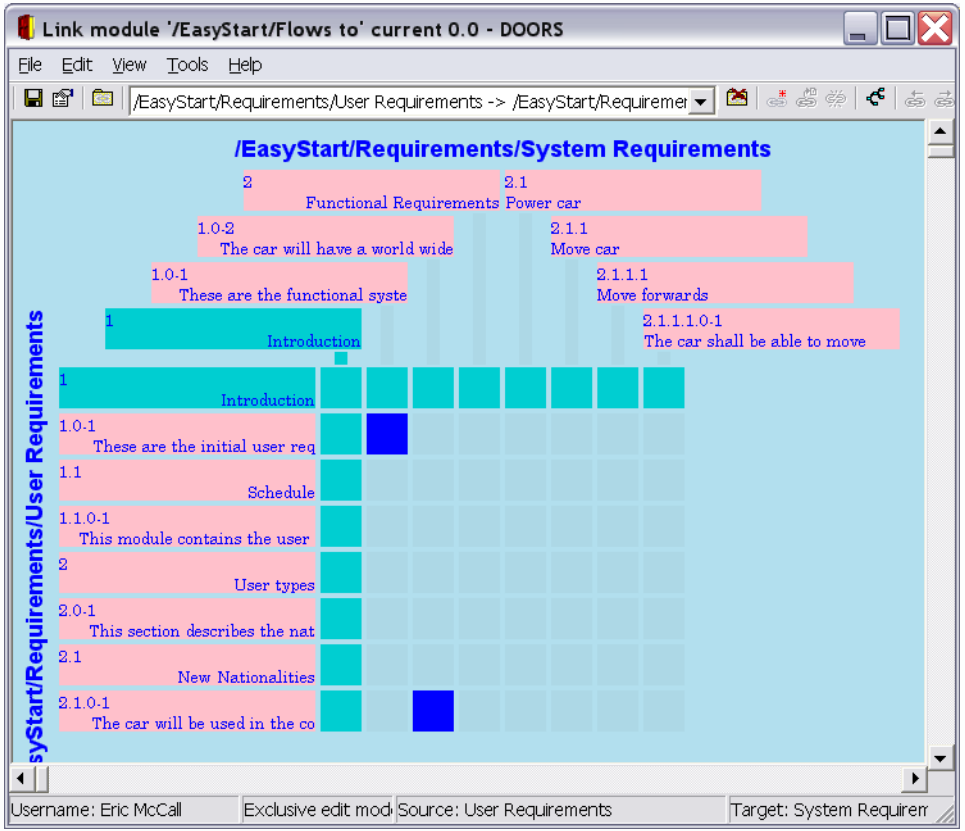
- Implizit
- Traceability-Listen

Anforderung	Testfall
UR100	T1, T2, T6
UR101	T1, T3

- Traceability-Matrix

	T1	T2	T3	T4
UR100	X	X		
UR101	X		X	

Beispiel Verfolgbarkeits-Information (DOORS)



Richtlinien zur Nutzung von Beziehungen

Definition des Zwecks

Beispiele

- Planung von Änderungen
- Durchführung von Änderungen
- Verstehen einer Entwurfskomponente
- Wiederverwendung einer Anforderung

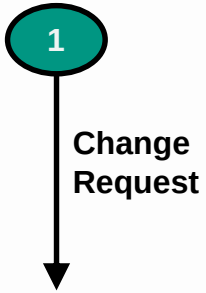
Festlegen der zu analysierenden Beziehungen

- Alle Beziehungen
- Nur bestimmte Beziehungen

Richtlinien zur Nutzung – Beispiel: Änderungen (I)

1. Identifiziere Startmenge von Änderungskandidaten
2. Identifiziere weitere Änderungskandidaten im Dokument
   - ➢ durch Analyse der „*repräsentiert*“ und „*wird verfeinert durch*“-Beziehungen
3. Identifiziere sekundäre Änderungskandidaten im Dokument
   - ➢ durch Analyse der „*ist abhängig von*“-Beziehungen
4. Prüfe, ob sekundäre Änderungskandidaten geändert werden müssen
5. Wiederhole ggf. Schritt 3
6. Identifiziere weitere Änderungskandidaten
   - ➢ durch Analyse der „*wird umgesetzt durch*“-Beziehungen

Richtlinien zur Nutzung – Beispiel: Änderungen (II)



BA-Temp-F12

Das Fenster wird geschlossen, wenn die Raumtemperatur kleiner als die Solltemperatur ist…

2 → **Übersicht**:
Das System schließt die Fenster

3 → Die Raumtemperatur wird mithilfe von 3 Raumtemperatur- Sensoren gemessen

⋮

6 → Das Fenster wird mit Hilfe von 2 Motoren geschlossen…

3.3.3 Aktivitäten – Änderungsmanagement

Änderungsmanagement (engl: Change Management) umfasst Aktivitäten zur…

- Identifikation von Änderungen
- Planung von Änderungen
- Durchführung von Änderungen

Zusammenfassung - Anforderungsmanagement

Anforderungsmanagement ist wichtig, um...

… die Zusammenarbeit von verschiedenen Rollen zu unterstützen *und*

… das Wissen über Anforderungen und Gestaltungsentscheidungen über die Lebenszeit des Systems aktuell zu halten.

Wichtige Aktivitäten

- Attributierung der Anforderungen (z.B. Priorität, Risiko, Klassen, Aufwand)
- Verfolgbarkeit innerhalb und zwischen Entwicklungsdokumenten
- Prozesse zum Umgang mit Änderungen

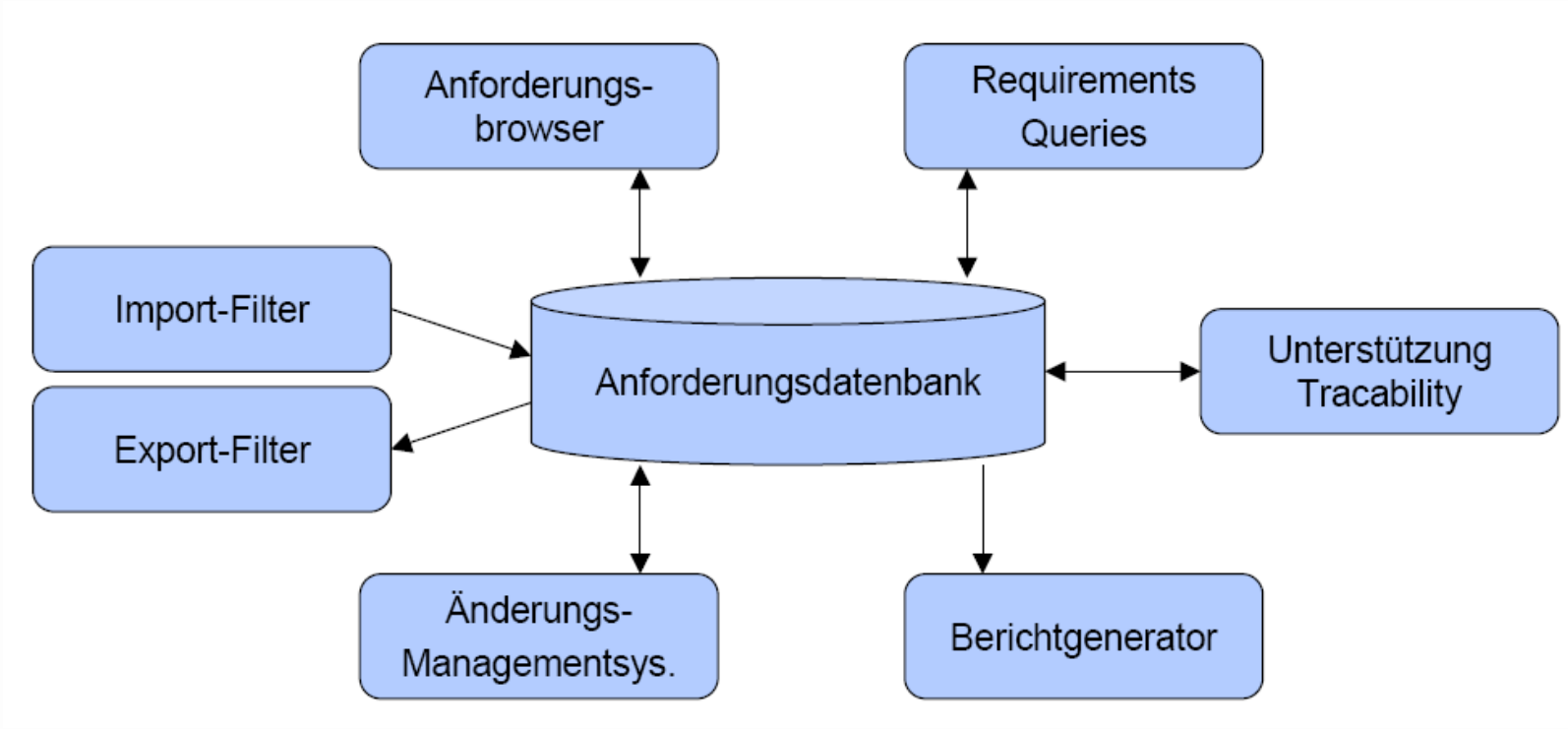
Hauptprobleme

- Dokumentations-Disziplin und Motivation der Beteiligten
- Definition eines effektiven aber gleichzeitig effizienten AM-Ansatzes

Aufgaben von Requirements Management Werkzeugen

Verwaltung aller Dokumente	Anforderungen, Testpläne, Modelle, Änderungswünsche
Verwaltung **logischer Beziehungen** zwischen den Dokumenten	Verfolgbarkeit
Editieren von Dokumenten	Mehrbenutzerfähigkeit, Zugriffskontrolle, Konfigurations- und Versionsmanagement
Organisation der Information	Gruppierung, Hierarchisierung, Attributierung mit Zusatzinformation
Reporte generieren, mit beliebiger Konfiguration	„Wo sind die Anforderungen realisiert?“ „Warum steht dieses Stück Code hier?“ etc

Grundlegende Struktur von RM Systemen



Übersicht Requirements Management Systeme

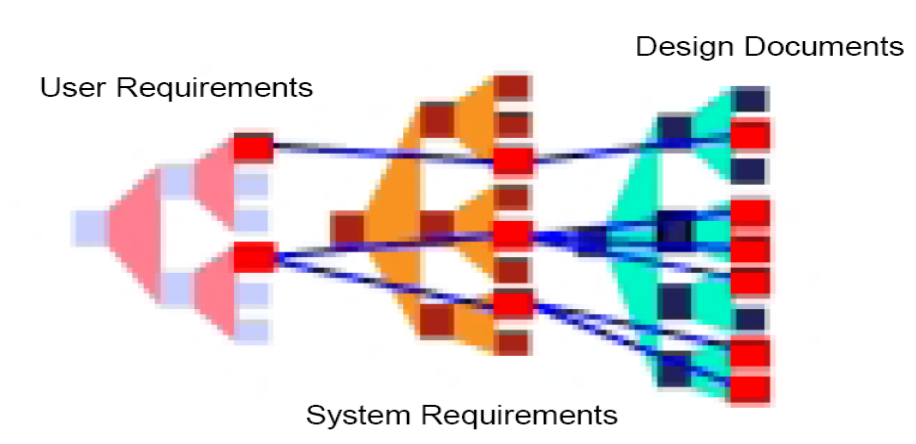
Tool Name	Vendor	Description
AnalystStudio	Rational Software	Tool Suite. Includes RequisitePro, Rose, SoDA and ClearCase
Caliber-RM	Technology Builders, Inc (TBI)	Requirements traceability tool
CORE	Vitech Corporation	Full life-cycle systems engineering CASE tool. It supports the systems engineering paradigm from the earliest days of concept development and proposal development, requirements management, behavior modeling, system design and verification process.
CRADLE/REQ	3SL (Structured Software Systems)	Requirements Management tool capable of storing within its database, graphs, spreadsheets, tables, diagrams and any other information as part of a requirement.
DOORS	Telelogic (was QSS)	Requirements traceability tool
DOORSrequireIT	Telelogic (was QSS)	Requirements trace tool that is integrated with Microsoft Word. Data can be merged with DOORS databases
GMARC	Computer Systems Architects (CSA)	Generic Modeling Approach to Requirements Capture (GMARC). Toolset will also generate quality metrics for a specification enabling formal proof that use of the GMARC has improved the requirement set.
icCONCEPT	Integrated Chipware	Requirements traceability tool. Replaces RTM
IRqA	TCP Sistemas e Ingenieria	Integral Requisite Analyzer. A requirements management tool, but also a requirements analysis environment, that includes facilities to support problem domain modeling and automatic domain analysis.
ITraceSE	ITrace Systems	Requirements traceability tool
Life*CYCLE	Computer Resources International	Requirements traceability tool. (No longer available)
RDT	IGATECH Systems Pty Limited	Requirements traceability tool
RequisitePro	Rational Software	Requirements traceability tool. Also part of AnalystStudio
RIMS	Sygenex Incorporated	Requirements and Information Mamagement System (RIMS).
RTM	Integrated Chipware	Requirements traceability software. See icCONCEPT product.
RTS	Electronic Warfare Associates, Inc.	Requirements Traceability Systems (RTS). Complete foundation for tracking the requirements of a software/hardware project through the accompanying documentation and source code. This includes tracking the development and testing status of requirements
SLATE	SDRC SSG	System Level Automation Tool for Engineers (SLATE) is used to capture, organize, build and document system-level designs from raw concepts through structural partitioning. Interfaces to Office 97, Project and CASE tools.
Systems Engineer	Blue Spruce	Requirements trace tool
Tofs	Tofs AB	Tool For Systems. Assists you in realizing and managing not only software, but also the manual (human) and hardware (electronic, hydraulic, mechanic, etc) parts of a system, which complete the system's missions together with the software.
Tracer	RBD, Inc.	Requirements traceability tool
XTie-RT	Teledyne Brown Engineering	Requirements traceability tool

http://www.incose.org/productspubs/products/setools/tooltax/reqtrace_tools.html, 2013

RM Werkzeug DOORS - Philosophie

Verwaltung verschiedener Klassen von Dokumenten, z.B.

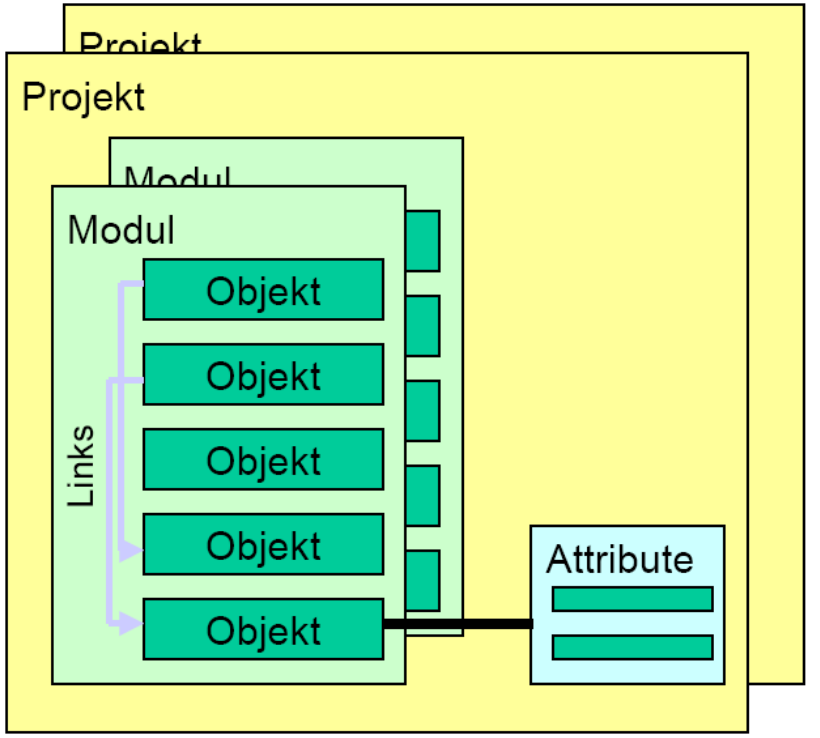
- User Requirements
- System Requirements
- Design Documents
- Test Cases



Nachvollziehen des Zusammenspiels der Dokumente über Links

➢ Traceability

RM Werkzeug DOORS - Grundstruktur



Ein Objekt entspricht einer Anforderung

Gleichartige Anforderungen (bzgl. Abstraktion) werden in einem Modul zusammengefasst

Modularten:

- „Formale“ Module (siehe Grafik)
- Deskriptive Module (Plain Text)
- Link Module (verwalten die Links)

Dokumentation von Requirements

- Jedes DOORS-Objekt sollte *genau eine* Anforderung enthalten
  - Objekte haben Überschrift-Eigenschaft
  - Objekte stehen in einer Objekt-Hierarchie (keine Vererbung o.ä.)

  } Realisierung von Dokumenten-Strukturen mit Kapitelnummern *(Ein- und Ausblenden von Ebenen ist möglich)*

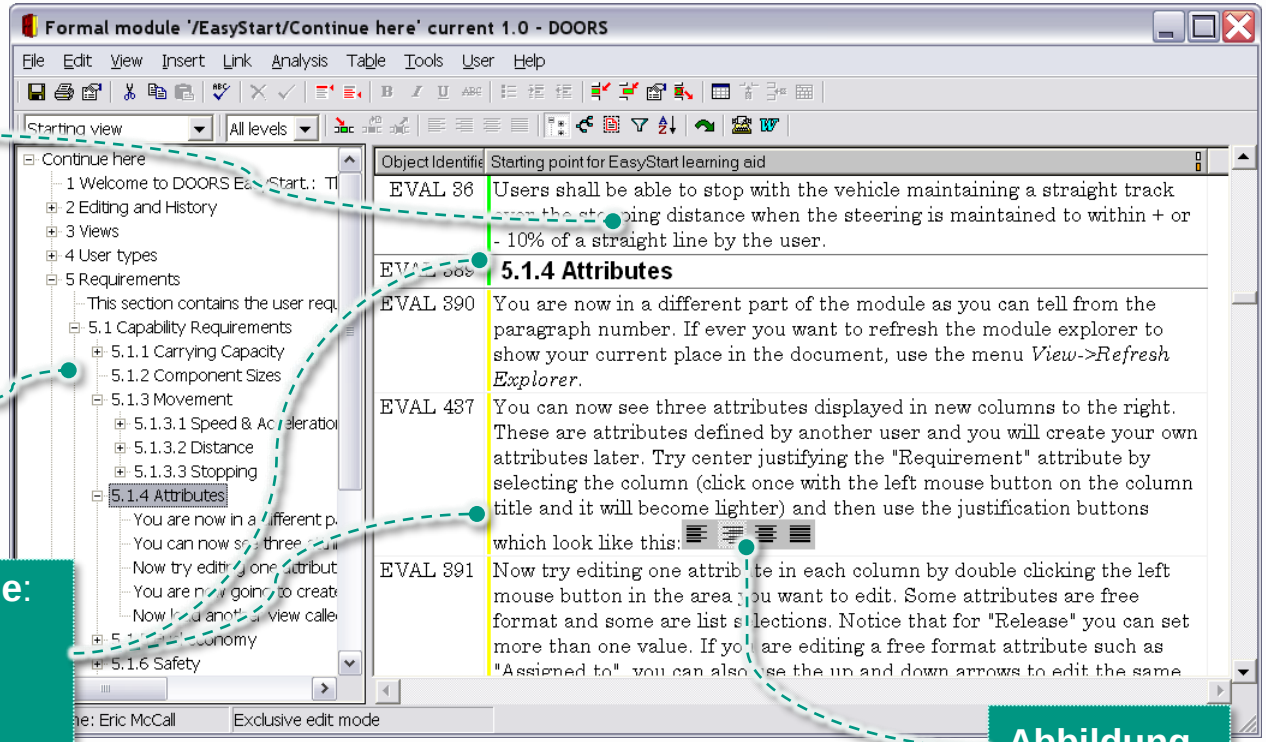
  - Der Inhalt eines Objekts ist völlig frei
    - ➢ die inhaltliche Verantwortung liegt voll beim Autor
  - Übliche Editierfunktionalität wird unterstützt

Dokumentation von Requirements

Textuelle Anforderung
(keine Vorgabe bezügl. Struktur o.ä.)

Dokumenten-Struktur

Bearbeitungsstandanzeige:
- Grün (nicht verändert)
- Gelb (neu)
- Rot (geändert)



Abbildung

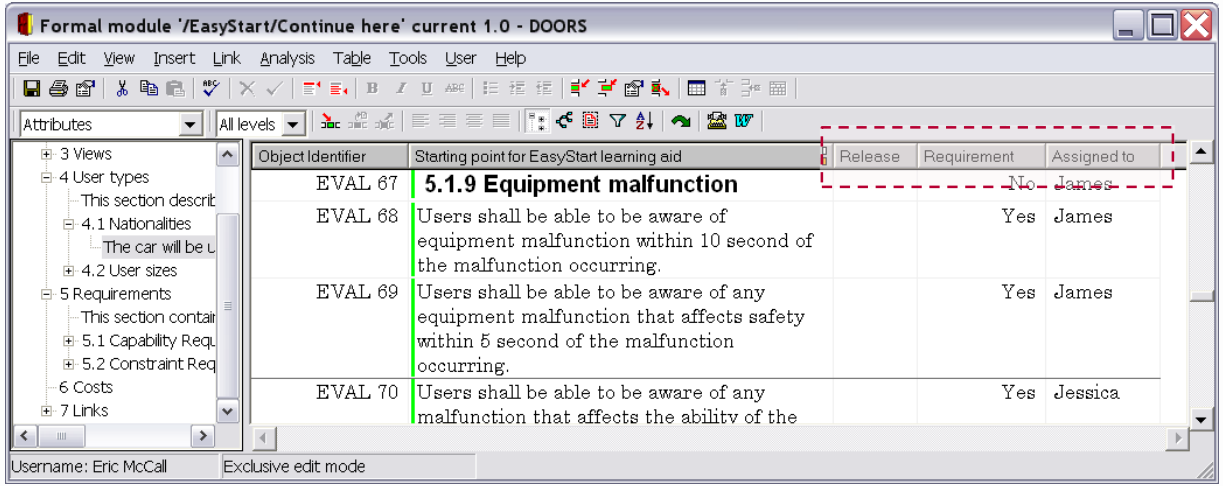
Attribute und Sichten (I)

Jedem Objekt sind Attribute zugeordnet

- Systemseitige Attribute (z.B. Object Identifier, Ersteller, Datum, ...)
- Benutzerdefinierte Attribute (z.B. Bearbeitungsstand, Testbarkeit, ...)

Attributtypen

- Zahlen
- Zeichenketten
- Aufzählungstypen
- Mengenwertige Aufzählungstypen



Attribute und Sichten (II)

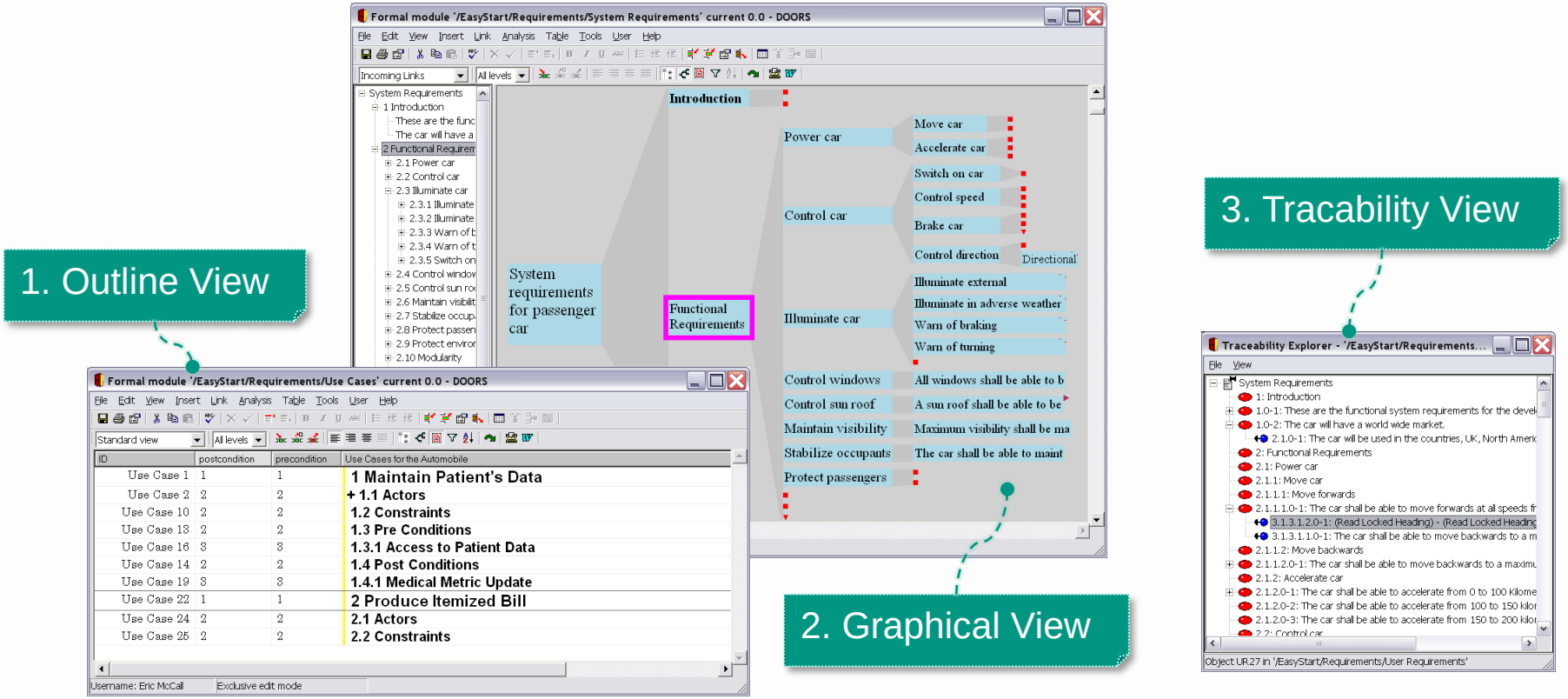
Drei Darstellungsformen

Outline View
textuelle Darstellung, mit und ohne Filterung

Graphical View
graphische Darstellung der Struktur

Traceability View
Zusammenhang der Anforderungen

Attribute und Sichten (III)



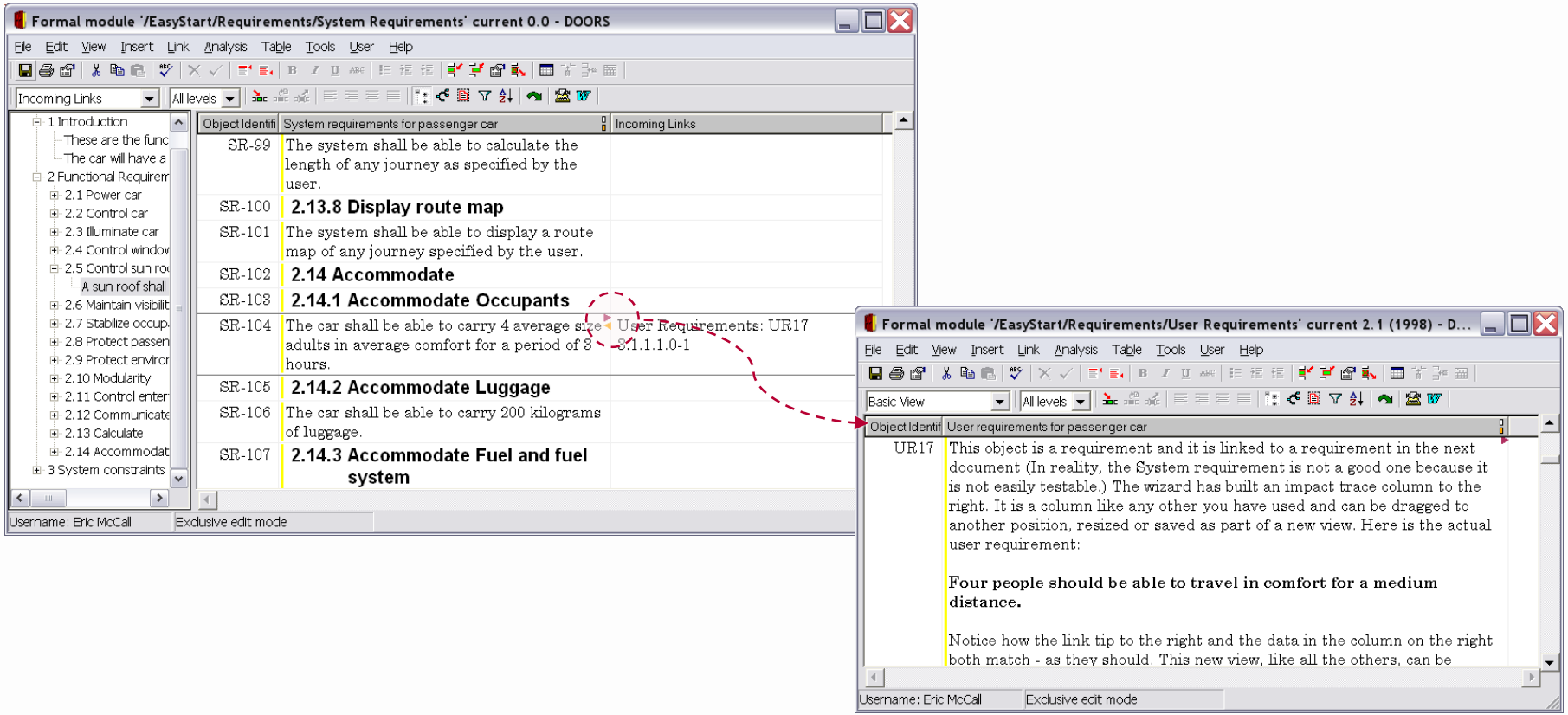
Links und Requirements Tracing (I)

Links verbinden zwei Objekten (bidirektional) miteinander

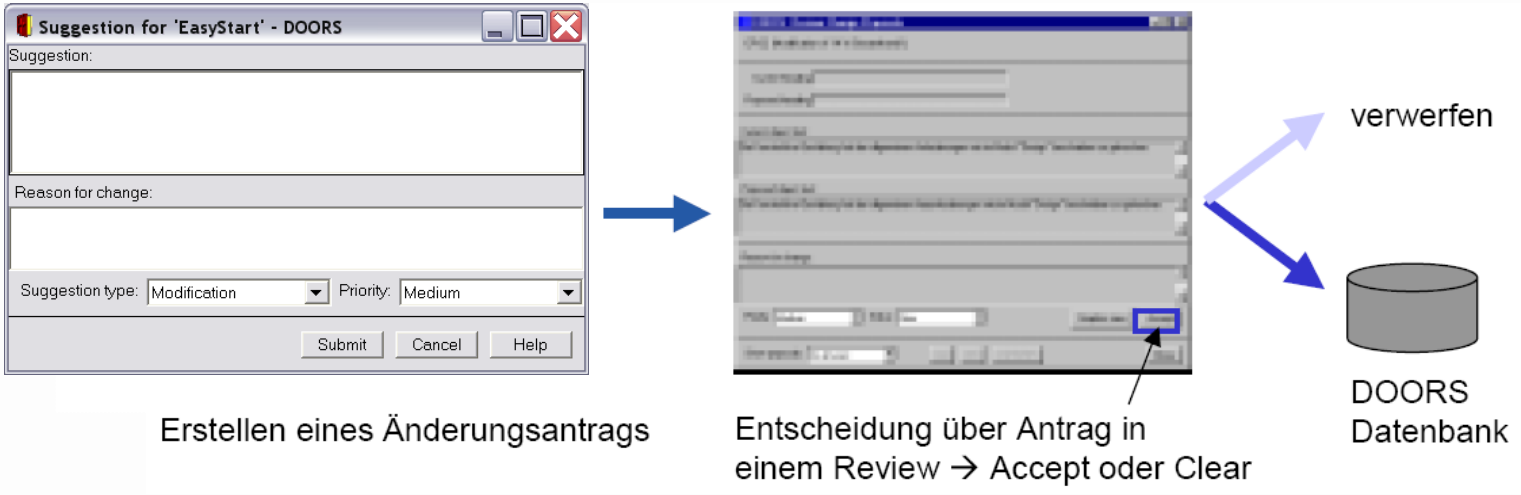
- Innerhalb eines Moduls
- Zwischen Modulen

SR-103	**2.14.1 Accommodate Occupants**	
SR-104	The car shall be able to carry 4 average size adults in average comfort for a period of 3 hours.	User Requirements: UR17 3.1.1.1.0-1

Links und Requirements Tracing (II)



Change Request System



- Einfaches Konfigurationsmanagement über sog. Baselines
  (d.h. Einfrieren eines Änderungsstandes unter einer neuen Versionsnummer)
- Für jedes Objekt wird die Historie mitgeführt (wer hat wann was verändert?)

Fragen?

Rollen

Aktivitäten

Änderungs-
management

Systeme