Projektgruppe

Christoph Fröhlich

Verfolgbarkeit von Anforderungen im Software-Entwicklungsprozess

4. Juni 2010

Motivation

- Anforderungen

- Software

Motivation

	Systemkomponente 1	Systemkomponente 2	Systemkomponente 3	Systemkomponente 4
Anforderung 1	X			
Anforderung 2		X	X	
Anforderung 3			X	
Anforderung 4	X			X

- X in Zeile heißt, dass Anforderung durch Systemkomponente erfüllt wird

Verfolgbarkeit von Anforderungen ist ja einfach zu realisieren…

… oder – doch nicht?

Motivation

- Anforderungen

- Software

- Anforderungen
- Änderungswünsche
- Hardwareanforderungen
- Testergebnisse
- Personaländerungen
- Software

Motivation

	Systemkomponente 1	Systemkomponente 2	Systemkomponente 3	Systemkomponente 4
Anforderung 1	X			
Anforderung 2		X	X	
Anforderung 3			X	
Anforderung 4	X			X

- X identifiziert betroffene Komponenten bei Änderung einer Anforderung

Aber stellt das schon Verfolgbarkeit sicher?

WARUM wurde denn beispielsweise etwas geändert?

Literaturgrundlage

- Chris Rupp: Requirements-Engineering und Management. 4. Auflage. Hanser, München, Wien 2007, ISBN 3-446-40509-7, S. 409-442
- Ramesh, Balasubramaniam und Matthias Jarke. „Toward Reference Models for Requirements Traceability.“ IEEE Trans. Softw. Eng. 27, no. 1 (2001): 58-93
- Gotel, Orlena C. Z. und Anthony C. W. Finkelstein. „An Analysis of the Requirements Traceability Problem.“ In, 94-101, 1994

Übersicht

- Motivation
- **Was ist Verfolgbarkeit**
- Warum Verfolgbarkeit gewährleisten
- Wie lässt sich Verfolgbarkeit realisieren
- Verbesserungsmöglichkeiten

Was ist Verfolgbarkeit

„Unter Traceability versteht man die Nachvollziehbarkeit von Informationen und deren Abhängigkeiten.“

„Im Rahmen des Requirements-Engineering ist meist die Nachvollziehbarkeit der Anforderungen von der Anforderungsfindung bis zur Anforderungsabnahme gemeint.“

Aus: „Requirements-Engineering und Management“

Was ist Verfolgbarkeit

- Beziehungen, Zusammenhänge und Abhängigkeiten zwischen Informationen dokumentieren
- Von Anforderungen zur fertigen Software
- Aber auch: Verfeinerungen von Anforderungen
- Verbindungen zwischen Informationen heißen Traces oder Traceability-Links

Übersicht

- Motivation
- Was ist Verfolgbarkeit
- **Warum Verfolgbarkeit gewährleisten**
- Wie lässt sich Verfolgbarkeit realisieren
- Verbesserungsmöglichkeiten

Warum Verfolgbarkeit gewährleisten

- Das Sicherstellen von Verfolgbarkeit ist mit mehr Arbeit für die Dokumentation verbunden.
  Lohnt sich diese?
- Im späten Projektverlauf muss nachvollzogen werden, warum Entscheidung getroffen wurde.
  Keine Dokumentation der Entscheidungsfindung macht dies schwer.
  Die selbe Arbeit wird dann mehrfach gemacht.

Hauptgründe

- Änderungen nachvollziehbar machen
- Verschiedene Sichten für unterschiedliche Projektbeteiligte
- Verschiedene Ebenen der Detaillierung
- Low-End vs. High-End Benutzer

Gründe für Änderungen

- Gute Systemanalyse verringert Anzahl nötiger Änderungen, kann jedoch nicht Vermeiden das Änderungen nötig werden
- Änderungen von Gesetzten, die sich auf den Geschäftsprozess oder die Technischen Anforderungen auswirken
- Wachsender Anspruch an das System (geänderte Geschäftsziele)
- Änderungen der Projektrandbedingungen (weniger Gelder stehen zur Verfügung)
- Abschätzungen über Machbarkeit stellen sich als falsch heraus (oft bei Einsatz neuer Technologien)

Der Änderungsprozess

- Änderungsgrund feststellen
  - Wirklich Änderung oder nur Verfeinerung?
- Betroffene Artefakte bestimmen
- Änderung annehmen oder ablehnen
  - Bei Ablehnung trotzdem Archivieren
- Änderung planen, realisieren, testen, abnehmen

Ebenen der Detaillierung

- Ermöglicht das zielgruppenspezifische Schreiben/Lesen von Anforderungen
- Anforderungen jeweils nur so stark detailieren wie es für die Zielgruppe nötig ist
- Grobe Anforderungen für schnellen Überblick der geforderten Funktionalität
- Anwendungsfälle schildern schon klarer was zum System gehören soll
- Nachfolgend: 5 Ebenen der Detailierung

Ebenen der Detaillierung – Ebene 0

- Mit wenigen Sätzen den Gesamtumfang des zu entwickelnden Systems beschreiben
- Soll Überblick verschaffen
- Meist in Fließtext gehalten
- Häufig auch Absichtserklärungen, Ziele des Systems und eine Liste mit Hauptmerkmalen enthalten

Ebenen der Detaillierung – Ebene 1

- Beschreibt Vorgänge in denen das System unterstützen soll
- Use-Cases oder funktionale Sichtweise

Ebenen der Detaillierung – Ebene 2

- Untersuchung der Geschäftsprozesse: „Wer macht was?“
  - Aktionen vom Anwender ausgeführt
  - Systemfunktionen
- Anforderungen ohne Differenzierung nach Teilsystemen
- Auf dieser Basis meist Auswahl der Architektur

Ebenen der Detaillierung – Ebene 3

- Mehrere Anforderungen dieser Ebene verfeinern eine Anforderung der Ebene 2
- Bilden detaillierteste Darstellung, was das System dem Anwender zur Verfügung stellen soll
- Oftmals hier auch Spezifikation der Oberflächen

Ebenen der Detaillierung – Ebene 4

- Auflösung des Systems in Soft- und Hardware
- Ggf. weitere Module, Konfigurationselemente, usw.
- Bisherige Anforderungen durch weitere (meist technische) Anforderungen ergänzen

Übersicht

- Motivation
- Was ist Verfolgbarkeit
- Warum Verfolgbarkeit gewährleisten
- **Wie lässt sich Verfolgbarkeit realisieren**
- Verbesserungsmöglichkeiten

Wie lässt sich Verfolgbarkeit realisieren

- Versionierung
- Verfeinerung
- Baselines
- Branches

- Low-End Referenz Modell
- High-End Referenz Modell

Versionierung

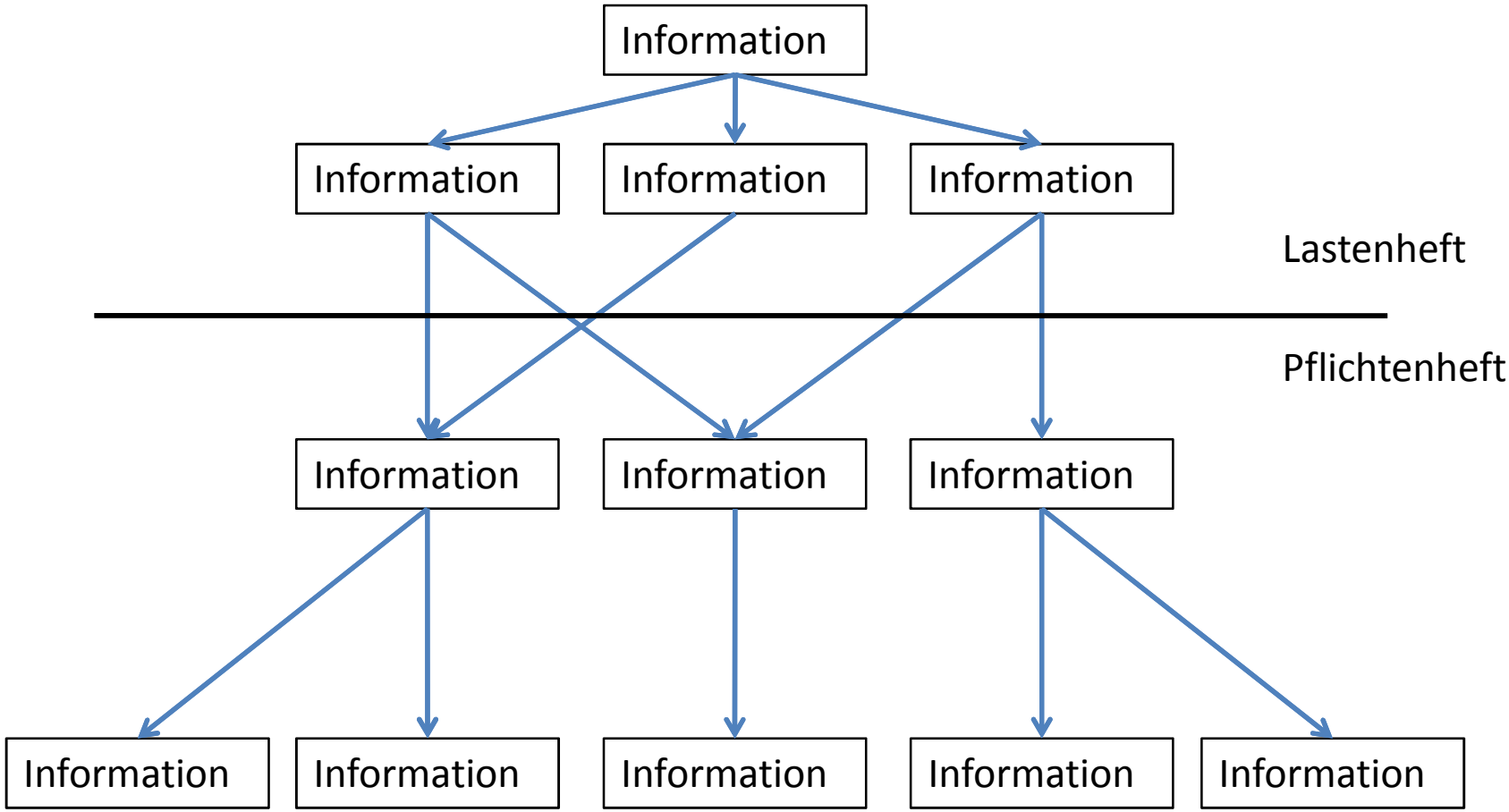
Anforderung 1	V1	V2	V3	V4
Anforderung 2	V1	V2		
Anforderung 3	V1	V2	V3	

- Neue Version einer Anforderung ersetzt die alte
- Für jede Anforderung eine Kette, letzte Element ist aktive Anforderung
- Sinnvolle Zusatzinformationen:
  Wann, Warum und von wem wurde eine neue Version erstellt?

Verfeinerung

- Erzeugt Baumstruktur:
  Jede Informationen hat genau einen Vorgänger aber mehrere Nachfolger, welche die Information verfeinern
  - § Ausnahme: Wechsel der Blickrichtung
- Unterschied zur Versionierung: Ursprungsanforderung verliert ihre Gültigkeit nicht
- Abstraktere Sicht auf Anforderungen bleibt erhalten

Verfeinerung



Die „UmZu-Regel“

- Problem bei der Anwendung von Versionierung und Verfeinerung ist zu Entscheiden, wann was vorliegt
- Hilfestellung durch die Queins‘sche UmZu-Regel
- Sprachliche Verbindung zweier Anforderungen durch „um“
- Ist Resultat ein sinnvoller Satz, handelt es sich um Verfeinerung, sonst Versionierung

- Es folgen 2 Beispiele aus „Requirements-Engineering und Management“

Die „UmZu-Regel“ – Beispiel 1

A1: Das System soll der Geschäftsleitung die Möglichkeit bieten, neue Daten anzulegen.

A2: Das System soll der Geschäftsleitung die Möglichkeit bieten, neue Mitarbeiterdaten anzulegen.

A1 + A2: Um neue Daten anzulegen, soll das System neue Mitarbeiterdaten anlegen.

Der Satz ergibt keinen Sinn, es handelt sich also um Versionierung

Die „UmZu-Regel“ – Beispiel 2

A3: Das System soll dem Gast die Möglichkeit bieten, seine Registrierungsdaten einzugeben.

A2: Das System soll dem Gast die Möglichkeit bieten, Name, Anschrift und Bankdaten einzugeben.

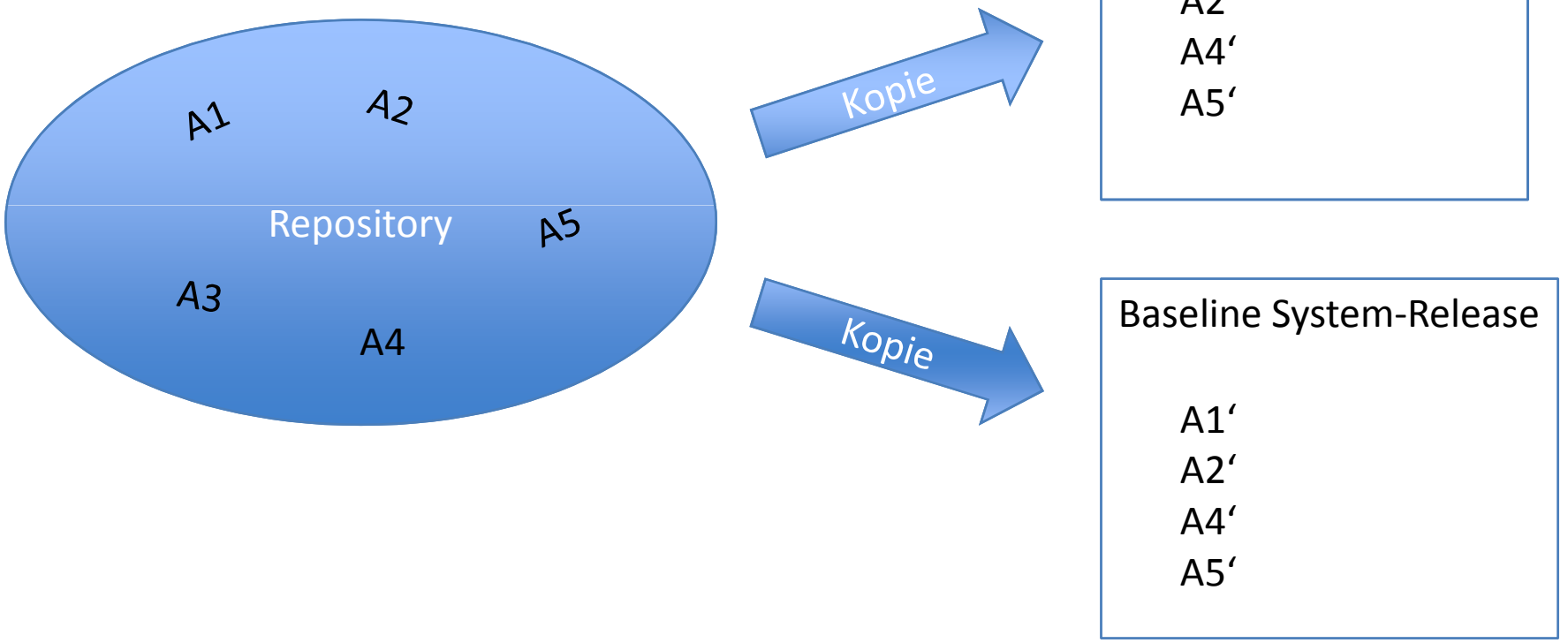
A3 + A4: Um die Registrierungsdaten einzugeben, soll das System die Möglichkeit bieten, Name, Anschrift und Bankdaten einzugeben.

Der Satz ergibt einen Sinn, daher Verfeinerung

Baseline / Document Release

- Zu definiertem Zeitpunkt eine Menge an Informationen „einfrieren“
- Hält den genauen Informationsstand zu diesem Zeitpunkt fest
- Informationen in einer Baseline dürfen nicht mehr geändert werden
- Bieten Grundlage für Gespräche zwischen Projektbeteiligten während Projekt weiter läuft
- Ermöglichen es ein Projekt später wieder aufzusetzen (Reproduktion)

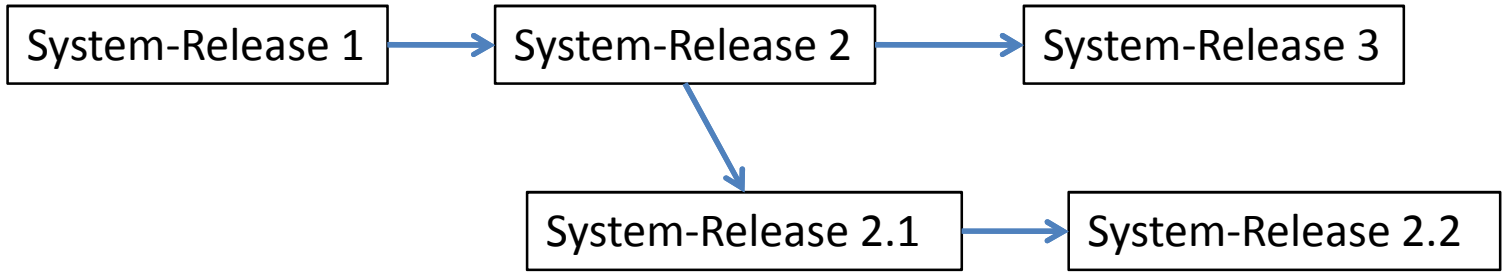
Baseline / Document Release



Branch

- Sich in kleinen Details vom Original unterscheidende Kopie einer Anforderung
- Beide Anforderungen sind aktiv
- Beide Anforderungen befinden sich auf dem selben Abstraktionsniveau (Unterschied zu Verfeinerung)
- Einsatz um „Zwischen-System-Release“ aus altem Stand der Spezifikation zu erzeugen

Branch



- System-Release 3 ist aktuelle Version
- Teilweise noch Version 2 im Einsatz
- Branch wird erstellt auf Grundlage von Version 2, um beispielsweise Fehler zu beheben

Branch

- Einsatz ist relativ kompliziert
  - Je häufiger Branches eingesetzt, desto mehr Informationen und Verbindungen entstehen
  - Müssen zusätzlich zur Versionierung/Verfeinerung verwaltet werden
  - Identifizierung von Anforderungen nicht mehr eindeutig: Zur ID kommt noch die Release Version
  - Ggf. sollen Branches wieder zusammengeführt werden

Branch

- Mehrere Lösungsvorschläge können nebenläufig entwickelt werden
- Es kann der gewünschte oder sogar mehrere (durch Zusammenführung von Branches) gewählt werden

Wie lässt sich Verfolgbarkeit realisieren

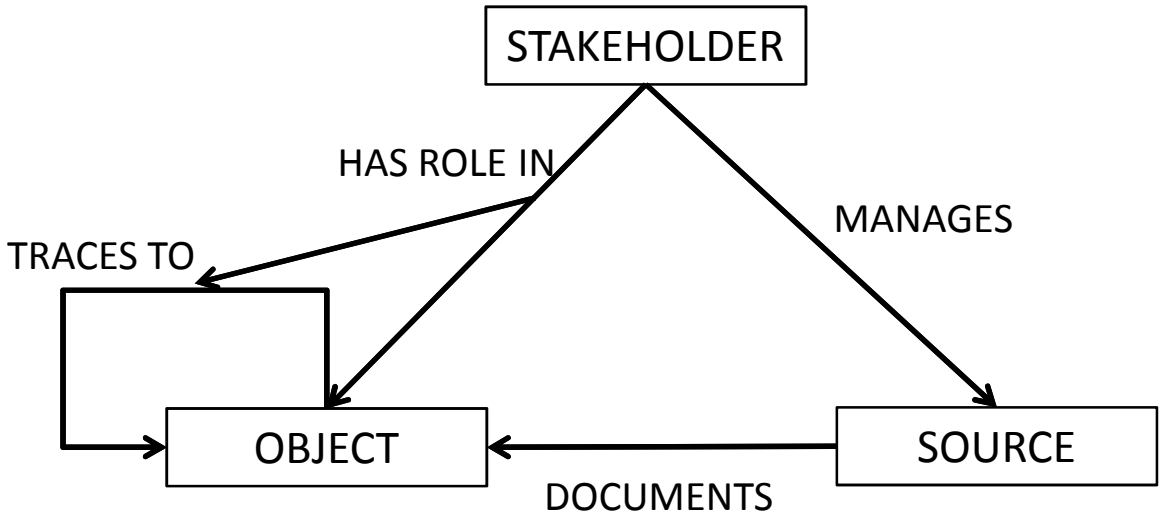
- Versionierung
- Verfeinerung
- Baselines
- Branches

- Low-End Referenz Modell
- High-End Referenz Modell

Verfolgbarkeit als Meta Modell

- Definiert Sprache in welcher Verfolgbarkeitsmodelle definiert werden können
- Modelle können für den jeweiligen Verwendungszweck angepasst werden
- Traces können in (verteilten) Datenbanken gespeichert werden

Verfolgbarkeit als Meta Modell



Was für Informationen sind gespeichert

- Alle Inputs und Outputs des Entwicklungsprozesses werden durch OBJECTs repräsentiert
- Werden durch die verschiedenen Phasen im Softwareentwicklungsprozess erstellt
- Verfolgbarkeit zwischen Objekten durch TRACES-TO Verbindungen

Wer ist betroffen

- STAKEHOLDERs repräsentieren die im Entwicklungsprozess beteiligten Aktoren
- Agieren in verschiedenen ROLEs

Wo ist die Information gespeichert

- Alle OBJECTs werden in SOURCEs dokumentiert
- SOURCEs können physische Medien sein, aber auch nicht greifbarer Natur
- STAKEHOLDERs verwalten (MANAGES) die SOURCEs

Wie ist die Information gespeichert

- Informationen können auf verschiedenen Detailebenen vorliegen
- Informationen können formalisiert sein
- Sie können, in ihrer physischen Form, verschieden aussehen:
  - § Texte
  - § Grafiken
  - § Diagramme
  - § Tabellen

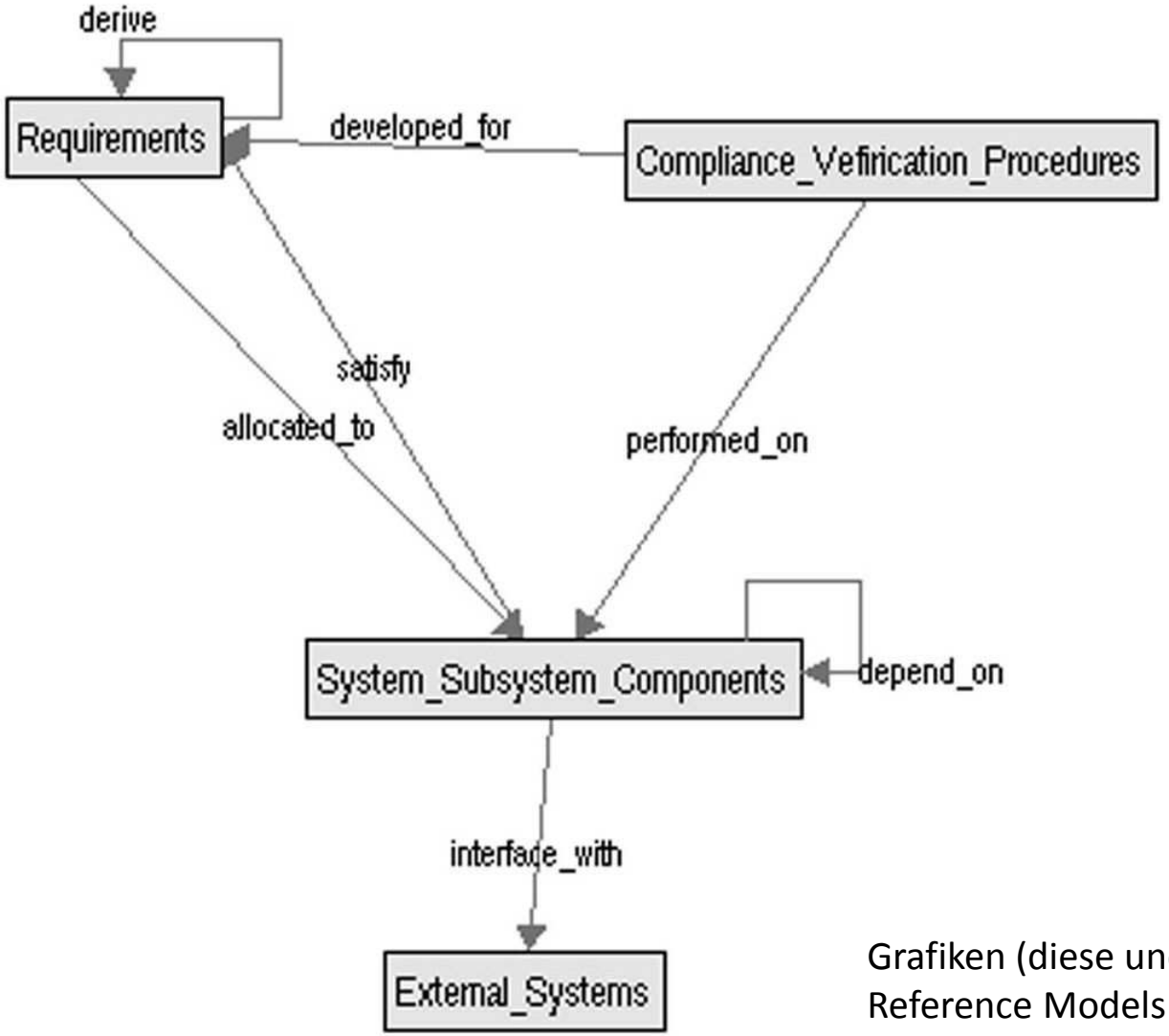
Warum wurde das OBJECT erstellt

- Die Begründungen für die Erstellung können als Spezialisierungen von OBJECTs repräsentiert werden
- Durch TRACES-TO Beziehungen können die Entscheidungen mit den konzeptionalen Objekten verbunden werden
- Noch weiter Fortgeschritten ist es so auch möglich einzelne Probleme, Argumente, Alternativen, etc. einzubeziehen

Wann wurde die Information erstellt

- Zeitinformationen können als Attribute den Objekten angehängt werden

Low-End Benutzung des Meta Modells



Grafiken (diese und folgende) aus „Toward Reference Models for Requirements Traceability“

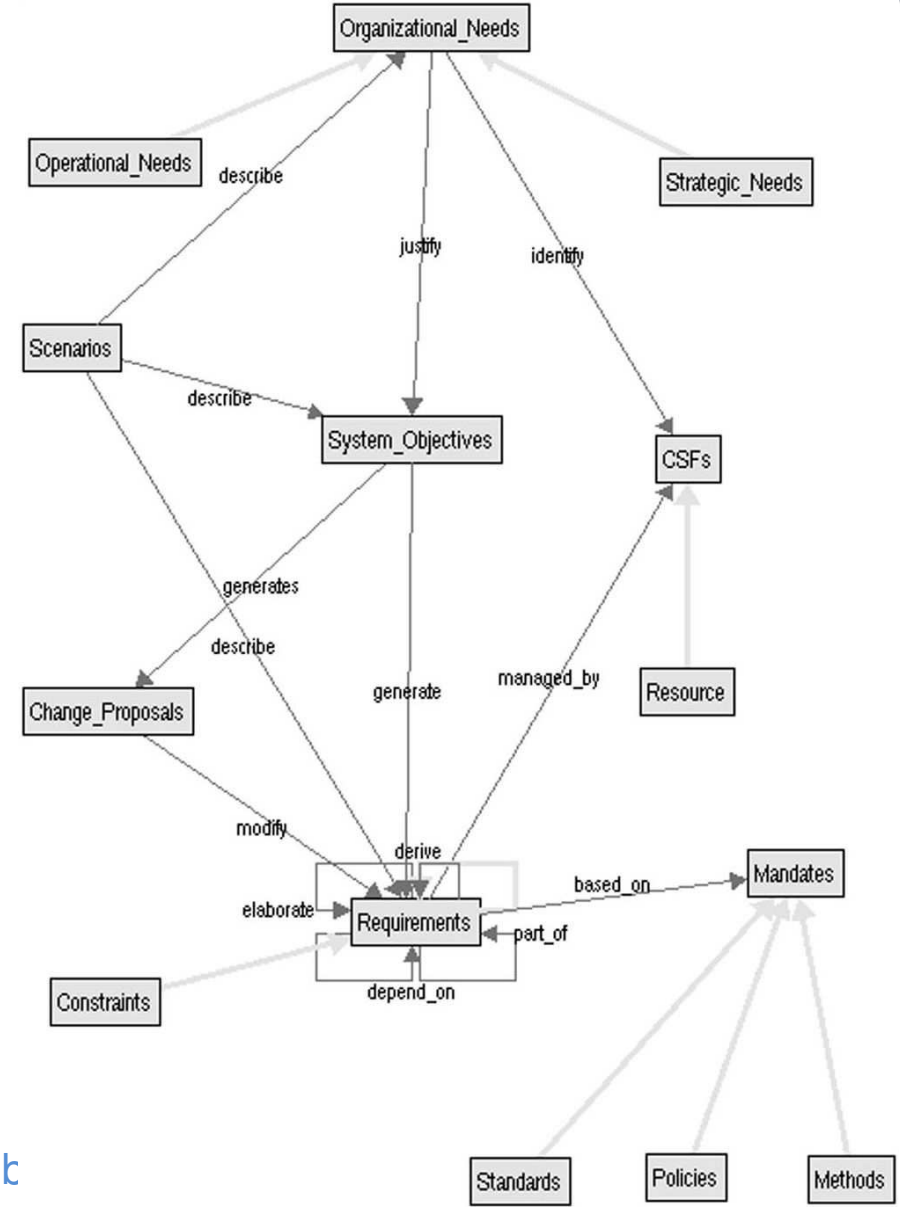
Probleme bei der Low-End Benutzung

- Änderungen sind nicht wirklich nachvollziehbar
  - Wer hat die Änderung gemacht?
  - Warum wurde die Änderung gemacht?
  - Wie wurde für (und gegen) die Änderung argumentiert?
- Bei Low-End Benutzung fehlt es bei der Dokumentation der Entscheidungsfindung

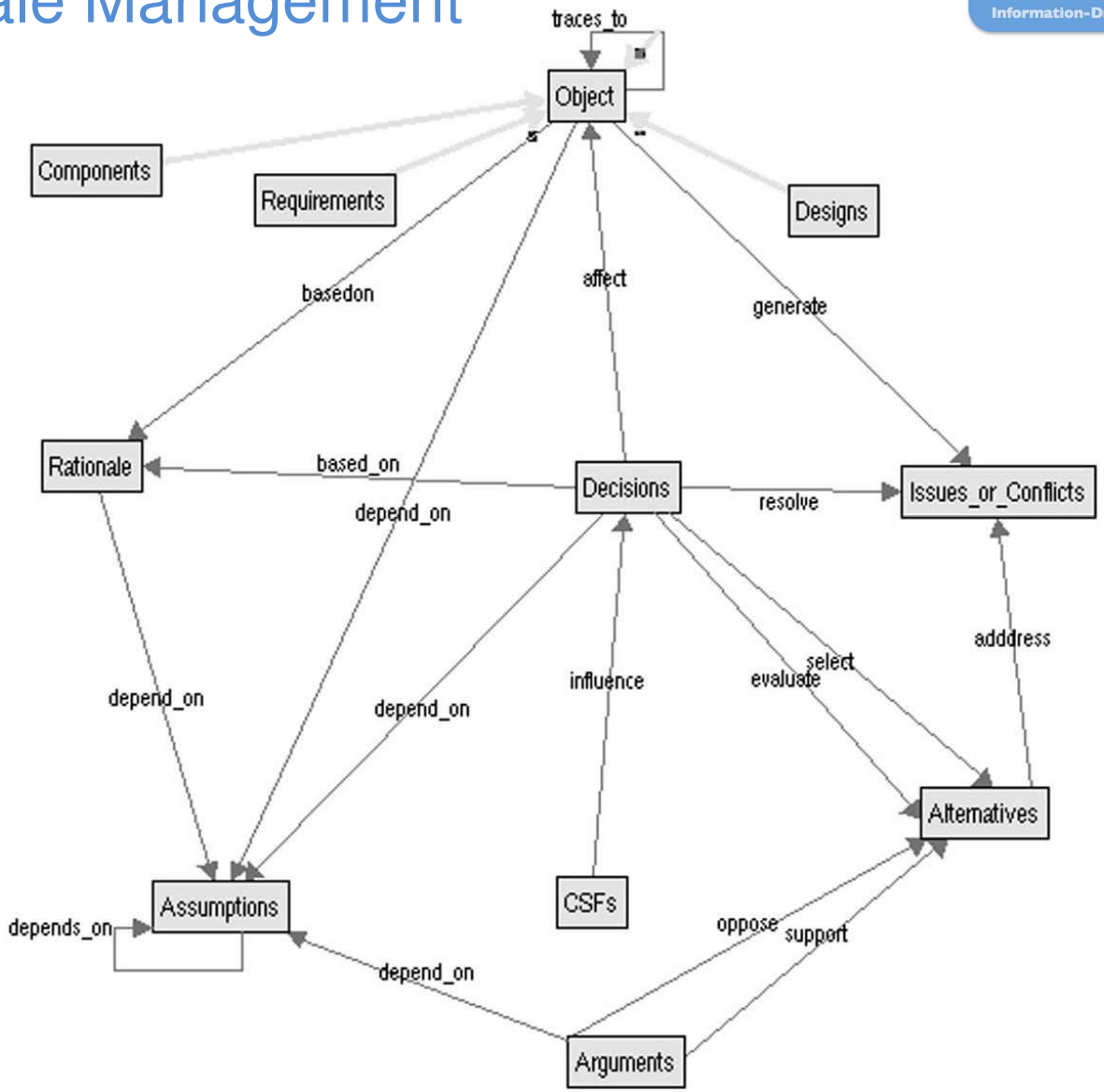
High-End Benutzung von Verfolgbarkeit

- Erfassen und nutzen von Verfolgbarkeitsinformationen auf viel höherem Level
- „Unternehmerischere“ Sicht auf die Dinge
- Das Modell wird in 4 Teile zerlegt:
  - Requirements Management
  - Rationale Management
  - Designe Allocation
  - Compliance Verification

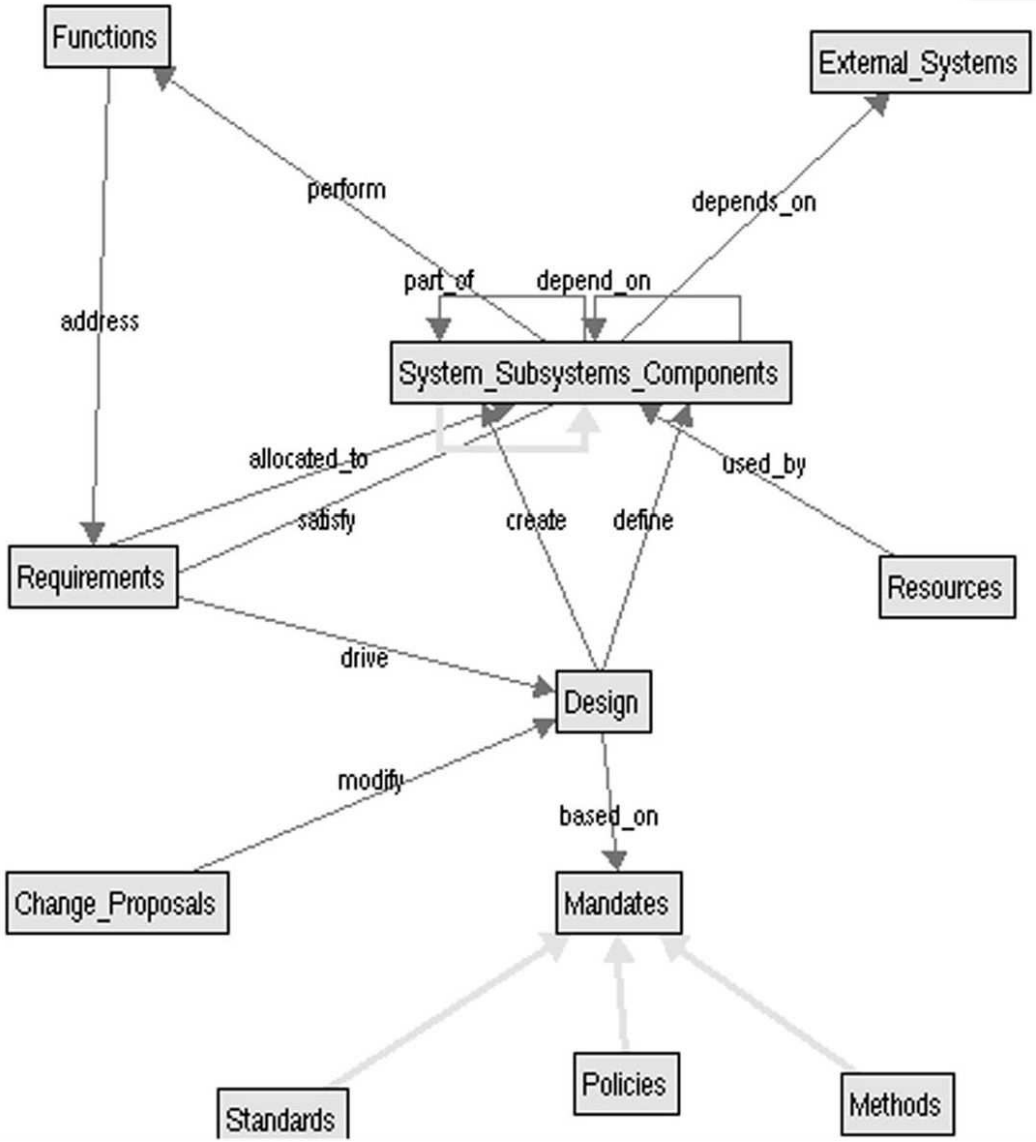
Requirements Management



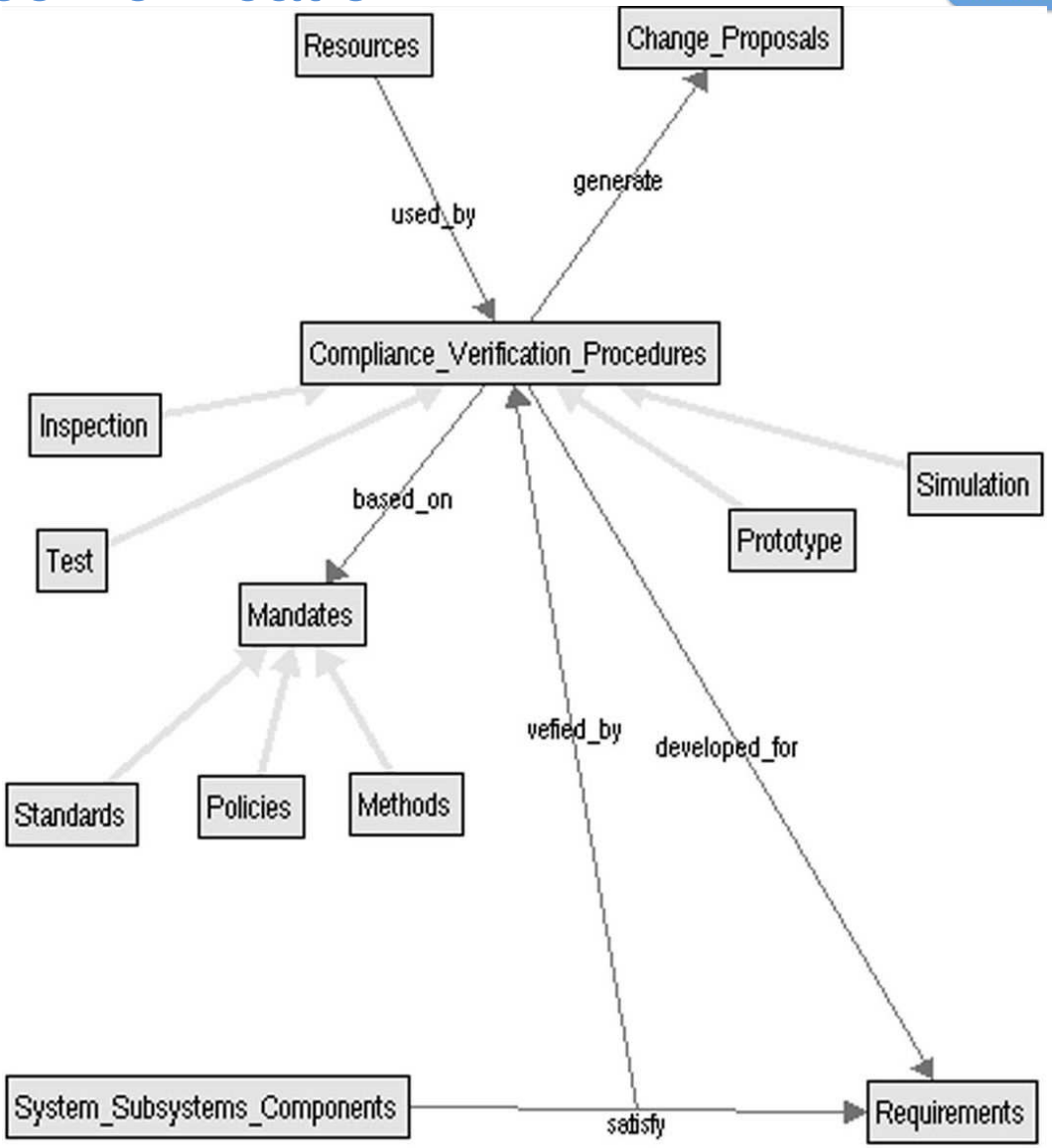
Rationale Management



Designe Allocation



Compliance Verification



Problem der High-End Benutzung

- Viel Dokumentationsaufwand
- Große Datenmenge an Informationen muss überblickt werden
- Viele Teile der Dokumentation erfolgen per Hand zu wenig Unterstützung durch Automatisierung

Übersicht

- Motivation
- Was ist Verfolgbarkeit
- Warum Verfolgbarkeit gewährleisten
- Wie lässt sich Verfolgbarkeit realisieren
- **Verbesserungsmöglichkeiten**

Verbesserungsmöglichkeiten

- Verschiedene Traceability Linktypen
- Abstaktionsmethoden
- Unterstützung von ad hoc queries für Abfragen
- Modell-Driven Erstellung von Traces