

Vorlesung „Methoden des Software Engineering“

Block A „Requirements Engineering“ **Anforderungvalidierung und -verwaltung**

Martin Wirsing

Einheit A.4, 04.11.2004

Ziele

- Techniken der Qualitätssicherung von Anforderungsbeschreibungen kennen lernen
- Technik der Inspektionen lernen
- Techniken und Werkzeuge des Änderungsmanagement kennen lernen

Qualitätsprobleme von Anforderungen

Typische Qualitätsprobleme von Anforderungen aus Sicht der Leser (siehe auch IEEE-Qualitätseigenschaften):

- Syntaktisch (d.h. Formulierungsproblem)
- Semantisch (d.h. Inhaltsproblem)
- Legitimation (d.h. Verfolgbarkeitsproblem)

Analytische Qualitätssicherung

- **Inspektion von Dokumenten**
 - Konsistenz, Klarheit der Dokumente
 - Angemessenheit für Leserkreis
- **Test (später)**
 - von Realisierung, Simulation oder Prototyp
 - Benutzungstests, Systemtest, Integrationstest, Komponententest
- **Formale Verifikation (später)**
 - Modellhafte Realisierung gegenüber Spezifikation: Model Checking
 - Eigenschaften einer Spezifikation: Beweiser

Inspektion

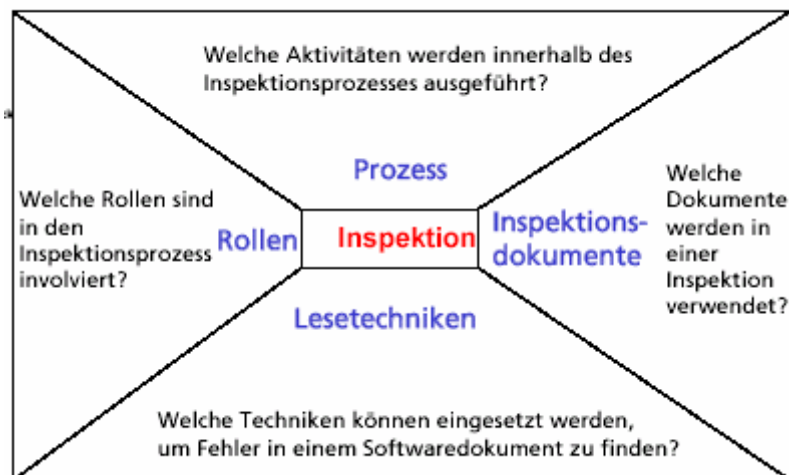
- **Inspektion**

Methode, um Qualitätseigenschaften von Softwaredokumenten zu überprüfen

- **Eigenschaften**

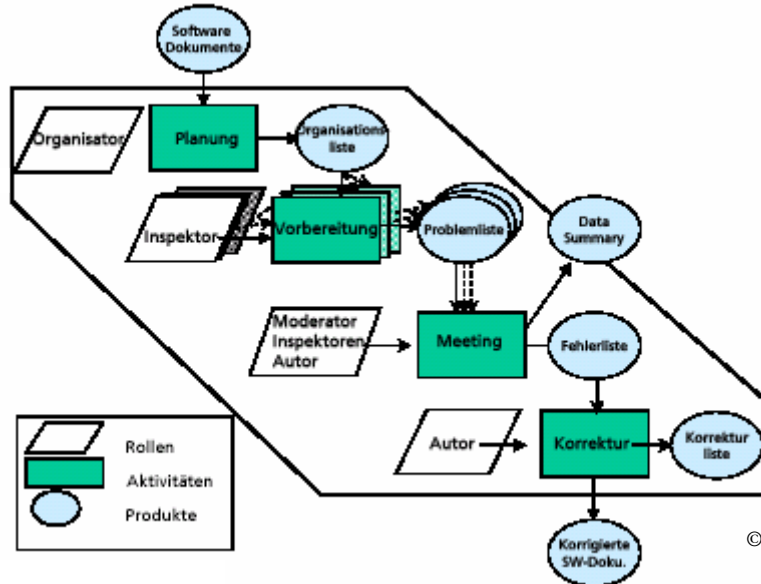
- Strukturierter, definierter Prozess
- Inspektionsteam besteht aus qualifizierten Personen
- Teilnehmer haben festgelegte Rollen
- Verwendung von bestimmten Lesetechniken zum Finden von Fehlern
- Inspektionsergebnisse werden dokumentiert

Aspekte der Inspektion



© Paech 2003

Prozess der Inspektion



Inspektionsdokumente

- **Organisationsliste**
Liste aller geplanten Inspektionsaktivitäten (innerhalb eines Projekts oder projektübergreifend).
- **Problemliste**
Dokumentation von Fehlern/Fragen/Verbesserungsvorschlägen während der Fehlersuche sowie des Aufwands für die Fehlersuche.
- **Fehlerliste**
Dokumentation der tatsächlichen Fehler in der Sammlungs- und Entscheidungsphase, sowie des Aufwands dafür.
- **Korrekturliste**
Dokumentation der Fehlerkorrektur.
- **Lesetechnik**
Dokumente zur Lesetechnik.

Vorbereitung: Problemliste

Inspektionsdokument: Problemliste

Inspektor-ID: _____

Dokumentname: _____

Dokumentversion: _____

Aufwand (Minuten): _____

Seite _____ von _____

Nr	Seite/Ort	Problemart	Beschreibung.

Ende der Fehlersuche (Datum): _____

Methoden des Software Engineering (c) 2004, Koch, Störle, Wirsing, LMU München

Fehlerliste/Korrekturliste

- vor dem Meeting auszufüllen -

Inspektionsnummer: _____
Meetingtermin: Datum: ____ Beginn: _____

Name/Version: _____ Dokument-Name: _____ Version: _____

Beteiligte: (Meeting) Autor: _____
Inspektor 1: Aufwand für Fehlersuche (Minute) ____
Inspektor 2: Aufwand für Fehlersuche (Minute) ____
Moderator: _____
Sonstige Teilnehmer: _____

- Nach dem Meeting auszufüllen -

Ende des Meetings: _____
Aufwand für das Meeting (Minuten): _____

Ergebnis des Meeting: Keine Reinspektion erforderlich
Reinspektion erforderlich

- Nach der Korrektur auszufüllen -

Ende der Korrektur (Datum): _____

Methoden des Software Engineering (c) 2004, Koch, Störle, Wirsing, LMU München

Fehlerliste/Korrekturliste

Nr.	Dokument	Seite	Zeile, Ort	Fehler Vorschlag	Kurzbeschreibung	Fehler wurde von x Inspektoren <u>in der</u> <u>Phase Vorbereitung</u> gefunden	Korrekturbemerkung des Autors	Korrektur- aufwand (min)

Ergebnisse der Inspektion

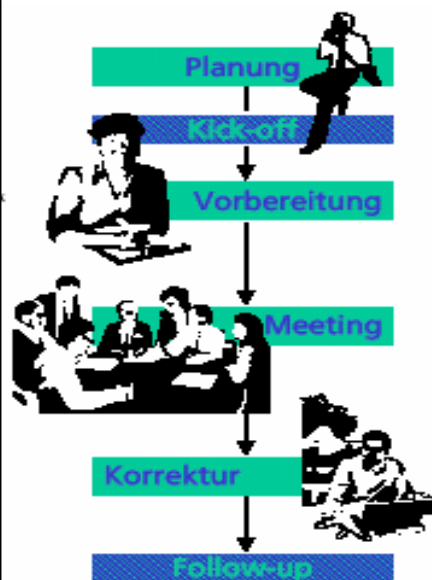
- **Ergebnisse**

- Verbessertes Softwaredokument
- Dokumentierte Information zur Charakterisierung
 - des inspizierten Dokumentes
 - des Inspektionsprozesses
 - der Software-Entwicklungsprozesse

- **Verwendung der Ergebnisse**

- Hilfe für den Autor bei der Überarbeitung seine Dokumentes
- Charakterisierung / Beurteilung/ Vorhersage / Verbesserung des Inspektions- und Softwareentwicklungsprozesses und **nicht der daran beteiligten Personen!**

Die einzelnen Aktivitäten



- Teilnehmer klären, Terminvereinbarung von allen Inspektionsaktivitäten, Räume festlegen.
- Suche nach und Dokumentation von Fehlern/Fragen/Verbesserungsvorschlägen während des Lesens des Softwaredokuments.
- Sammeln der Fehler. Entscheidung, ob ein potentieller Fehler ein tatsächlicher Fehler ist. Suche nach weiteren Fehlern. Entscheidung über Re-Inspektion.

© Paech 2003

Rollen

- **Organisator**
Nimmt zu inspizierendes Softwareprodukt entgegen. Plant alle Inspektionsaktivitäten für ein Projekt.
- **Inspektoren**
Analysiert (liest) ein Softwaredokument. Ziel: Fehlersuche. Dokumentation von Fehlern, Fragen und Verbesserungsvorschlägen.
- **Autor**
Hat das Softwaredokument erstellt.
- **Moderator**
Überprüft der Einhaltung des Inspektionsprozesses. Leitet das Meeting. Führt im Meeting durch das Softwaredokument. Übernimmt oft auch die Dokumentation der Inspektionsergebnisse.
- **Protokollführer**
Notiert die Fehler während des Meetings

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Rollenträger

- **Organisator**
Nach der Planung oft nicht in weitere Inspektionsaktivitäten involviert.
Rolle kann von Führungskraft (Qualitätsmanager) ausgeübt werden
- **Inspektoren**
Aus direktem Umfeld.
Distanz, aber dennoch mit genügend technischen Kenntnissen
- **Autor**
Kann selbst Inspektor sein. Sollte nicht Moderator sein.
- **Moderator**
Sollte nicht der Autor sein.
Kann Inspektor sein (Konfliktpotential abschätzen).
- **Protokollführer**
Diese Rolle kann von einem der Inspektoren oder dem Moderator übernommen werden

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Lesetechnik

- **Lesetechnik**
Hilfsmittel für Inspektoren. Lesetechniken erleichtern die Überprüfung der Qualitätseigenschaften von Softwaredokumenten.
- **Klassifikation**
 - Ad-hoc (keine konkrete Vorgehensweise)
 - **Checklistenbasiertes Lesen**
 - Fehlerklassenbasiertes Lesen
 - **Perspektivenbasiertes Lesen**

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Lesetechniken

- **Checklisten**
 - schnellere, objektivere und wiederholbare Analyse von Anforderungen
 - wichtige Ressource eines Unternehmens, die den gegenwärtigen Stand
 - von Erfahrungen reflektieren
 - Für Entdeckung semantischer Mängel nicht so gut geeignet
- **Perspektiven-basiertes Lesen**
 - Familie von Techniken für unterschiedliche Ziele, unterschiedliche Artefakte, verschiedene Sichten auf Artefakte
 - maßgeblich am Fraunhofer IESE entwickelt
 - im industriellen Umfeld erprobt und eingesetzt (Deutschland, USA)
 - systematischer als Checkliste
 - Besonders geeignet für Entdeckung semantischer Mängel

Anwendung von Checklisten

- **Allgemeine Checkliste** aus der Literatur auswählen
- Checkliste **unternehmens- und projektspezifisch** erweitern
- Checkliste ist nie vollständig, daher nur **Gedankenstütze** !

- **Jeder Punkt der Checkliste muss für jede einzelne Anforderung geprüft werden. Zwei Vorgehensweisen:**
 - Einen Punkt der Checkliste für alle Anforderungen prüfen, dann mit dem nächsten Punkt der Checkliste fortfahren
 - Eine Anforderung auf alle Punkte der Checkliste prüfen, dann mit der nächsten Anforderung fortfahren

Typische Checkliste für Anforderungen

- **Einhaltung des Dokumentationsstandards (Syntax)**
 - Alle Abschnitte sinnvoll ausgefüllt?
 - Rechtschreib- und Grammatikprüfung erfolgreich ?
- **Qualität des Anforderungsdokuments (Semantik)**
 - Sind die Anforderungen konsistent ?
 - ... in sich vollständig ? (z.B. alle Begriffe definiert)
 - ... verständlich ?
 - ... eindeutig ?
 - ... testbar ? ... realisierbar ?
- **Verfolgbarkeit zu Kunden-/Benutzerwünschen**
 - Sind alle Anforderungen auch tatsächlich vom Kunden/Benutzer gewünscht und korrekt wiedergegeben? Fehlen Anforderungen?

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Beispiel: Checkliste für Use Case basiertes RE (I)

Table 29-3 Quality Assessment Checklist for Team Skill 2: Understanding User and Stakeholder Needs

Structured interview, process, and results	Was a structured interview employed?	<input type="checkbox"/>
	Did it cover all the major facets of product requirements, purpose, usage, reliability, performance, deployment, support, and so on?	<input type="checkbox"/>
	Was it sufficiently free of interviewer biases so as to assure a quality result?	<input type="checkbox"/>
	Were a sufficient number of users or stakeholders identified and interviewed?	<input type="checkbox"/>
	Are there other key influencers whose needs must be understood?	<input type="checkbox"/>
Understanding of users and user needs	Do you understand who the users are and what capabilities they possess to apply your application?	<input type="checkbox"/>
	Did you discover any primary user or demographic differences that need to be addressed in the product?	<input type="checkbox"/>
	Did the highest-priority needs converge after a reasonable number of interviews?	<input type="checkbox"/>
	Are the user data, needs data, and any suggested features summarized somewhere for future reference?	<input type="checkbox"/>

© Leffingwell, Widrig

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Beispiel: Checkliste für Use Case basiertes RE (II)

Requirements workshop process and results	Was a workshop conducted that included the requisite stakeholders? <input type="checkbox"/>
	Was it conducted in such a way as to encourage input by all stakeholders? <input type="checkbox"/>
	Did the results converge on a common understanding of the system to be built? <input type="checkbox"/>
	Was the development team engaged in such a way as to provide reasonable assurances of technical and project timeline feasibility? <input type="checkbox"/>
Preliminary list of prioritized features	Does a prioritized list of features exist? <input type="checkbox"/>
	Did the development team define rough estimates of effort for each? <input type="checkbox"/>
	Was the risk of each feature established? <input type="checkbox"/>
	Is this information captured somewhere for continuous reference? <input type="checkbox"/>
Storyboards, example use cases, and other expository artifacts	If the application is innovative, did you develop some means to demonstrate the application to the user? <input type="checkbox"/>
	Was their reaction taken into consideration and is it now reflected in your current understanding of the system? <input type="checkbox"/>
	Can you describe a few exemplary use cases that describe how the system is intended to be used? <input type="checkbox"/>

© Leffingwell, Widrig

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Richtlinien für Checklisten

- Nicht länger als eine Seite
- Formulierung von Fragen
- Die Beantwortung der Frage mit „Nein“ bedeutet Fehler
- Gliederung der Fragen nach Qualitätsaspekten/Struktur des zu inspizierenden Dokumentes
- Die Fragen beziehen sich auf häufige/wichtige Fehler
- Die Fragen können von „Guidelines“ oder projektspezifischen Qualitätszielen abgeleitet werden
- Aus der Literatur können nur Anregungen/Beispiele entnommen werden. Die Checkliste muss spezifisch für das Softwareartefakt/das Projekt/das Unternehmen angepasst werden

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Perspektiven-basiertes Lesen

- Perspektiven sollten repräsentativ für die Nutzer der Anforderungsdokumente sein
- Perspektiven ergänzen sich
- Vorgehensweise:
 - Erstellung einer Abstraktion anhand einer Anleitung
 - Beantwortung einer Fragenliste während und nach Erstellung der Abstraktion
- Geeignete Abstraktionen für Use Cases:
 - z.B. Entwicklerperspektive: Klassenmodell
 - z.B. Testerperspektive: Testfall
 - z.B. Nutzerperspektive: Szenario

Typische Perspektive eines Testers

Imagine you are the tester of the system requirements document.

As part of your work you have to develop a test plan and test cases from the systems requirements document. It is essential for your work that you are able to derive testing processes and test case from separate parts of the system requirement. The main quality aspect of the systems requirements document is the verifiability of the system requirements.

Based on the use cases of the subsystem under inspection, sketch test cases for system and acceptance testing.

1.

For each functionality relevant for the subsystem (see test plan matrix created in step 1) identify the use cases that realize this functionality. To do so, create a matrix with the name of the use cases in the first row and list the functionalities in the first column. Mark each cell with a cross if the use case realizes the functionality
Template:.....

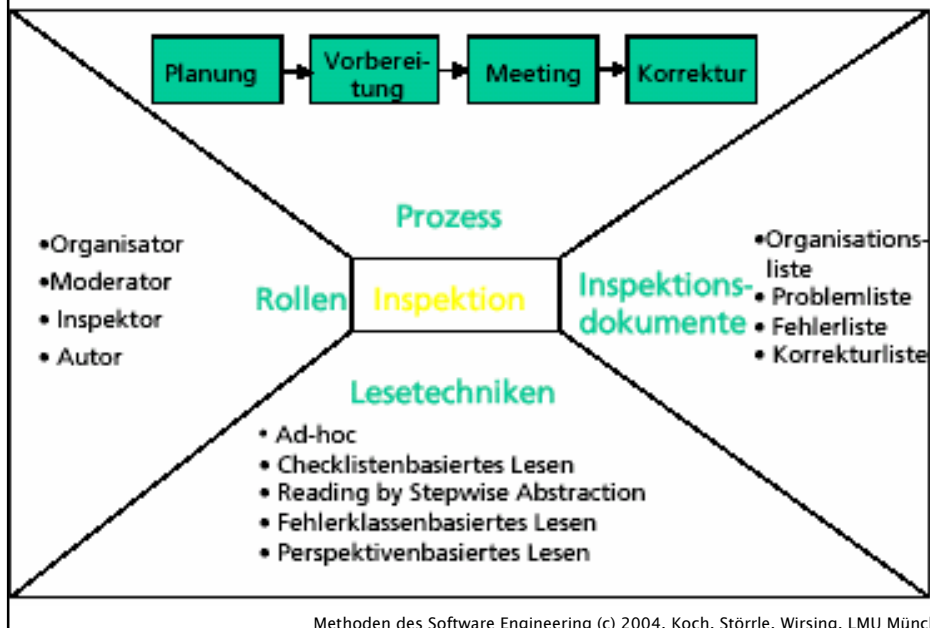
1. For each action of the actors all the expected system reactions are clearly described in the use cases.

2. All possible actor actions are considered in the use cases.

3. All actors that are interested in a certain system reaction are considered in the use case.

4 All functions are realized in at least one use case.

Inspektionen auf einen Blick



Anforderungsmanagement

• Anforderungsmanagement

Prozess der Verwaltung der Änderungen der Anforderungen an ein System

• Aktivitäten

- Verwaltung der Änderungen von Anforderungen
- Verwaltung der Beziehungen zwischen Anforderungen
- Verwaltung der Abhängigkeiten zwischen dem Anforderungsdokument und anderen Dokumenten der SW-Entwicklung

• Grundvoraussetzung: Anforderungen müssen verfolgbar (traceable) sein

Eine Anforderung ist **verfolgbar**, wenn man nachvollziehen kann,

- wer die Anforderung vorgeschlagen hat,
- warum es die Anforderung gibt,
- welche anderen Anforderungen dazu in Beziehung stehen und
- wie Anforderungen zu anderen Informationen wie Entwurf, Implementierung, Test und Benutzeranleitungen in Beziehung stehen .

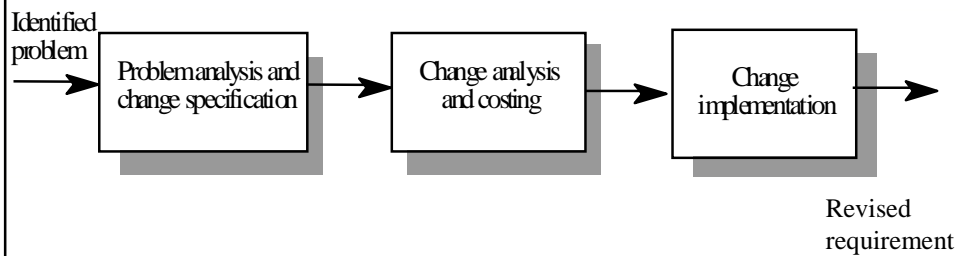
Beispiele für Relationen zur Verfolgung von Anforderungen (Requirements tracing)

- **Verfolgung von**
 - Systemeigenschaften (Features) zu den Benutzerwünschen und dem für den Wunsch verantwortlichen Benutzer
 - Use Cases zu geforderten Systemeigenschaften/-Zielen
 - der Realisierung eines Use Case zum zugehörigen Use Case
- **Und/Oder-Verfeinerung von Zielen**

Gründe für Änderung von Anforderungen

- **Fehler, Konflikte, Inkonsistenzen bei Anforderungen**
die während der Analyse, Validierung und Implementierung von Anforderungen entdeckt werden und korrigiert werden müssen.
- **Besseres Verständnis der Kunden/Benutzer des geplanten Systems**
Durch die Entwicklung der Anforderungen erhalten Kunden und Nutzer des Systems ein besseres Verständnis ihrer eigenen Anforderungen an das System
- **Technische Probleme, Terminprobleme und Kosten**
Bei der Implementierung einer Anforderung können technische, Zeit- und Kostenprobleme auftreten.
- **Änderung von Kundenprioritäten oder Kundenorganisation**
Wegen Änderung des Geschäftsfelds, Wechsel von Mitarbeitern, neuer Mitbewerber, ...
- **Änderung der Systemumgebung**
kann zu Änderung von Systemanforderungen führen

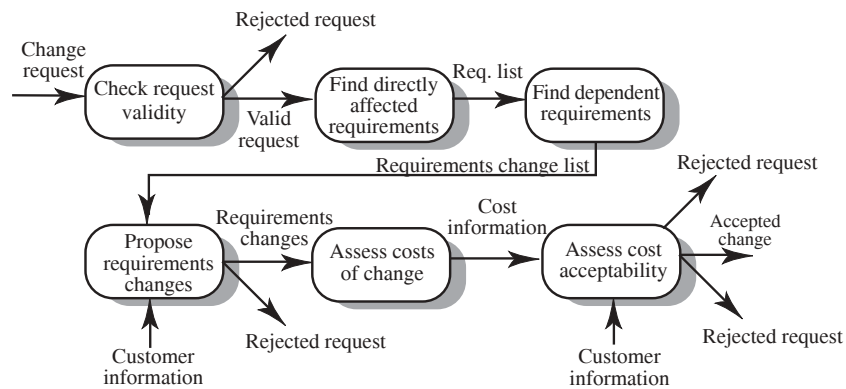
Aktivitäten des Änderungsmanagements



Vorgehensweise zur Änderung von Anforderungen

- **Identifizierung von Problemen und Vorschläge für Änderungen**
- **Analyse der Änderungsvorschläge**
umfasst folgende Schritte
 - Übersicht über Änderungsvorschläge
 - Für jeden einzelnen Änderungsvorschlag:
 - Überprüfung, ob Änderungsvorschlag berechtigt ist. Kunden können Anforderungen missverstehen und unnötige Änderungen vorschlagen.
 - Identifizierung der von der Änderung direkt betroffenen Anforderungen und der aufgrund von Abhängigkeiten indirekt betroffenen Anforderungen
 - Ausarbeitung des Änderungsvorschlags
 - Kostenschätzung für die Änderungen und Verhandlung mit dem Kunden, ob die Kosten für die vorgeschlagenen Änderungen akzeptierbar sind.
- **Implementierung der Anforderungsänderungen**
 - Erstellung eines Satzes von Änderungen des Anforderungsdokuments und Validierung der Änderungen .

Schematische Darstellung der Änderungsanalyse



© Kotonya, Sommerville

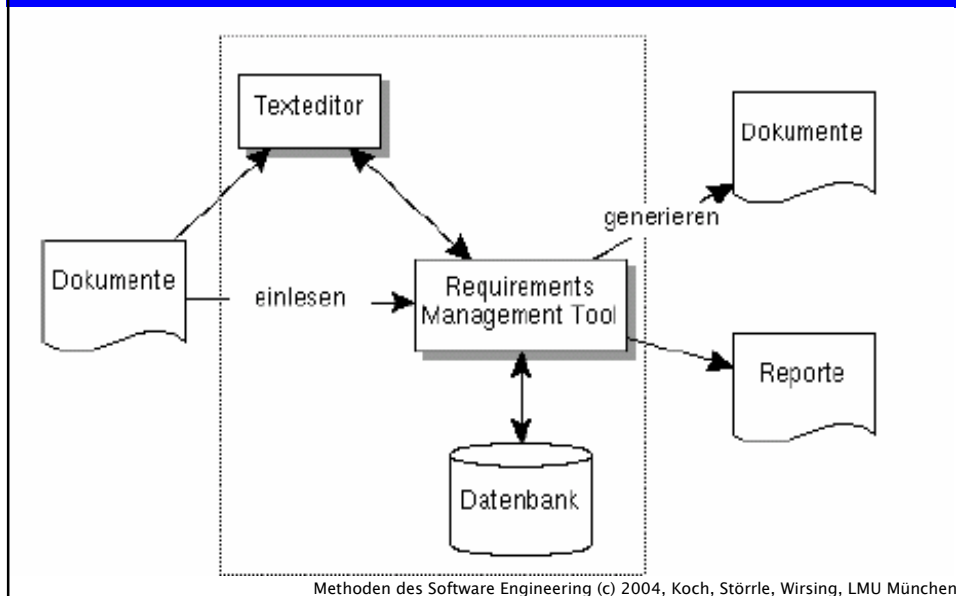
Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Aufgaben von Anforderungsmanagement- Werkzeugen

- **Verwalten aller Dokumente**
(z.B. Anforderungen, Modelle, Testpläne, 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**, beliebig konfigurierbar, z.B.:
 - Wo sind die Anforderungen realisiert?
 - Warum steht dieses Stück Code hier?

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

RE-Werkzeuge: Struktur



RE-Werkzeuge: Übersicht

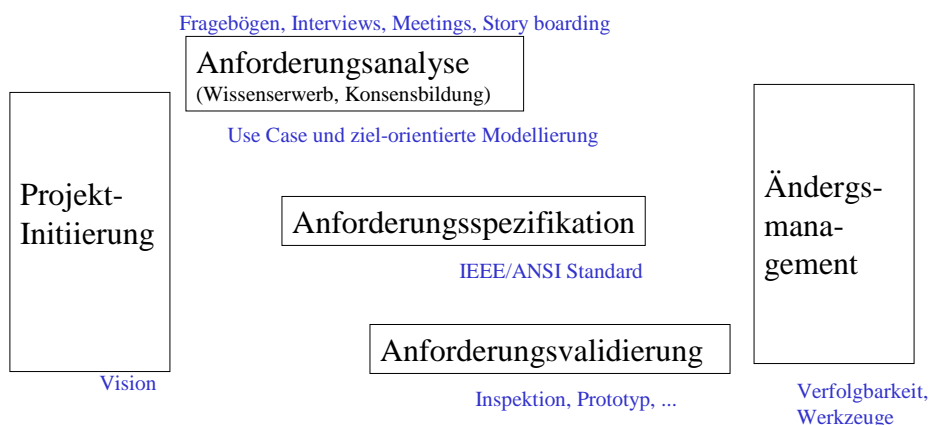
Werkzeug	Hersteller	Web-Adresse
CaliberRM	Starbase Cooperation	www.starbase.com
Core	Vitech Corp	www.vtcorp.com
Cradle	Structured Software Systems	www.3sl.co.uk
DOORS	Telelogic, AB	www.telelogic.de
DOORSrequireIT	Telelogic, AB	www.telelogic.de
RDT	IGATECH Corporation	www.igatech.com
RequisitePro	Rational Software Corporation	www.rational.com
RTM	Integrated Chipware, Inc.	www.chipware.com
SLATE	EDS	www.tdtech.com
Vital Link	Compliance Automation, Inc.	www.complianceautomation.com
CARE	SOPHIST Group	www.sophist.com

Vergleichende Händlerstudie: <http://www.incose.org/tools/tooltax.html>

Zusammenfassung: Anforderungsvalidierung

- Anforderungsvalidierung
 - Anforderungen sind schwierig zu validieren, da meist kein Referenzdokument vorhanden ist, gegen das geprüft werden kann
 - Techniken: Inspektion, Prototypen, Testen, Formale Techniken
 - Checklisten und perspektiven-basiertes Lesen helfen die Validierung von Anforderungen systematisieren, d.h. fehlende Erfahrung ausgleichen
- Anforderungsmanagement
 - Prozess der Verwaltung der Änderungen der Anforderungen muss geplant werden
 - Grundvoraussetzung: Verfolgbarkeit von Anforderungen
 - Werkzeugunterstützung unabdingbar

Zusammenfassung: Aufgaben des Requirements Engineering



Literatur

- Helmut Balzert: Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Heidelberg: Spektrum Akad. Verlag, 1997
- V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard und M.V. Zelkowitz: The Empirical Investigation of Perspective-Based Reading, Journal of Empirical Software Engineering 2, 1996, 133-164
- Tom Gilb, Dorothy Graham: Software Inspection. Addison-Wesley, 1993
- Gerald Kotonya, Ian Sommerville: Requirements Engineering – Processes and Techniques, Wiley 2000
- D. Leffingwell, D. Widrig: Managing Software Requirements – A Use Case Approach. Addison-Wesley, 2003