Willkommen zur Vorlesung
Softwarekonstruktion
im Wintersemester 2012 / 2013

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

Vorlesungswebseite (bitte notieren):

http://www-jj.cs.tu-dortmund.de/secse/pages/teaching/ws12-13/swk/index_de.shtml

Agenda

- Organisatorisches
- Vorstellung des Fachgebietes
- Vorlesungsinhalte

Organisatorisches – Agenda

Studienordnung:
- Einordnung / Kompetenzen / Struktur / Prüfung

Vorlesung:
- Bildungsvertrag, Termine, Feedback

Übung:
- Konzept / Termine

Klausur

Bachelor Informatik / Angewandte Informatik

- Wahlpflichtmodul
- Teilnahmevoraussetzungen
  - Erfolgreich abgeschlossenes Modul „Software-Technik“ (SWT)
  - Kenntnisse: Objektorientierung, Programmierpraxis

Studienordnung
vermittelte Kompetenzen

Erlangbare Kompetenzen innerhalb der Vorlesung:

- grundlegende Prinzipien der ingenieurmäßigen Konstruktion von Software
- ihre Vor- und Nachteile für eine gegebene Problemstellung und deren Anwendung auf Probleme mittlerer Größe
- formale Spezifikationssprachen zur Beschreibung der Architektur von Systemen
- Ansätze aus der Logik zur Überprüfung von Systementwürfen
- die Organisation großer Softwaresysteme unter Prüfung verschiedener Gesichtspunkte

Studienordnung
Struktur laut Modulhandbuch (Bachelor)

3 SWS:

- 2 SWS Vorlesung
- 1 SWS Übung

4 Credits

- 3 Credits Vorlesung
- 1 Credits Übung

Aufwand 120 Stunden

- 45 Stunden Präsenz (15*(2+1))
- 75 Stunden Vor-/Nachbereitung und Hausübungen (15*5)

Veranstaltungssprache Deutsch

Diplom

- Prüfungsleistung: Klausur

Bachelor

- Studienleistung: Übungsschein
- Modulprüfung: Klausur (bei bestandener Studienleistung)

Bildungsvertrag
Wir bieten ...

Fachliche Einführung in das Thema Softwarekonstruktion

Engagierte Betreuung

- Interessante Vorlesung
- Regelmäßige Sprechstunden (Termin s. Homepage)
- Betreute Übungen
- Korrigierte Hausübungen
- Transparente Anforderungen
- Möglichkeiten zum direkten Feedback

Möglichkeit zum Erwerb des Scheins

Bildungsvertrag
Wir erwarten ...

Aktives Auseinandersetzen mit den Vorlesungsinhalten

- Aktive Teilnahme an der Vorlesung
- Vor- und Nachbereitung der Vorlesung
- Aktive Teilnahme an den Übungen
- Bearbeitung der Hausübungen

Termine:

- Mo. 16:00 bis 17:40 Otto-Hahn-Str. 14 – E23
  10-min. Pause ca. um 16:45

Vorlesung
Feedback

Wir haben besonderes Interesse an vorlesungsbegleitendem Feedback, um etwaige Verbesserungsvorschläge ggf. schon während des Semesters zu berücksichtigen.

Übliche Kontaktmöglichkeiten:

- E-mail jan.jurjens@cs.tu-dortmund.de
- Tel.: 0231 755-7208
- Sprechstunde: Dienstags 12.00-13.00 Uhr
- Anonymes Kontaktformular: Link von Vorlesungswebseite

Darüberhinaus: Vorlesungsbegleitendes Feedback über Fragebögen in unregelmäßigen Abständen.

Termine (**14 tägig**):

- Do, 10:15 - 11:45, Start: 18.10.2012, MSW16 – E29
- Do, 14:15 - 15:45, Start: 18.10.2012, MSW16 - E29
- Fr, 10:15 - 11:45, Start: 18.10.2012, OH 14 – 304

Fragestunde (inhaltliches, Miguel Büscher):

- Mo, 10:15 – 11:00, Start: 22.10.2012, OH 14 - 340

Kontakt (organisatorisches):

- Bei Fragen zu den Übungen und ihrer Durchführung:
  - Sebastian Pape (Kontakt via Mail, Sprechstunde, Übungen)

Übung
Termine

VL-Woche	KW	Woche vom	Übung	Gruppe	Bemerkung
1	41	08.10.2012	-	-	
2	42	15.10.2012	1	1 - 3	
3	43	22.10.2012	1	4 - 6	
4	44	29.10.2012	2	1 - 3	Allerheiligen am 1. November
5	45	05.11.2012	2	4 - 6	
6	46	12.11.2012	3	1 - 3	
7	47	19.11.2012	3	4 - 6	
8	48	26.11.2012	4	1 - 3	
9	49	03.12.2012	4	4 - 6	
10	50	10.12.2012	5	1 - 3	
11	51	17.12.2012	5	4 - 6	
					Weihnachtspause
12	2	07.01.2013	6	1 - 3	
13	3	14.01.2013	6	4 - 6	
14	4	21.01.2013	7	1 - 3	
15	5	28.01.2013	7	4 - 6	

Anmeldung:

- Via AsSESS
- http://ess.cs.uni-dortmund.de/ASSESS/index.php?do=lecturelist

- Anmeldung nach der heutigen Vorlesung möglich
- Anmeldung bis 12.10.2012, 10:00 Uhr
- Verteilung mittels Algorithmus (Solver)
  (nicht priorisiert nach zeitlichem Eingang der Anmeldung)
- Verteilung wird am 15.10.2012 bekannt gegeben

Übung
Ablauf Präsenzübung

Übungsablauf

- Die Übung wird als Präsenzübung durchgeführt.
- Die Übungszettel werden jeweils 14-tägig veröffentlicht und sind dann Inhalt der nächsten Übung.
- Die Übungszettel werden während der Übung alleine oder in Gruppen bearbeitet.
- Der anwesende Tutor steht für Fragen zur Verfügung.
- Am Ende der Übung werden **von den Studierenden** Lösungen vorgeschlagen und die Aufgaben besprochen.
- Ein Lösungsvorschlag zur Präsenzübung wird dann über unsere Webseite veröffentlicht.

Übung
Hausübungen I

Übungskonzept:

- Insgesamt werden 7 Übungszettel veröffentlicht.
- 6 davon enthalten eine Hausübung.
- Hausübungen sollen bis zum entsprechenden Termin gelöst und abgegeben werden. (Mehr dazu gleich.)
- Abgabe in den Übungen oder durch Einwurf in den Briefkasten im Gebäude in der OH 20
- Keine Abgabe per Mail

Übungskonzept:

- Die Aufgaben sollen in Gruppen bearbeitet werden.
- Inhaltliche und konzeptionelle Arbeit auch gerne in größeren Gruppen.
- Ausarbeitung und Abgabe der Arbeit aber nur in Gruppen von min. 2 und max. 3 Studierenden. Die Zusammenarbeit ist entsprechend auf den Abgaben zu vermerken.
- Bei Abgabe von Duplikaten erhält keine der Gruppen Punkte.
- Die Abgaben werden korrigiert und die Gruppe erhält die korrigierte Lösung in der Übungsgruppe zurück.

  $\Longrightarrow$ Gruppennummer auf die Abgabe schreiben

Übung
Leistungsnachweis

Übungskonzept:

- Bei jeder Hausübung gibt es 10 Punkte.
- Diplom-Studierende nach DPO 2001
  - erhalten einen unbenoteten Schein durch erfolgreiche Teilnahme an der Abschlussklausur.
  - Teilnahme an den Übungen und Hausübungen ist freiwillig.
- Bachelor-Studierende
  - benötigen für die Zulassung zur Klausur einen Leistungs-nachweis über die erfolgreiche Teilnahme an den Übungen
  - 50% der Punkte aus den Hausübungen (30 von 60)
  - aber mindestens jeweils 30% der Punkte aus den Aufgaben 1+2, 3+4 und 5+6 (jeweils 6 von 20)

Übung
Hausübungen – Zeitschema

Sonntag	Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag
14 Okt	15	16	17	18 Übung W1	19 Übung W1	20
21	22	23	24	25 Übung W2	26 Übung W2	27
28	29	30	31 Okt	1 Nov	2	3
4	5	6	7	8	9 Abgabe-ende	10 Nov

Übung
Hausübungen – Termine

Übungs-Nr.	Ausgabe	Abgabe
1	18.10.2012	09.11.2012, 11:30 Uhr
2	02.11.2012	23.11.2012, 11:30 Uhr
3	15.11.2012	07.12.2012, 11:30 Uhr
4	29.11.2012	21.12.2012, 11:30 Uhr
5	13.12.2011	18.01.2013, 11:30 Uhr
6	10.01.2013	**25.01.2013, 11:30 Uhr**

Inpud-Forum

Ziel: Diskussion der Studierenden untereinander

Keine Kommunikation mit den Veranstaltern dort

- Keine garantierten Antwortzeiten
- Für dringendes: Mail oder Sprechstunde nutzen

Organisatorische + inhaltliche FAQ

- Für Fragen von Studierenden, die auch für andere interessant sein könnten

Moderation durch die Veranstalter

Prüfung: Klausur

- schriftlich
- 60 Minuten

Klausurtermine:

- Dienstag, 12.02.2013, 08:30-10:00 Uhr
  HS 1, HS 2 (Hörsaalgebäude II), Emil-Figge-Str. 50
- Dienstag, 19.03.2013 08:30-10:00 Uhr
  HS 1 (Hörsaalgebäude II), Emil-Figge-Str. 50

Bachelor Informatik / Angewandte Informatik:

- Die Prüfungsleistung wird anhand der Modulprüfung in Form einer schriftlichen Prüfung ermittelt.
- Die Bearbeitung der Übungsaufgaben ist Voraussetzung zur Teilnahme an der Modulprüfung.

Kontakt

Jan Jürjens:

- http://www-jj.cs.tu-dortmund.de/secse/pages/people/juerjens/index_de.shtml

Sebastian Pape:

- http://ls14-www.cs.tu-dortmund.de/pape

Vorlesungsseite:

- http://www-jj.cs.tu-dortmund.de/secse/pages/teaching/ws12-13/swk/index_de.shtml

Inpud-Forum:

- Einrichtung ist beantragt.

Übungsanmeldung

- http://ess.cs.uni-dortmund.de/ASSESS

Vorstellung des Fachgebietes

- Lehrangebot
- Forschung
- Abschlussarbeiten und Hiwi-Jobs

Wer bin ich?

- Professor für Software Engineering an der TU Dortmund
- Wissenschaftskoordinator „Enterprise Engineering“ am Fraunhofer ISST
- Leiter der Fraunhofer-Attract-Projektgruppe „Architectures for Auditable Business Process Execution (Apex)“

Vorher u.a.:

- Royal Society Industrial Fellow bei Microsoft Research Cambridge
- Research Fellow am Robinson College (Univ. Cambridge)
- Postdoc an der TU München
- Promotion zu „Principles for Secure Systems Design“ (Univ. Oxford)
- Forschungsaufenthalte am LFCS (Univ. Edinburgh) und Bell Labs (Palo Alto)
- Studium an Univ. Bremen und Univ. Cambridge

Wer ist meine Forschungsgruppe?

- Misha Aizatulin (Microsoft Research Cambridge)
- H. Selcuk Beyhan (Logica (Germany))
- Francois Dupressoir (Microsoft Research Cambridge)
- Michael Giddings (Open University)
- Thorsten Humberg (Fraunhofer ISST)
- Christopher McLaughlin (Gartner)
- **Sebastian Pape (TUD) => Übungsbetreuung**
- Dr. Thomas Ruhroth (TUD)
- Andreas Schmitz (Fraunhofer ISST)
- Stefan Taubenberger (Münchener Rückversicherung)
- Daniel Warzecha (Fraunhofer ISST)
- Dr. Sven Wenzel (TUD)
- Christian Wessel (TUD)

Vertrauenswürdige IT-Systeme

IT Systeme durchziehen heute fast alle Funktionen in Wirtschaft und Gesellschaft. IT hat direkten (oft invasiven) Einfluss auf fast alle Aspekte menschlichen Lebens.

Die Erwartungen an die Vertrauenswürdigkeit dieser Systeme sind daher in den letzten 10 Jahren stark gestiegen. Diese Erwartungen werden oft nicht erfüllt. Teil des Problems ist, dass die bislang verwendeten System- und Software-Entwicklungsmethoden mit den gestiegenen Erwartungen bei gleichzeitig steigender Systemkomplexität nicht mithalten konnten.

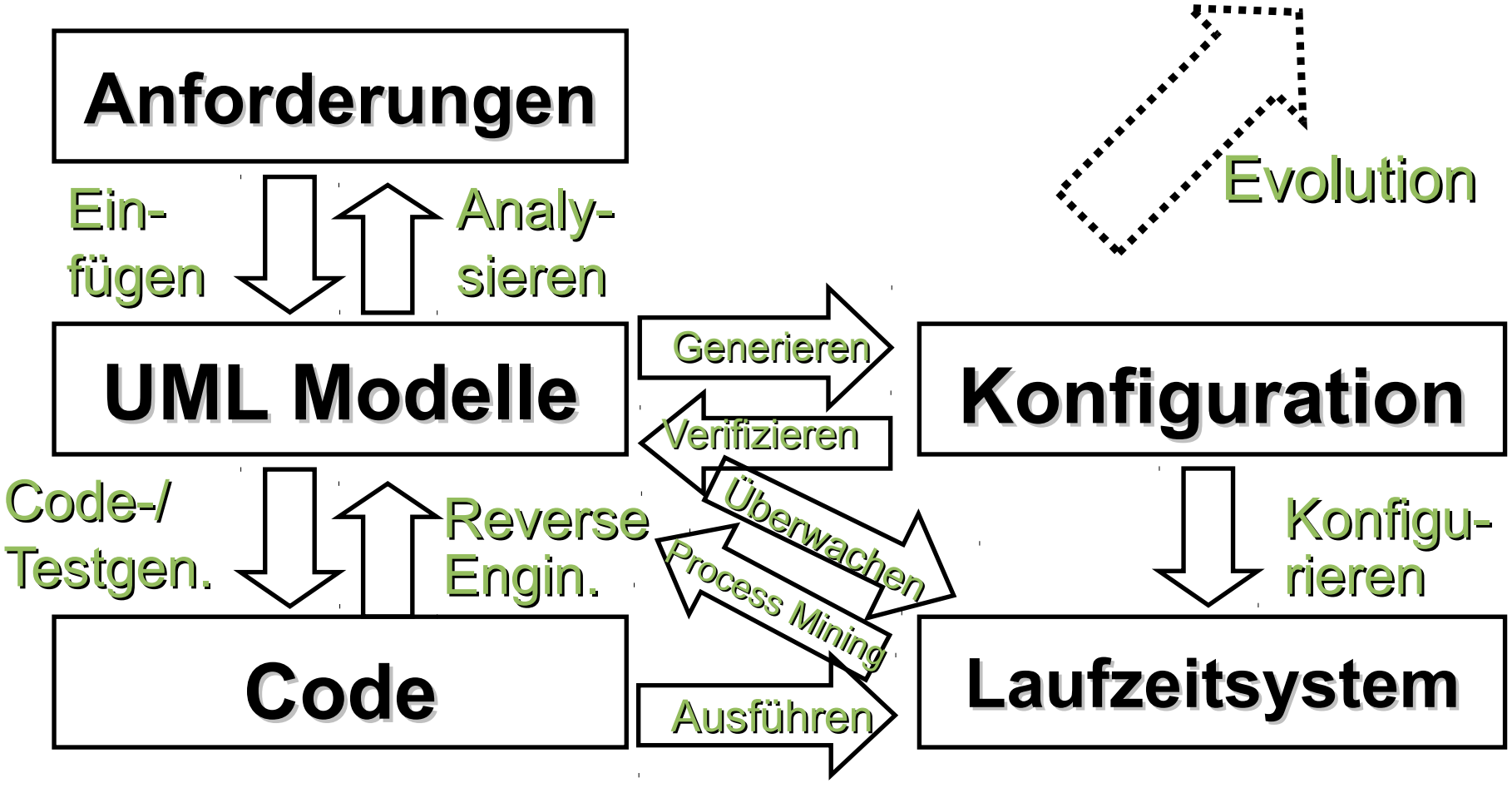
Offene Systeme

Aus Flexibiliäts- und Kostengründen sind moderne IT Systeme meist über offene Infrastrukturen realisiert, zum Beispiel:

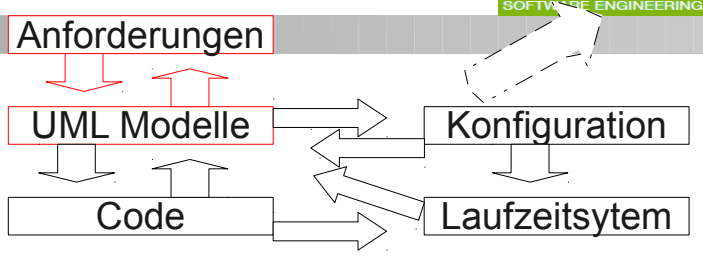
- Internet
- Mobile Netze

Aufgrund ihrer Offenheit sind sie dem Zugriff von Personen ausgesetzt, die in verschiedenem Maße vertrauenswürdig sind. Dieser Zugriff muss daher systemseitig reguliert werden. Aus Flexibiliäts- und Kostengründen wird dies oft auf der Softwareebene gelöst. Eine vertrauenswürdige IT braucht also sichere Software.

Modellbasierte Entwicklung



Modellierung mit UMLsec

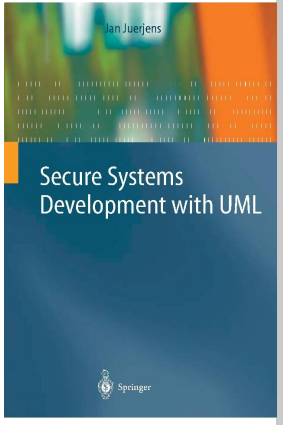


Ziel:

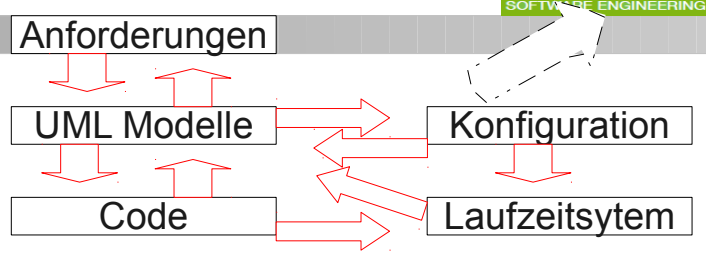
- Dokumentation und automatische Analyse von sicherheits-relevanten Informationen (z.B. Sicherheits-Eigenschaften und -Anforderungen) als Teil der Systemspezifikation.

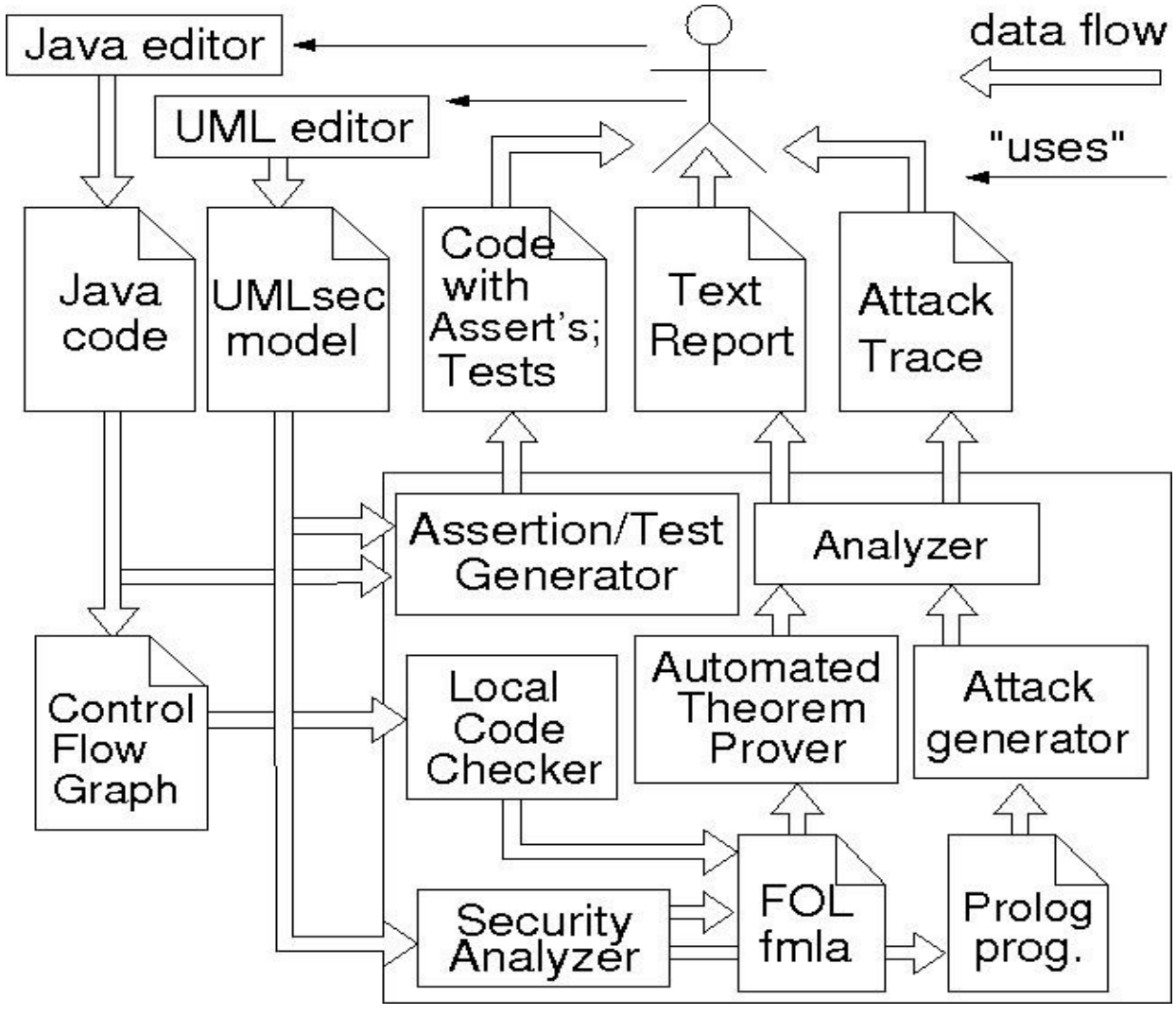
Idee:

- UML für System-Modellierung.
- Sicherheitsrelevante Informationen als Markierungen (Stereotypen) einfügen. Definiere dazu UML-Erweiterung UMLsec.
- Formale Semantik mit stromverarbeitenden Funktionen als Grundlage für Verifikation.

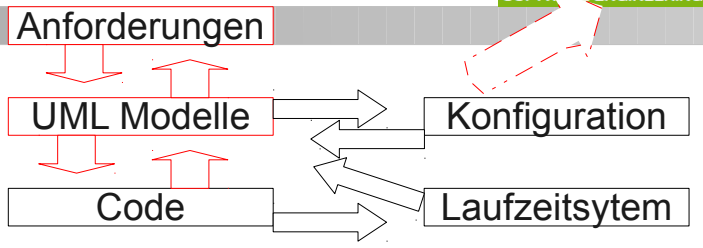


Werkzeugunterstützung

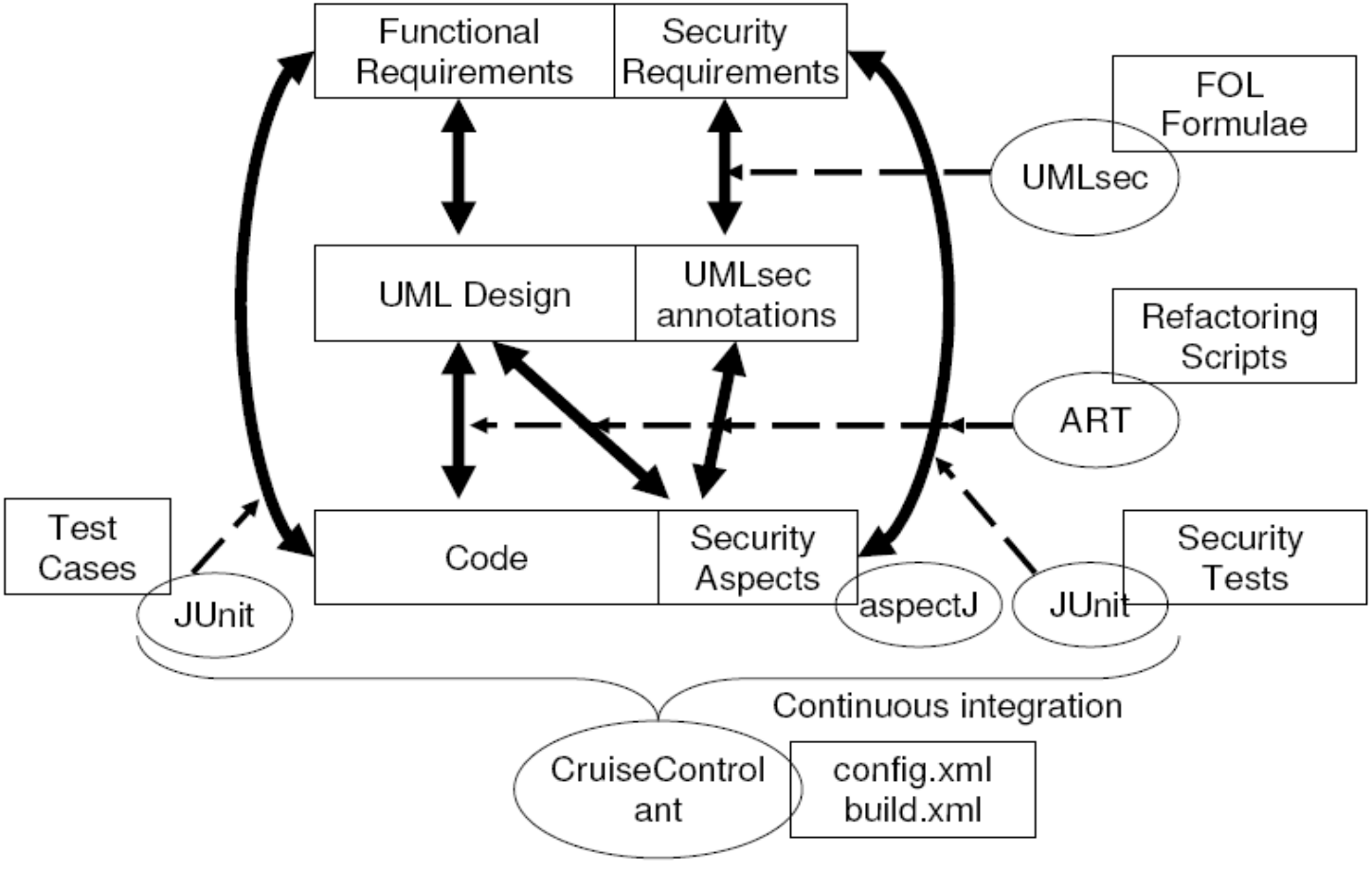




Sichere Evolution: Werkzeugunterstützung



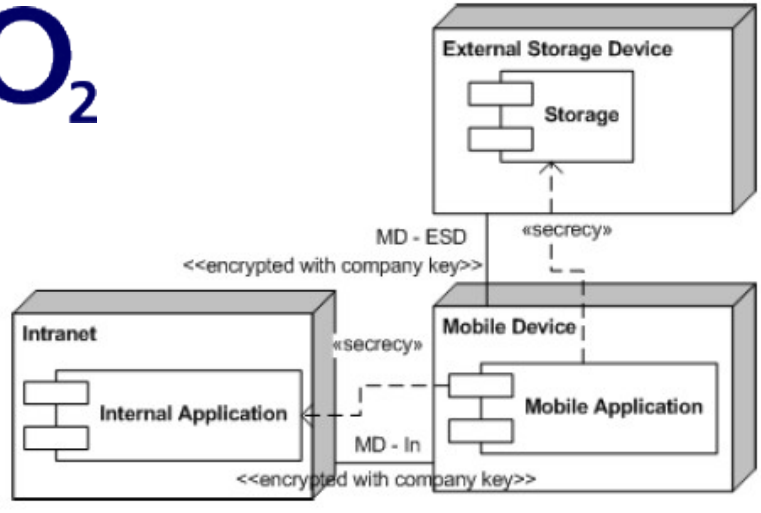
Nachverfolgbarkeit von Anforderungen vs. Implementierung bei Evolution durch Refactoring bewahren.



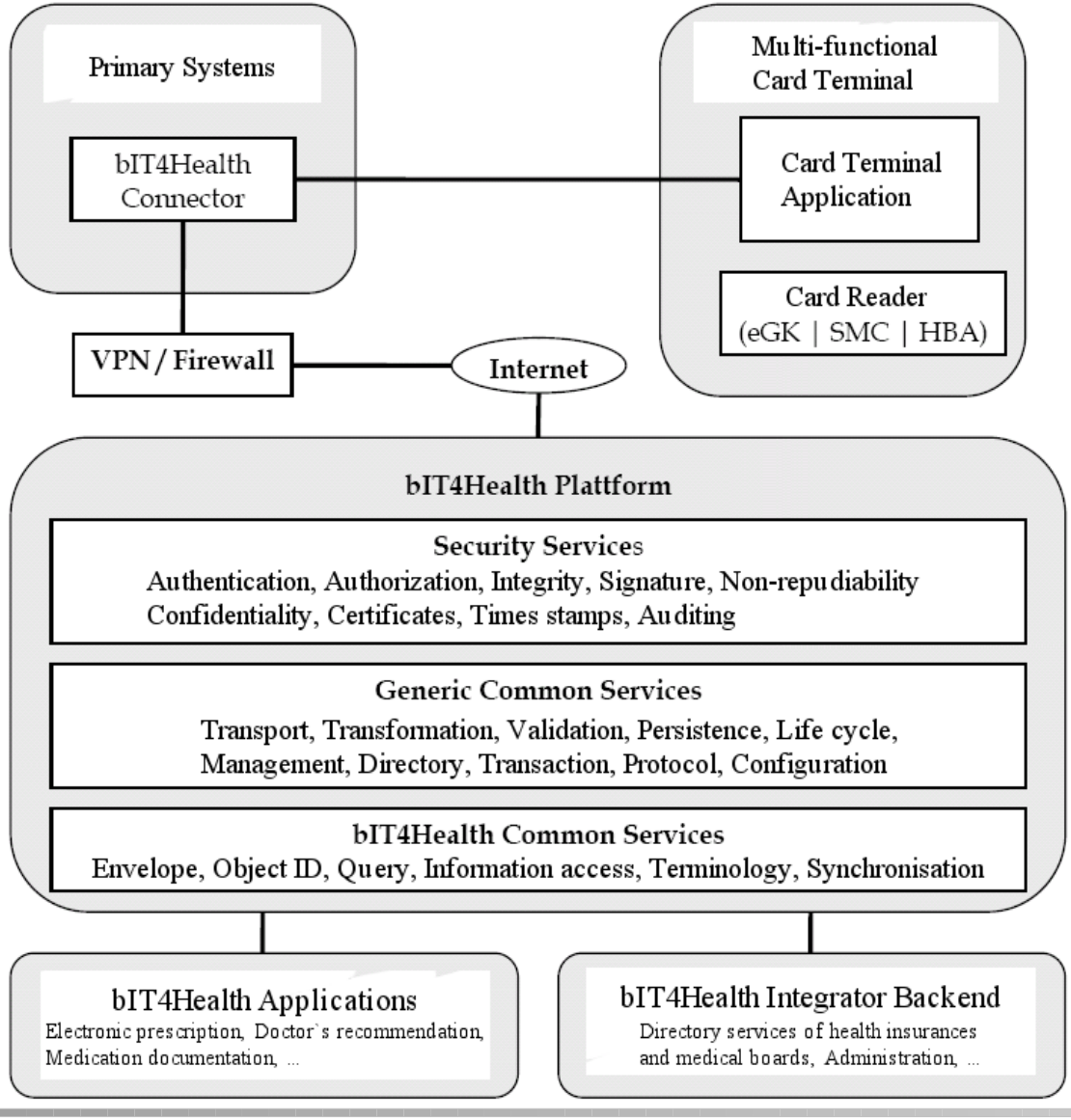
Mobile Informationssysteme bei O_2

- Anwendung von UMLsec auf mobile Kommunikations-Architekturen bei O_2 (Germany).
- Alle 62 Sicherheitsanforderungen aus der Security Policy erfolgreich verifiziert.
- Modellbasierte Techniken bringen Zusatzaufwand.
- Macht sich bezahlt bei wichtigen Sicherheitsanforderungen und Konzentration auf kritische Architekturanteile (auch im Vergleich mit anderen Qualitätssicherungs-Ansätzen mit vergleichbarer Verlässlichkeit)
- UMLsec adäquat für mobile Architekturen.

O_2



- Architektur mit UMLsec analysiert.
- Einige Schwachstellen aufgedeckt (fehlender Vertraulichkeitsschutz für digitale Rezepte).



Bank-Informationssystem bei der HypoVereinsbank

Modellbasierte Sicherheitsanalyse von webbasierter Bankanwendung ("digitaler Formularschrank").

Geschichtete Architektur (SSL Protokoll, darauf Client Authentisierungs-Protokoll)

Anforderungen:

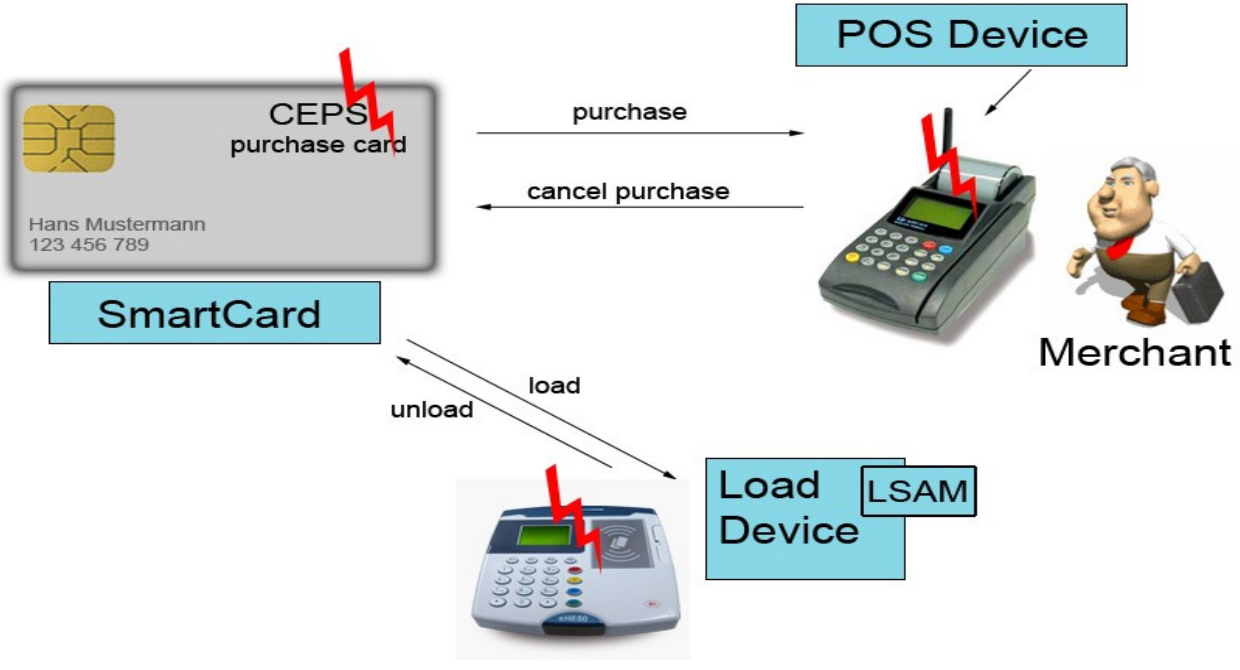
- Vertraulichkeit
- Authentisierung

Mobiles Bezahlsystem

Common Electronic Purse Specifications:
Globaler Standard für e-Geldbörsen (Visa et al.).

Smartcard enthält Kontostand, sichert Transaktionen mithilfe Krypto.

Formale Analyse von Load und Purchase Protokollen: signifikante Schwachstellen: Kauf-Umleitung, Betrug Ladegerätbetreiber vs. Bank.



Biometrische Authentisierung

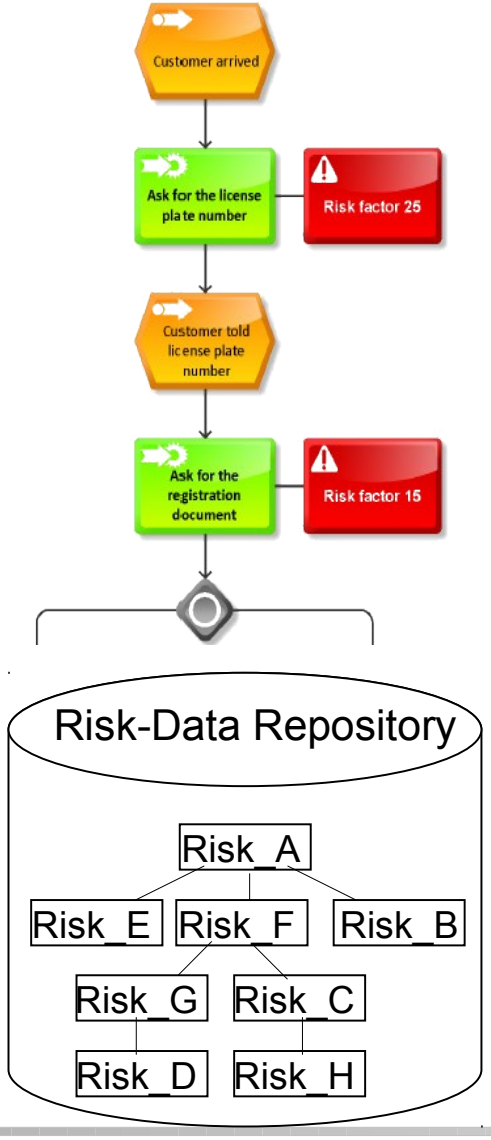
Smartcard-basiertes System.

Analysiert mit UMLsec parallel zur Entwicklung durch Firma in gemeinsamem Projekt.

Entdeckten drei signifikante Schwachstellen in verschiedenen Versionen (Fehlbedienungszähler umgangen durch Löschen / Wiederholen von Nachrichten; Smartcard unzureichend authentisiert durch Mischen von Sitzungen).

Endgültig entwickelte Version sicher.

- Idee: Automatische Analyse von Geschäftsprozessmodellen auf operationale Risiken, z.B. Gegenüber Benutzerberechtigungen zur Laufzeit, sowie der Benutzer-berechtigungen gegenüber der Sicherheitspolitik.
- Automatische Risiko-Identifikation und -Bewertung.
- Laufendes Projekt (Fraunhofer Attract): Architekturen für auditierbare Geschäftsprozessausführung (Apex).



Internes Informationssystem bei BMW

MetaSearch Engine: Personalisierte Suche im Firmen-Intranet (passwort-geschützt).

BMW Group

Einige Dokumente sehr sicherheitskritisch.

Über 1.000 potentielle Benutzer, 280.000 Dokumente, 20.000 Anfragen pro Tag.

Nahtlos in unternehmensweite Sicherheitsarchitektur integriert. Bietet Sicherheitsdienste für Anwendungen (Benutzerauthentisierung, rollenbasierte Zugangskontrolle, globales Single-Sign-On), Ansatzpunkte für weitere Sicherheitsdienste.

Erfolgreich mit UMLsec analysiert.

Hiwi-Tätigkeiten

Es gibt verschiedene Möglichkeiten für eine Beschäftigung als Hiwi am Fraunhofer ISST oder am LS 14 / TUD:

- Unterstützung der folgenden Projekte (bespielsweise durch Java-Programmierung eines UML-Analyse Werkzeuges oder konzeptuelle Arbeiten im Bereich modell-basierte Sicherheitsanalyse): "Architectures for Auditable Business Process Execution (APEX)", „SecureClouds“, „ClouDAT“
- Unterstützung in der Lehre (Tutorien, Folienerstellung etc)

Informationen unter:

http://www-jj.cs.tu-dortmund.de/secse/pages/home/jobs_de.shtml

Abschlussarbeiten (z.B. Bachelor)

Abschlussarbeiten können in inhaltlicher Beziehung zu einer Hiwi-Tätigkeit am Fraunhofer ISST oder LS 14 / TUD durchgeführt werden (oder auch unabhängig davon).

Sie können insbesondere in Zusammenhang mit Anwendungsprojekten am ISST durchgeführt werden, wodurch sich vielfältige Möglichkeiten zu Kooperation mit Unternehmen ergeben, zB:

- Apex: Versicherungen / Banken (Münchener Rückversicherung, Signal Iduna, Wüstenrot), Softwarehersteller (SAP, IDS Scheer)
- Secure Clouds / ClouDAT: Cloud-Software-Anbieter (LinogistiX), IT-Berater (Admeritia, ITESYS, TÜV-IT)

Informationen unter:

http://www-jj.cs.tu-dortmund.de/secse/pages/teaching/thesis/index_de.shtml

Themenvorstellung am 5. November im Rahmen dieser Vorlesung.

Einige Beispiel-Themen für Abschlussarbeiten

- Formale Abbildung von regulatorischer Compliance auf Security Policies
- Modellierung und Automatische Sicherheits-Analyse für Cloud Computing Systems
- Business Process Mining
- Spezifikation von IT-Sicherheitszielen für die Geschäftsprozessmodellierung und deren Integration in die Ausführung im Workflow
- Design und Entwicklung einer Schnittstelle zwischen der Business Prozess Management Suite ARIS und dem Sicherheitsanalysetool UMLsec zur Compliance Analyse in der Versicherungsdomäne
- Generierung von Geschäftprozessen mit OpenArchitectureWare unter Berücksichtigung von Sicherheitseigenschaften
- Werkzeuggestützte Modell-basierte Sicherheitsanalyse
- Werkzeugunterstützte Analyse von sicherheitskritischen SAP-Berechtigungen im Finanzbereich
- Modell-basiertes Return on Security Investment (ROSI) im IT-Sicherheitsmanagement

Weitere relevante Lehrveranstaltungen

Dieses Semester:

- Seminar „Ausgewählte Themen des Modell-basierten Sicherheits-Engineerings“. http://www-jj.cs.tu-dortmund.de/secse/pages/teaching/ws12-13/mbse-sem/index_de.shtml
- Vorlesung „Sicherheit: Fragen und Lösungsansätze“. http://www-jj.cs.tu-dortmund.de/secse/pages/teaching/ws12-13/sfl/index_de.shtml

SS 2013:

- Fachprojekt „Softwaretechniken für sichere Cloud-Computing-Systeme (4 SWS)“
- Methodische Grundlagen des Software Engineering (Master-Basismodul Software) (4+2 SWS)
- Seminar „Ausgewählte Themen des Modell-basierten Sicherheits-Engineerings“.

Zuordnung der Wahlveranstaltungen zu Schwerpunktgebieten (Diplom):

- Sicherheit und Verifikation
- Software-Konstruktion

Forschungsbereich Master: Software, Sicherheit und Verifikation

Informationen unter:
http://www-jj.cs.tu-dortmund.de/secse/pages/teaching/index_de.shtml

Und danach ?

[http://www.fraunhofer.de/de/presse/presseinformationen/2012/april/ertraege-aus-der-wirtschaft.html]

“Die Fraunhofer-Gesellschaft ist auch im Jahr 2011 weiter gewachsen. Das Finanzvolumen stieg um 12 Prozent auf 1,85 Milliarden Euro an.

Im Vorjahr hat Fraunhofer 1300 neue Beschäftigte eingestellt. Damit stieg die Zahl der Mitarbeiterinnen und Mitarbeiter auf mehr als 20 000 an. »Um die wachsende Anzahl an Forschungsprojekten und das steigende Auftragsvolumen bearbeiten zu können, benötigen wir auch künftig weitere neue qualifizierte Mitarbeiterinnen und Mitarbeiter«, betont der Personalvorstand der Fraunhofer-Gesellschaft.

Fraunhofer ist ein beliebter Arbeitgeber. Das hat die Mitarbeiter-Befragung im Vorjahr ergeben. 86 Prozent der Mitarbeiterinnen und Mitarbeiter sind stolz darauf, bei Fraunhofer zu arbeiten. Im Durchschnitt sagen das in Deutschland nur 60 Prozent über ihren Arbeitgeber.“

http://www.randstad-award.de/randstad-award-deutschland/presse/news/news/items/349.html

“Der Randstad Award für den attraktivsten Arbeitgeber geht in diesem Jahr an die Fraunhofer-Gesellschaft. Auf Platz 2 schaffte es EADS, dicht gefolgt von BMW auf Platz 3.“

Und: Promotion projekt-begleitend möglich.

Kontakt: http://jan.jurjens.de

Ziele dieser Vorlesung

- Kenntnis der grundsätzlichen verschiedenen Dimensionen des Software Engineerings (technisch, organisatorisch, psychologisch)
- Vermittlung eines Überblicks über das Spektrum gängiger Konzepte zur Spezifikation und zum Testen
- Erläuterung ausgewählter Methoden und Sprachen zur Spezifikation
- Einbettung dieser Spezifikationskonzepte in die Softwaretechnik
- Modellbasiertes Software-Engineering
- Erläuterung verschiedener Testtechniken
- Einbettung in den gesamten Softwareprozess

Begriffsbildung Software Engineering

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines. [NR68]

Software Engineering ist die Anwendung wissenschaftlicher Erkenntnisse mit dem Ziel, Computer mittels Programmen, Verfahren und zugehörigen Dokumenten dem Menschen nutzbar zu machen [Den91]

... [is] the systematic approach to the development, operation, maintenance, and retirement of software.“ [IEEE83]

Begriffsbildung Software Engineering

A software engineer must of course be a good programmer, be well-versed in data structures and algorithms, and be fluent in one or more programming languages. ... The software engineer must be familiar with several design approaches, be able to translate vague requirements and desires into precise specifications, and be able to converse with the user of a system in terms of the application rather than in ‚computerese'. [GJM91]

Daraus folgende, notwendige Fähigkeiten:

- Kommunikation auf verschiedenen Abstraktionsebenen
- Erstellung und Verwendung von Modellen / Spezifikationen
- Kommunikation mit Personen mit unterschiedlichen Zielsetzungen, Vorstellungen, Ausbildungen
- Arbeitsplanung und -koordination

Defizite des Software Engineerings

Phänomen der Software-Entwicklung:
Trotz der Abhängigkeit von Software gibt es

- Keine zuverlässige Herstellung von Software im industriellen Maßstab
- Kosten- und Terminüberschreitungen
- Bei Auslieferung: ungenügende Softwarereife
- Keine Produktivitätskontrolle wie in anderen industriellen Fertigungsbereichen
- Keine Qualitätskontrolle wie in anderen industriellen Fertigungsbereichen, gerade das Testen ist immer noch unterbewertet.

Herausforderungen

Ursachen der Phänomene (nach [CKI88]):

- Unzureichende Verbreitung von Anwendungsdomänen-Wissen
  - Projekte in neuer Branche
  - Technologiezentriertheit
- Ändernde und sich widersprechende Anforderungen
  - Markt
  - verschiedene Kunden
  - Erkenntnisprozesse
  - Missverständnisse
- Kommunikations- und Koordinierungs-Pannen
  - unterschiedliche Vorstellungen und Zielsetzungen
  - Inkongruenzen zwischen Kompetenz und Verantwortung
  - Kapitulation

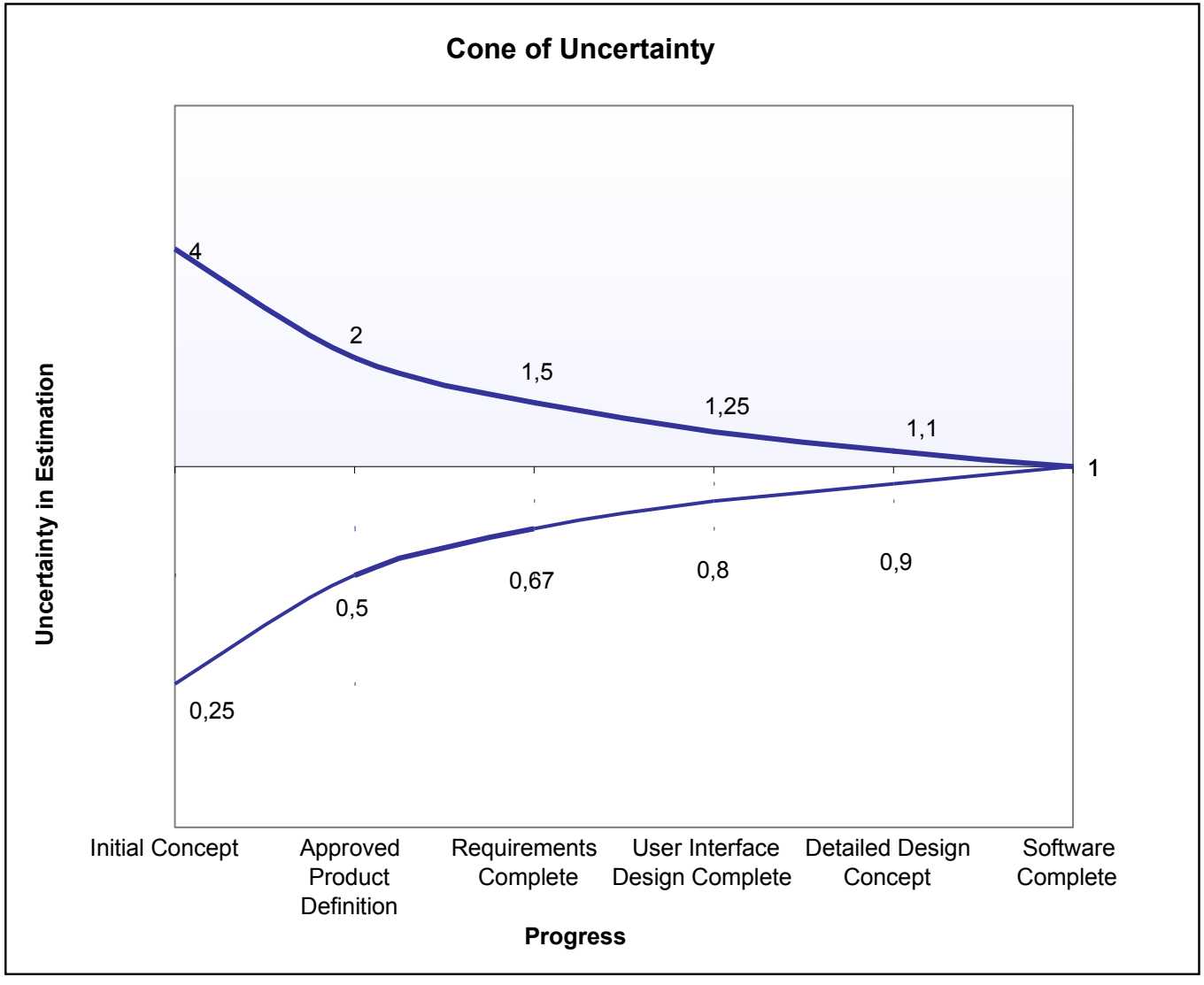
Phänomen I: Abbruchrate

Anzahl Function	Früher als geplant	Termingerecht	Verspätet	Abgebrochen
1 FP	14,86%	83,16%	1,92%	0,25%
10 FP	11,08%	81,25%	5,67%	2,00%
100 FP	6,06%	74,77%	11,83%	7,33%
1.000 FP	1,24%	60,76%	17,67%	20,33%
10.000 FP	0,14%	28,03%	23,83%	48,00%
100.000 FP	0,00%	13,67%	21,33%	65,00%
Durchschnitt	5,53%	56,94%	13,71%	23,82%

Abbruchrate großer Projekte nach [Jon96]

1 Function Point (FP) sind im Schnitt 55 Lines of Code (LOC) in C++/Java, 20 LOC Perl, 13 LOC SQL, etc.
Desktop-Projekt mit 1 FP dauert ca. eine Woche mit einem Entwickler
Allgemeines Projekt mit 100.000 FP und 100 Entwicklern: ca. 17 Monate
(inkl. Planung, Entwurf, Test etc.; ohne Urlaub, Krankheiten etc.)
Formel z.B. für Desktop Projekt: StaffMonth=0,157 * FunktionPoints^0,591 * MaximumTeamSize^0,810

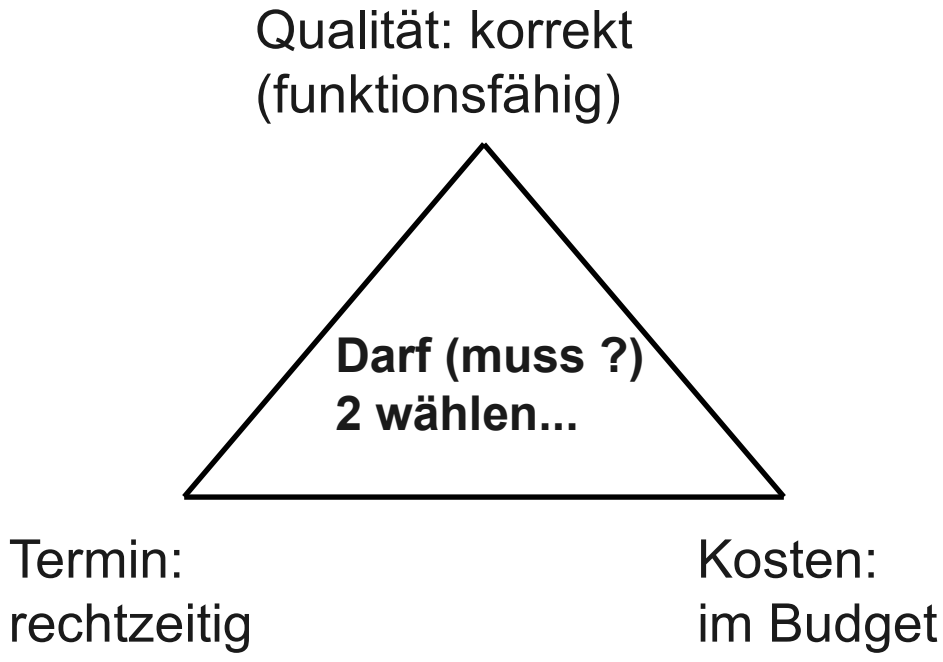
Unsichere Aufwandschätzung



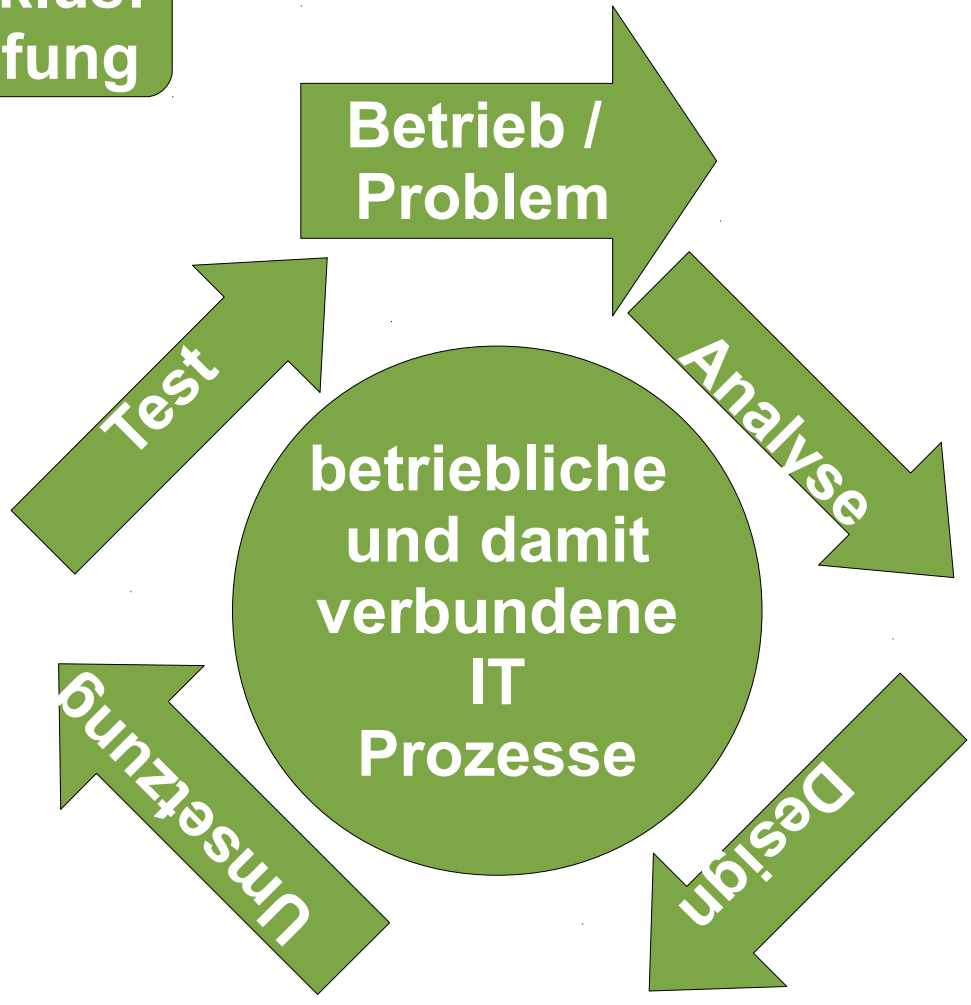
Nach: Steve McConnell: Software Estimation, Microsoft Press, Redmond, Washington, 2006.

Interessenkonflikte

Phänomen II: Termine, Kosten, Qualität

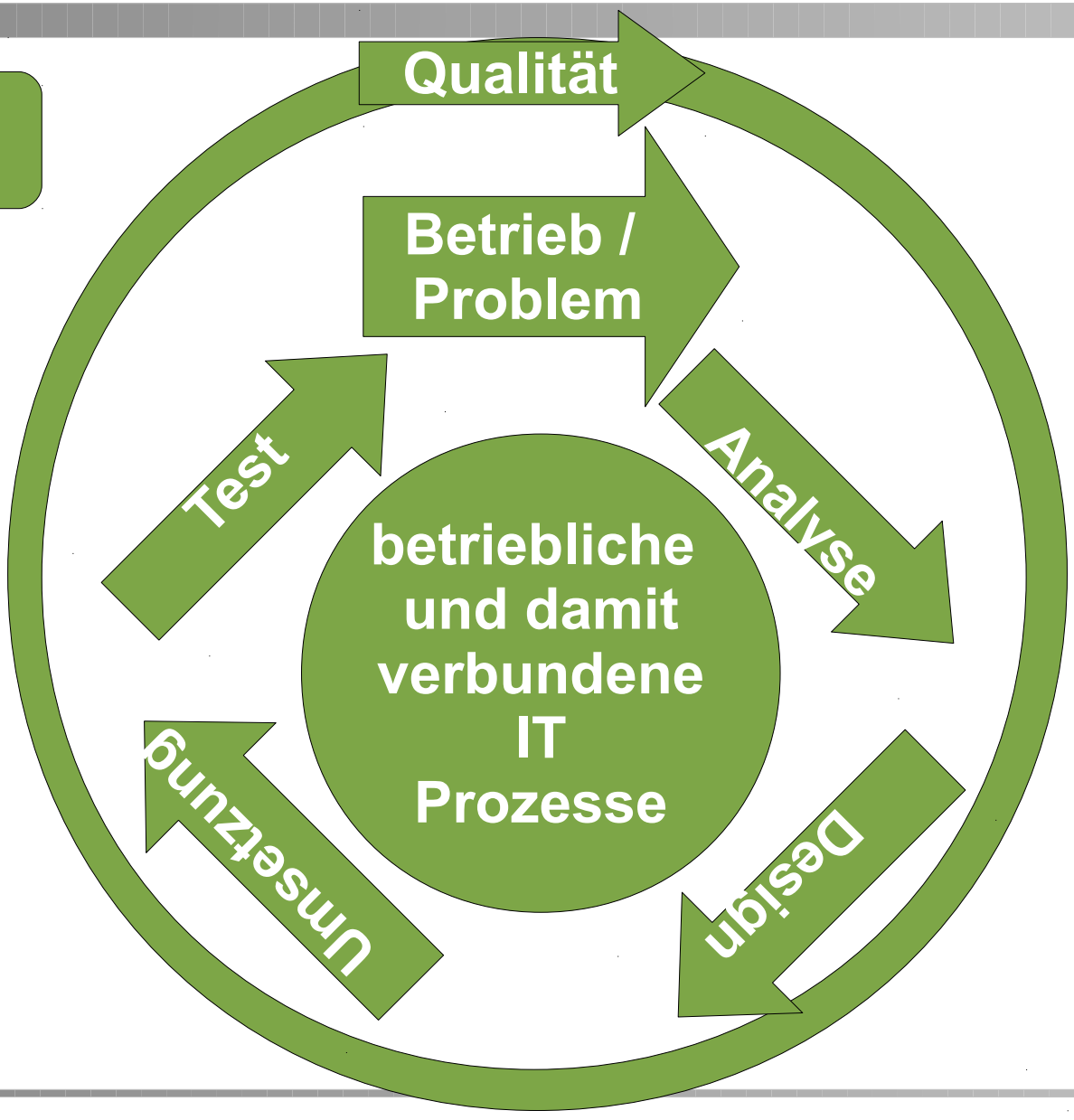


Der SE Lebenszyklus: Punktuelle Vertiefung



Vorlesungsüberblick
Inhaltlicher Zusammenhang

Umfassend: Qualität



Themen der Vorlesung

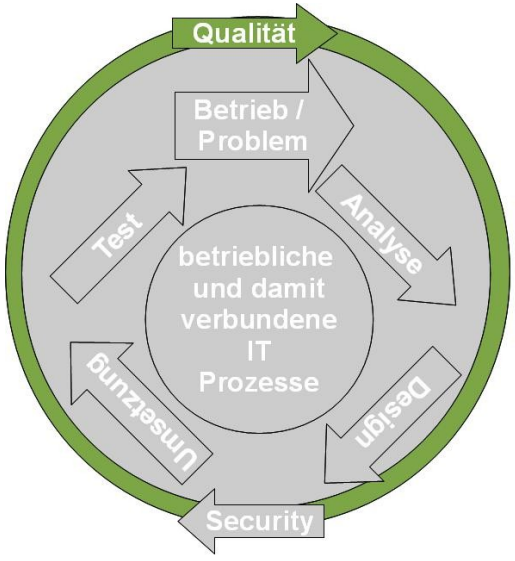
Kapitel 1: Qualitätsmanagement
* Teil 1.0 (Qualität: Grundlagen)
* Teil 1.1 (Prozess-Qualität)
* Teil 1.2 (Softwaremetriken)

Kapitel 2: Testen
* Teil 2.0 (Einführung Testen)
* Teil 2.1 (Grundlagen des Softwaretestens)
* Teil 2.2 (Testen im Softwarelebenszyklus)
* Teil 2.3 (Statischer Test)
* Teil 2.4 (Black Box Test)
* Teil 2.5 (White Box Test)
* Teil 2.6 (Testmanagement)
* Teil 2.7 (Testwerkzeuge)

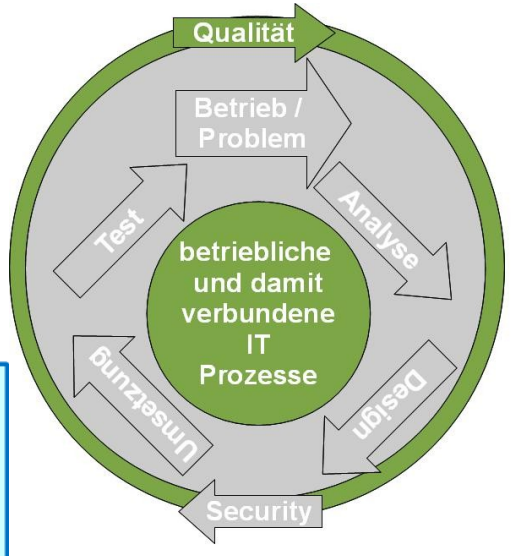
Kapitel 3: Modellbasierte Softwareentwicklung
* Teil 3.0 (Modellbasierte Softwareentwicklung)
* Teil 3.1 (OCL)
* Teil 3.2 (Algebraische Spezifikation als Grundlage für Modellbasierte Softwareentwicklung)

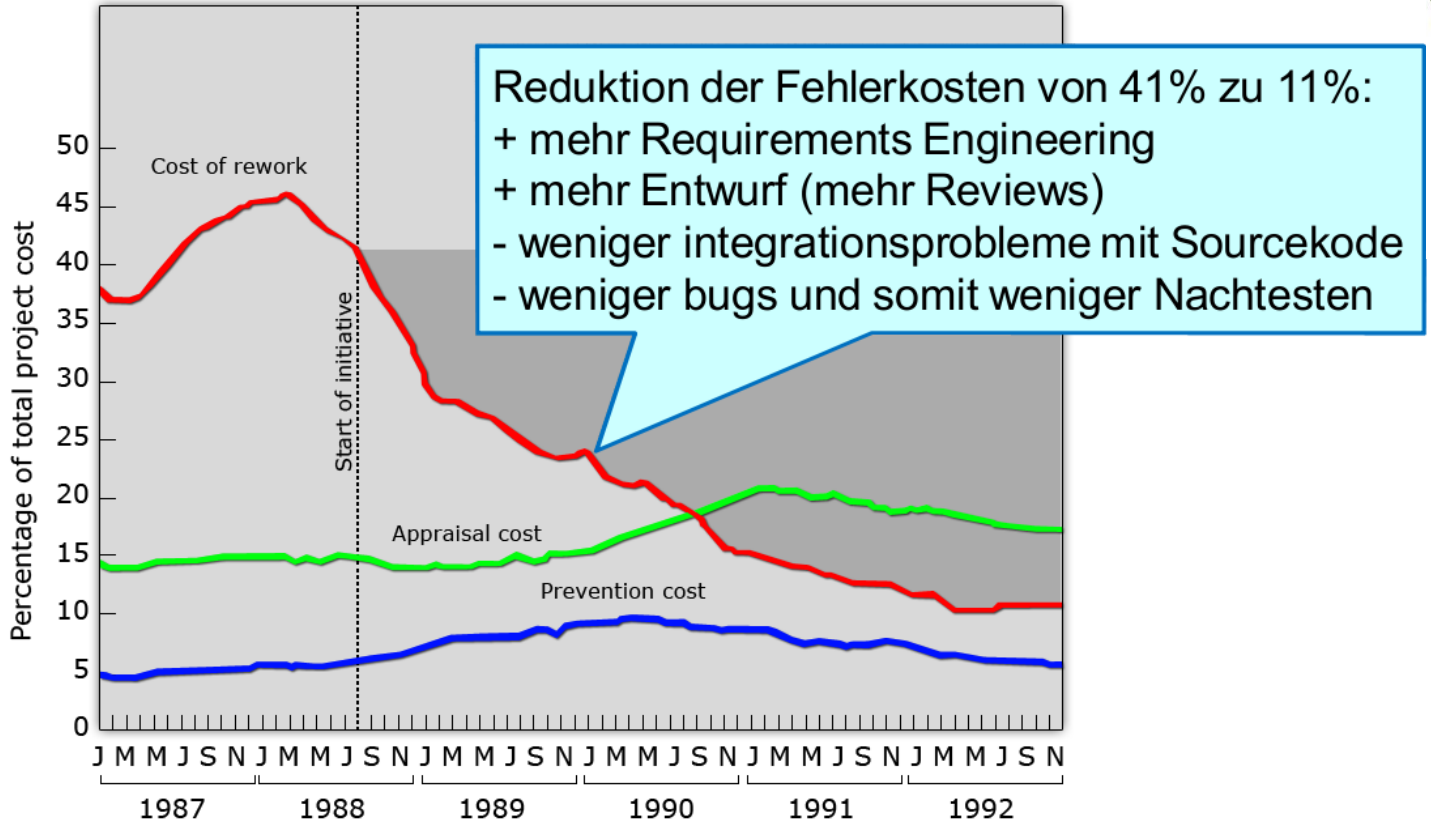
Teil 1.0: Qualität: Grundlagen

- Was ist Qualität?
- Qualitätsmerkmale
- Qualitätsmanagement
- Qualitätssicherungsprozesse



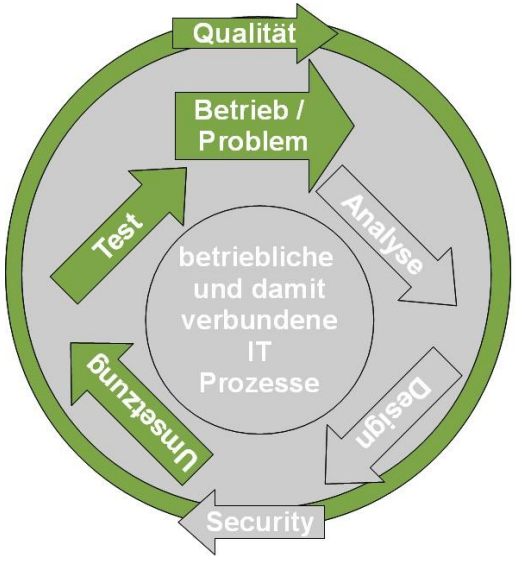
Teil 1.1: Prozess-Qualität

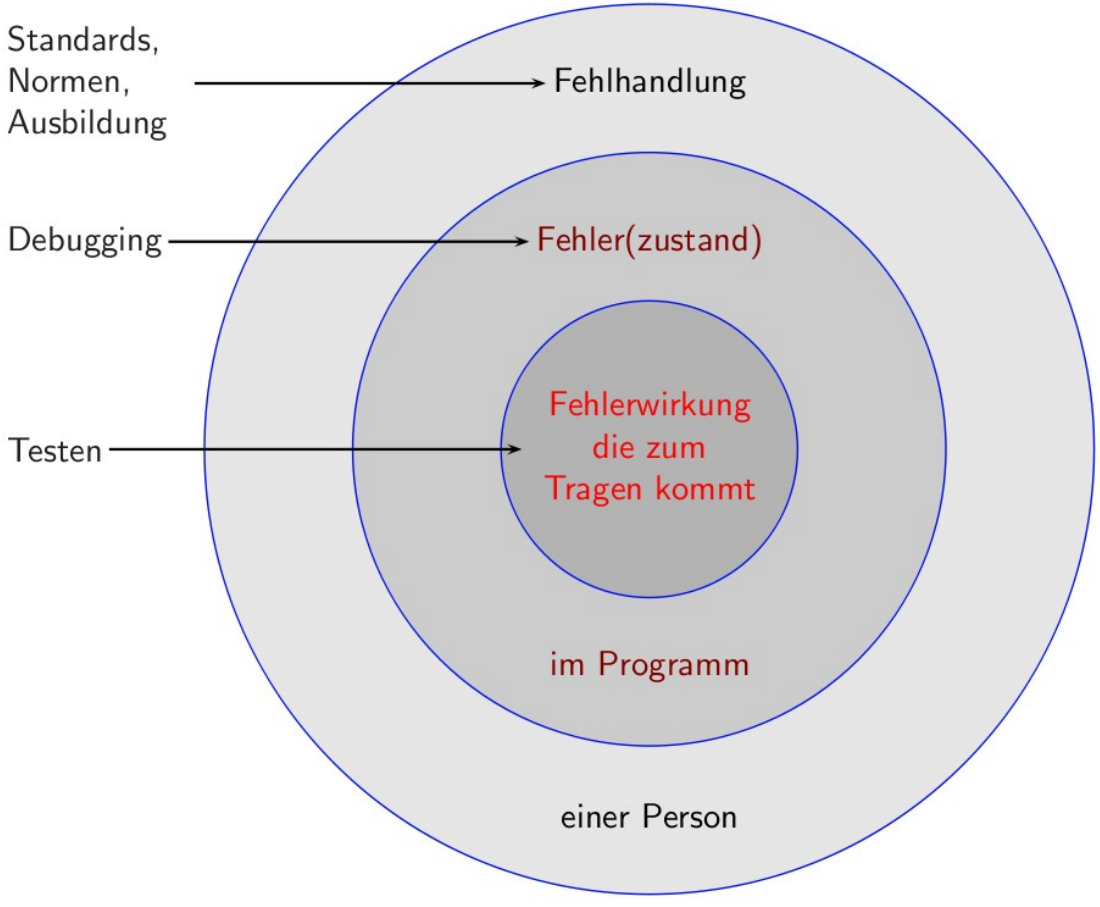


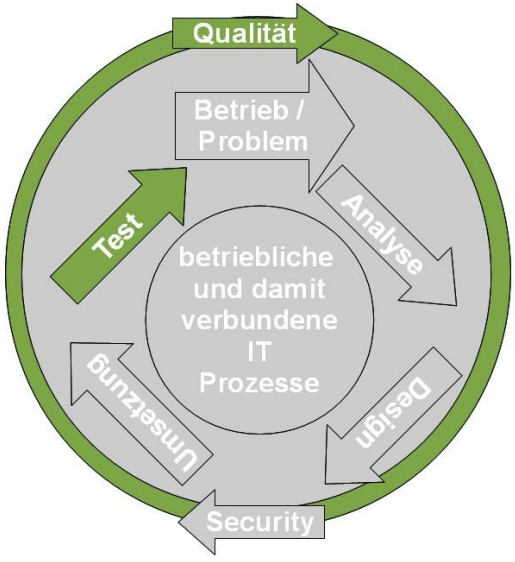


Teil 1.2: Softwaremetriken

- Metriken
- Direktes und indirektes Messen
- Vorgehensweisen
- Effekte

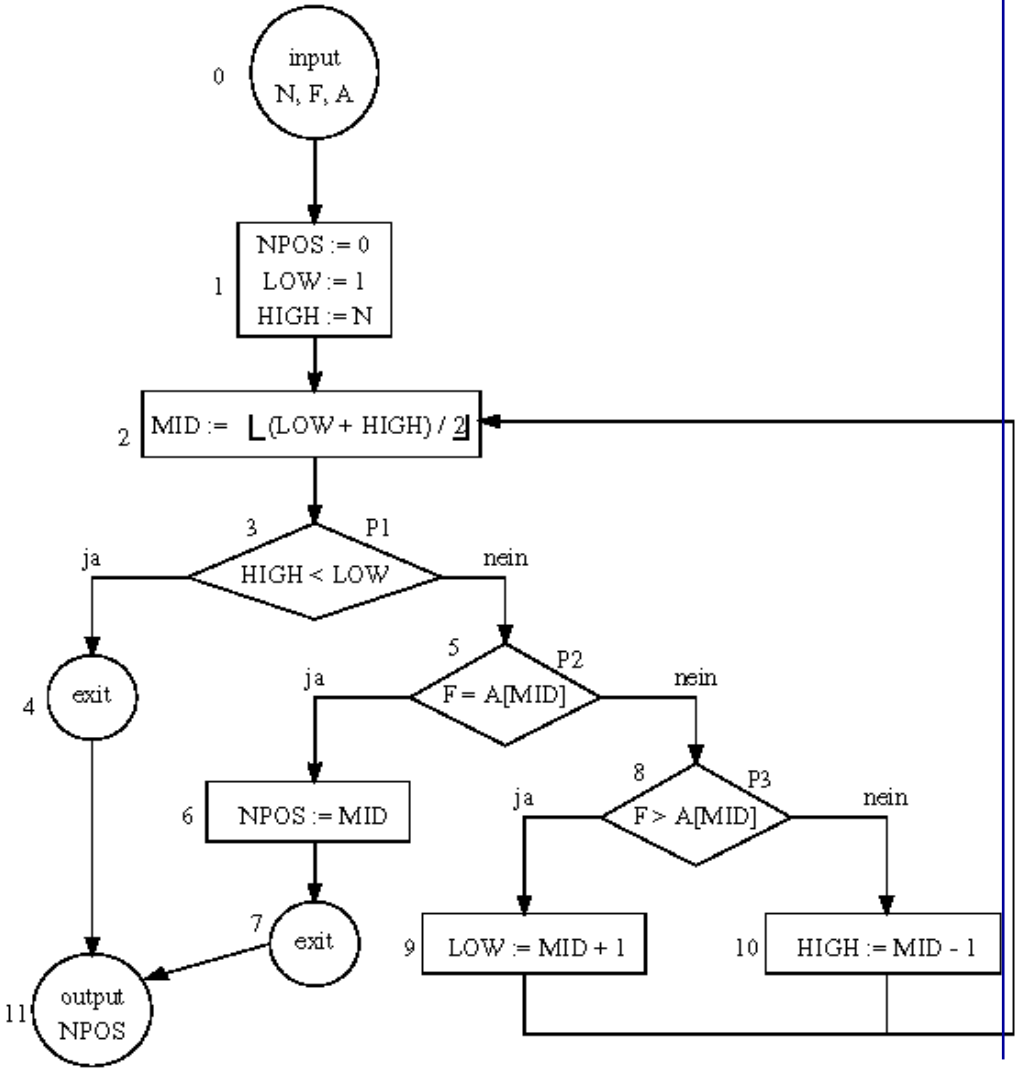


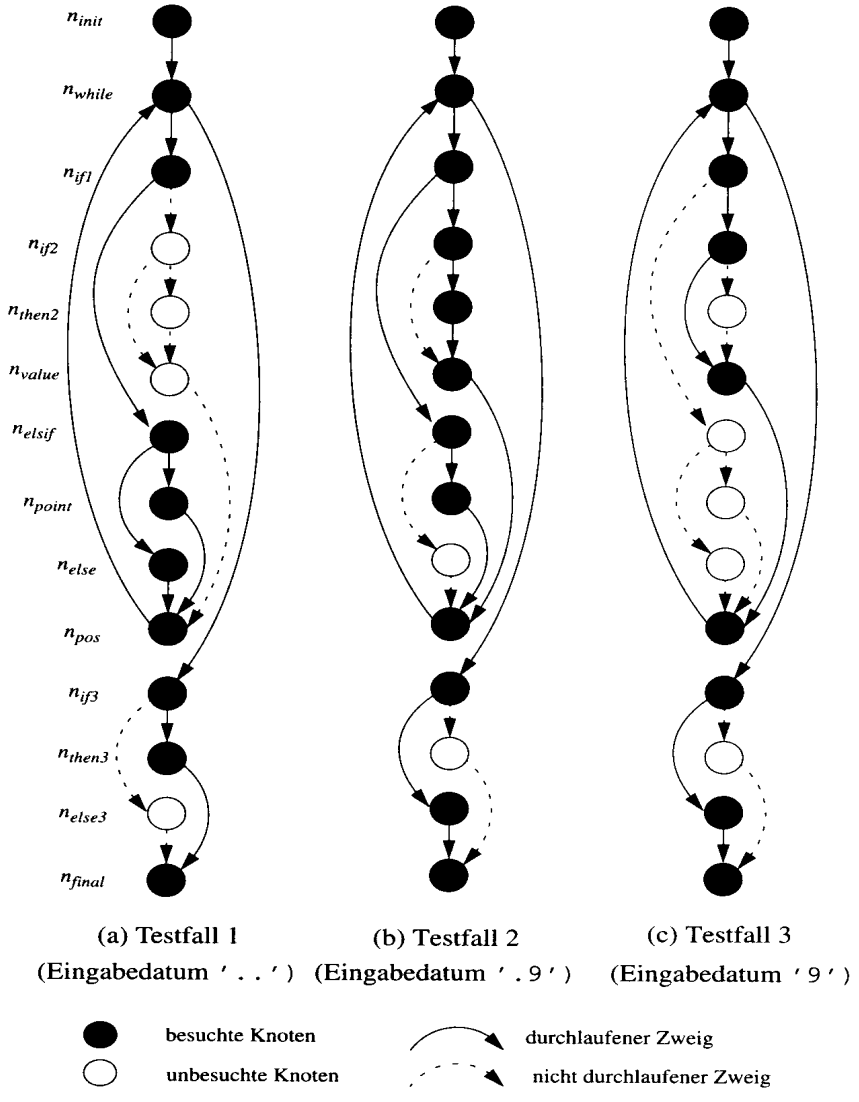

Standards,
Normen,
Ausbildung
Fehlhandlung
Debugging
Fehler(zustand)
Testen
Fehlerwirkung
die zum
Tragen kommt
im Programm
einer Person


Qualität
Betrieb /
Problem
Analyse
Design
Umsetzung
Test
betriebliche
und damit
verbundene
IT
Prozesse
Security

Kapitel 2: Testen

- Fehler in Softwaresystemen
  - Ursachen, Arten, Kosten und Folgen von SW-Fehlern
  - Ursachen von SW-Fehlern
  - Klassifikation von SW-Fehlern
  - Qualitätsmanagement
- Vorgehensmodelle
- Testverfahren
- Einleitende Begriffe (rund ums Testen)
  - Analysierende Prüfverfahren
  - Reviews, Inspektionen, Walkthroughs
  - statische und dynamische Tests

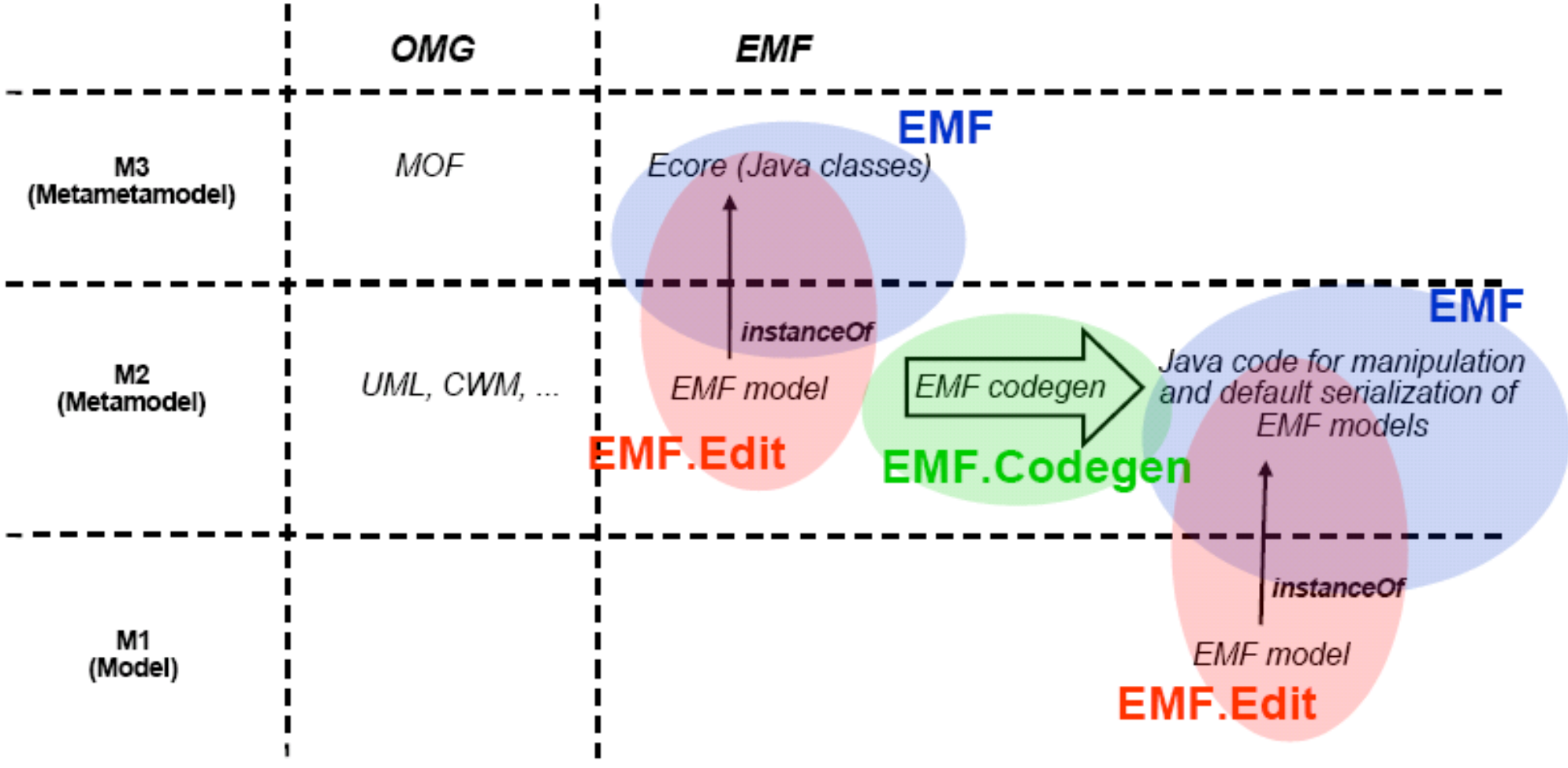

0
input
N, F, A
1
NPOS := 0
LOW := 1
HIGH := N
2
MID := ⌊(LOW + HIGH) / 2⌋
3
P1
HIGH < LOW
ja
nein
4
exit
5
P2
F = A[MID]
ja
nein
6
NPOS := MID
7
exit
8
P3
F > A[MID]
ja
nein
9
LOW := MID + 1
10
HIGH := MID - 1
11
output
NPOS


n_{init}
n_{while}
n_{if1}
n_{if2}
n_{then2}
n_{value}
n_{elsif}
n_{point}
n_{else}
n_{pos}
n_{if3}
n_{then3}
n_{else3}
n_{final}
(a) Testfall 1
(Eingabedatum '..')
(b) Testfall 2
(Eingabedatum '.9')
(c) Testfall 3
(Eingabedatum '9')
besuchte Knoten
unbesuchte Knoten
durchlaufener Zweig
nicht durchlaufener Zweig

Teil 3.0: Modellgetriebene SW-Entwicklung

- Modelle und Modellierung
  - Modell-Begiff, Metamodell
  - Domänenspezifische Sprachen
- UML
  - Architektur von UML, MetaModell (Infrastructure)
  - Konzepte der Sprachdefinition (Superstructure)
  - Semantik von UML, Erweiterungen / Tailoring von UML
- Modelltransformationen
  - Allgemeines Konzept der Modelltransformation
  - Verschiedene Konzepte + Transformationssprachen
- Szenarien der Modellgetriebenen SW-Entwicklung
  - Kommunikation mit Modellen, Spezifizieren mit Modellen
  - Testen + Simulieren mit Modellen, Programmieren mit Modellen
  - Berichte aus der MDA-Praxis

Teil 3.1: EMF


OMG
EMF
M3
(Metametamodel)
MOF
EMF
Ecore (Java classes)
instanceOf
M2
(Metamodel)
UML, CWM, ...
EMF model
EMF.Edit
EMF codegen
EMF.Codegen
EMF
Java code for manipulation
and default serialization of
EMF models
instanceOf
M1
(Model)
EMF model
EMF.Edit

Operation	Description
hasReturned() : Boolean	True if type of template parameter is an operation call, and the called operation has returned a value.
result()	Returns the result of the called operation, if type of template parameter is an operation call, and the called operation has returned a value.
isSignalSent() : Boolean	Returns true if the OclMessage represents the sending of a UML Signal.
isOperationCall() : Boolean	Returns true if the OclMessage represents the sending of a UML Operation call.
parameterName	The value of the message parameter.

- $\Sigma = (S,F)$ heißt algebraische Signatur
  - S eine Menge von Sorten
  - F eine Menge von Operationssymbolen
  - Auf F ist eine Abbildung definiert

    type: $F \rightarrow S^* \times S$
  - Für type(f)=$(s_1,...,s_n,s)$ schreiben wir f: $s_1,...,s_n \rightarrow s$
- Σ-Algebra (Definition)
  - Seien $\Sigma = (S,F)$ eine Signatur.
  - Für alle $s \in S$ sei A_s eine Menge.
  - Für alle f: $s_1,...,s_n \rightarrow s \in F$ sei $f_A: A_{s1} \times ... \times A_{sn} \rightarrow A_s$ eine Abbildung.
  - Dann ist das Paar
    - $A = ((A_s)_{s \in S}, (f_A)_{f \in F})$ eine Σ-Algebra

Literatur

Kapitel 1, 3:

* Jochen Ludewig / Horst Lichter: Software Engineering - Grundlagen, Menschen, Prozesse, Techniken, dpunkt.verlag, 2. Auflage, 2010

Kapitel 2:

* Andreas Spillner, Tilo Linz: Basiswissen Softwaretest, Aus- und Weiterbildung zum Certified Tester Foundation Level nach ISTQB-Standard. 4., überarbeitete Auflage, dpunkt.verlag, 2010, 308 Seiten, 39 Euro (D), ISBN 987-3-89864-642-0

* Eike Riedemann: Testmethoden für sequentielle und nebenläufige Software-Systeme. Teubner, Stuttgart, 512 S., 1997 (als Buch leider vergriffen; aber vollständig als PDF-Dateien hier herunterladbar sowie vorhanden in der Bereichsbibliothek Informatik (3406/Ried) und in der Uni-Bibliothek/Lehrbuchsammlung (L Sr 366, fünfmal)).