Vorlesung (WS 2013/14)

Softwarekonstruktion

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

Teil 0: Organisatorisches und Einleitung

v. 18.10.2013

Vorlesungswebseite (bitte notieren):

http://www-secse.cs.tu-dortmund.de/secse/pages/teaching/ws13-14/swk/index_de.shtml
(oder: http://jan.jurjens.de : rechte Spalte: „Softwarekonstruktion“; oder: Link im LSF)

Agenda

- **Organisatorisches**
  - **Studienordnung: Einordnung / Kompetenzen / Struktur**
  - **Vorlesung: Bildungsvertrag, Termine, Feedback**
  - **Übung: Konzept / Termine**
  - **Klausur**
- Vorstellung des Fachgebietes
- Vorlesungsinhalte

Studienordnung
Struktur laut Modulhandbuch (Bachelor)

Bachelor Informatik / Angewandte Informatik: **Wahlpflichtmodul.**

Teilnahmevoraussetzungen:

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

Umfang: 3 SWS (2 SWS Vorlesung, 1 SWS Übung)

4 Credits: 3 Credits Vorlesung, 1 Credit Übung

Aufwand: **120 Stunden** über 15 Semesterwochen:

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

Veranstaltungssprache: Deutsch.

Bildungsvertrag: Wir bieten ...

- **Fachliche Einführung** in das Thema Softwarekonstruktion.
- **Engagierte Betreuung:**
  - Interessante Vorlesung.
  - Regelmäßige Sprechstunden.
  - 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.

Wegen des inhaltlichen hohen Anspruchs der Vorlesung ist die **regelmäßige Anwesenheit in der Vorlesung notwendig** für eine erfolgreiche Belegung der Vorlesung, da die Inhalte zu komplex für eine Aneignung im Selbststudium sind.

Vorlesung
Termine, Folien

Vorlesungstermin: Fr. 10:15 bis 11:45 Otto-Hahn-Str. 14 – E23

Abstimmung: Start um 10.00 oder 10.15 ?

Aktuelle Informationen zur Vorlesung (**bitte regelmäßig beachten wegen möglicher Vorlesungsausfälle o.ä.**):

http://www-secse.cs.tu-dortmund.de/secse/pages/teaching/ws13-14/swk/index_de.shtml

(oder: http://jan.jurjens.de : rechte Spalte: „Softwarekonstruktion“).

Vorlesungsfolien werden auf o.g. Webseite zur Verfügung gestellt (planmäßig spätestens 18.00 Uhr am Vortag der Vorlesung).

Enthalten Diskussionsfolien; zugehörige Antwortfolien werden erst nach der Vorlesung online gestellt.

Vollständigkeit der Folien (Klausurvorbereitung) vs. „Textwand“: Teilweise **Anhänge**, deren Inhalt in Vorlesung nur mündlich wiedergegeben wird (z.B. für Klausurvorbereitung).

Feedback

Wir **bitten um vorlesungsbegleitendes Feedback**, um Verbesserungen semesterbegleitend durchführen zu können.

Übliche Kontaktmöglichkeiten:

- Nach der Vorlesung
- E-mail jan.jurjens@cs.tu-dortmund.de
- Tel.: 0231 755-7208
- Sprechstunde: Fr 9-10 (bitte vorher per email anmelden)
- Anonymes Kontaktformular:
  http://www-secse.cs.tu-dortmund.de/secse/pages/teaching/feedback_de.shtml
  (s. Link von Vorlesungswebseite). SWK ankreuzen !

Darüberhinaus: interne Umfragen für vorlesungsbegleitendes Feedback.

Übung
Termine

Termine (**14 tägig**):

- Mo 14:15 - 15:45, Start: 28.10./4.11.2013, OH 14 – 104
- Mi 10:15 - 11:45, Start: 30.10./5.11.2013, OH 14 – E02
- Mi 14:15 - 15:45, Start: 30.10.2013, SRG1 – 3.011
- Mi 16:15 - 17:45, Start: 30.10.2013, OH 14 – E02
- Do 10:15 - 11:45, Start: 31.10./6.11.2013, OH 14 – E02
- Do 14:15 - 15:45, Start: 31.10./6.11.2013, MSW16 – E29

Übung
Termine

VL-Woche	KW	Woche vom	Übung	Gruppe	Bemerkung
1	42	14.10.2013	-	-	
2	43	21.10.2013	-	-	
3	44	28.10.2013	1	1 - 6	Allerheiligen am 1. November
4	45	04.11.2013	1	7 - 10	
5	46	11.11.2013	2	1 - 6	
6	47	18.11.2013	2	7 - 10	
7	48	25.11.2013	3	1 - 6	
8	49	02.12.2013	3	7 - 10	
9	50	09.12.2013	4	1 - 6	
10	51	16.12.2013	4	7 - 10	
					Weihnachtspause
11	2	6.01.2014	5	1 - 6	
12	3	13.01.2014	5	7 - 10	
13	4	20.01.2014	6	1 - 6	
14	5	27.01.2014	6	7 - 10	
15	6	03.02.2014	-	-	

Anmeldung:

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

- Anmeldung heute ab 13 Uhr möglich.
- **Anmeldung bis 23.10.2013, 09:00 Uhr.**
- Verteilung mittels Algorithmus (Solver).
  (nicht priorisiert nach zeitlichem Eingang der Anmeldung)
- **Bekanntgabe der Verteilung am 25.10.2013.**

Übung
Ablauf Präsenzübung

Übungsablauf:

- Übung wird als **Präsenzübung** durchgeführt.
- Übungszettel werden jeweils **14-tägig** veröffentlicht und sind Inhalt nächster Übung.
- Übungszettel während Übung alleine oder in Gruppen bearbeiten.
- Anwesender Tutor steht für Fragen zur Verfügung.
- Am Ende der Übung werden **von den Studierenden** Lösungen vorgeschlagen und Aufgaben besprochen.
- **Lösungsvorschlag** zur Präsenzübung wird über unsere Webseite veröffentlicht.

Übung
Hausübungen I

Übungskonzept:

- Insgesamt werden 6 **Übungszettel** veröffentlicht.
- 5 davon enthalten eine Hausübung (+ 1 Online-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 e-Mail !**

Übungskonzept:

- 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.
- Abgaben werden korrigiert und die Gruppe erhält die korrigierte Lösung in der Übungsgruppe zurück.

→ **Gruppennummer auf die Abgabe schreiben.**

Übung
Leistungsnachweis

Übungskonzept:

- Bei jeder **Hausübung** gibt es 10 Punkte (die 10 Punkte für die Online-Übung (Übung 4) gelten als Bonus-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 (25 von 50+10)
  - aber mindestens **jeweils 30% der Punkte aus den Aufgaben 1+2+3** (9 von 30) **und 5+6** (6 von 20)

Übung
Hausübungen – Zeitschema

Sonntag	Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag
20	21	22	23	24	25	26
27	28 Übung W1 Ausgabe Übungszettel im WWW	29	30 Übung W1	31 Okt Übung W1	1 Nov	2
3	4 Übung W2	5	6 Übung W2	7 Übung W2	8	9
10	11	12	13	14 Abgabe-ende	15	16

Inpud-Forum

Ziel: **Diskussion** der Studierenden untereinander.

Keine Kommunikation mit Veranstaltern dort:

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

Organisatorische + inhaltliche FAQ

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

Moderation durch Veranstalter.

Studienordnung
Prüfung

Bachelor Informatik / Angewandte Informatik**:**

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

=> **Bearbeitung der Übungsaufgaben** ist **Voraussetzung** zur Teilnahme an der Modulprüfung.

Diplom:

- **Prüfungsleistung**: Klausur

Prüfung
Termine

Prüfung: Klausur

- schriftlich
- 60 Minuten

Klausurtermine:

- Mittwoch, 26.02.2014, 09:00 - 10:30 Uhr
  HS 1, HS 2, HS 3 (Emil-Figge Str. 50)
- Donnerstag, 27.03.2014, 10:45 - 12:15 Uhr
  HS 1 (Emil-Figge Str. 50) und E23 (Otto-Hahn-Str. 14)

Kontakt

Vorlesungsseite (bitte regelmäßig beachten):
http://www-secse.cs.tu-dortmund.de/secse/pages/teaching/ws13-14/swk/index_de.shtml
(oder: http://jan.jurjens.de : rechte Spalte: „Softwarekonstruktion“; oder: Link im LSF)

Ansprechpartner:

- Vorlesung:
  - Jan Jürjens
- Übung:
  - Jens Bürger
  - Nina Harmuth
  - Janine Hemmers
  - Sebastian Pape
- Inpud-Forum: http://inpud.cs.uni-dortmund.de
- Übungsanmeldung: http://ess.cs.uni-dortmund.de/ASSESS

Agenda

- Organisatorisches
- **Vorstellung des Fachgebietes**
  - **Forschung**
  - **Abschlussarbeiten, Hiwi-Jobs, weiteres Lehrangebot**
- Vorlesungsinhalte

Vertrauenswürdige IT-Systeme

- IT Systeme durchziehen fast alle Funktionen in Wirtschaft und Gesellschaft.
- IT hat direkten **Einfluss auf** fast alle **Aspekte menschlichen Lebens**.
- **Erwartungen an Vertrauenswürdigkeit** der Systeme in letzten 10 Jahren stark gestiegen.

  → **Oft nicht erfüllt**.
- **Teil des Problems**: Bislang verwendete System- und Software-Entwicklungsmethoden konnten mit gestiegenen Erwartungen bei gleichzeitig steigender Systemkomplexität nicht mithalten.

Offene Systeme

Aus Flexibiliäts- und Kostengründen:
Moderne IT Systeme meist
über **offene Infrastrukturen realisiert**.

Zum Beispiel:

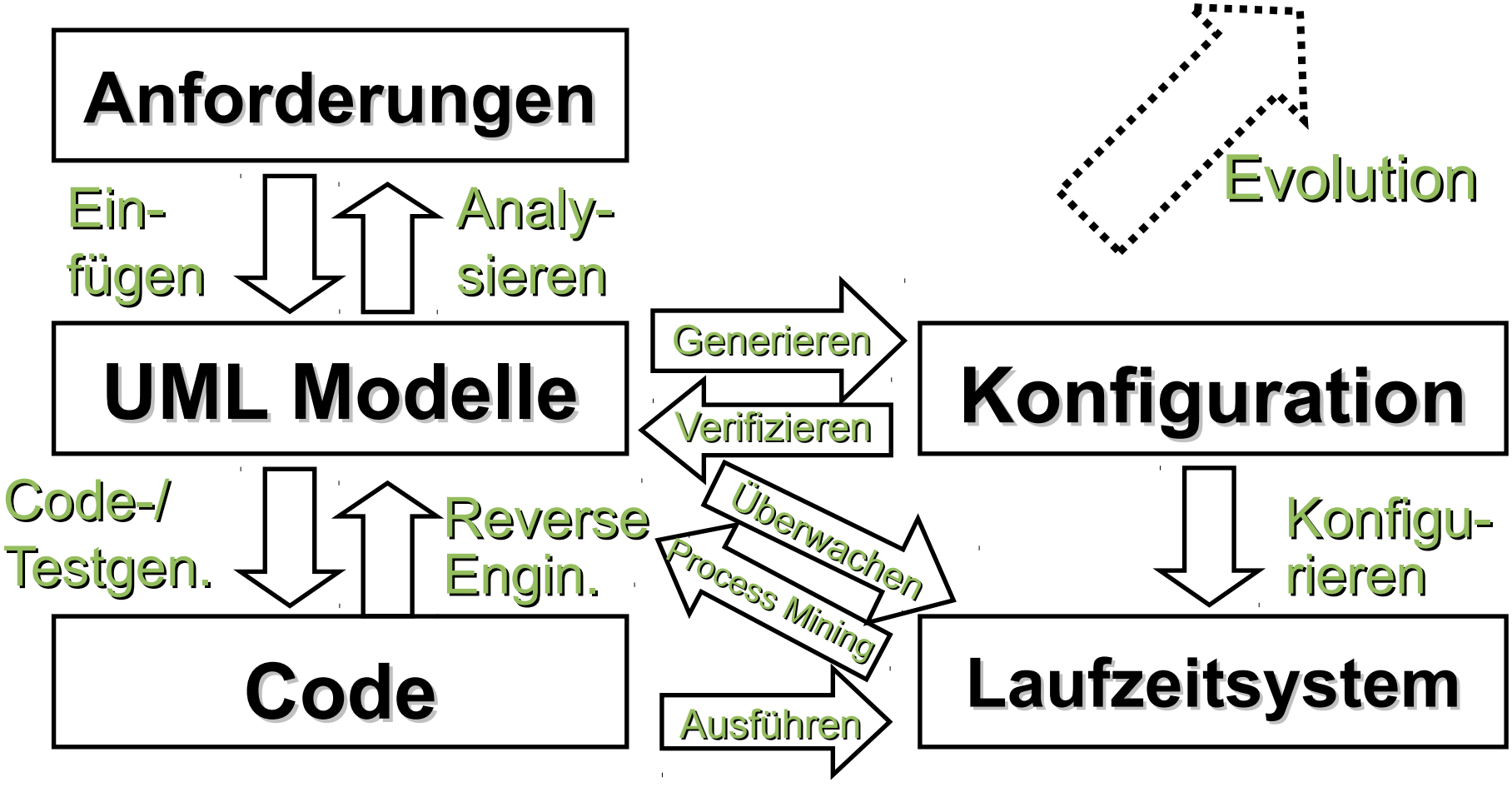
- Internet
- Mobile Netze

Sie sind somit dem Zugriff von Personen ausgesetzt, die nicht unbedingt vertrauenswürdig sind.

Zugriff muss daher **systemseitig** reguliert werden. Aus Flexibilitäts- und Kostengründen oft auf Softwareebene gelöst.

Vertrauenswürdige IT braucht also **sichere Software**.

Modellbasierte Entwicklung



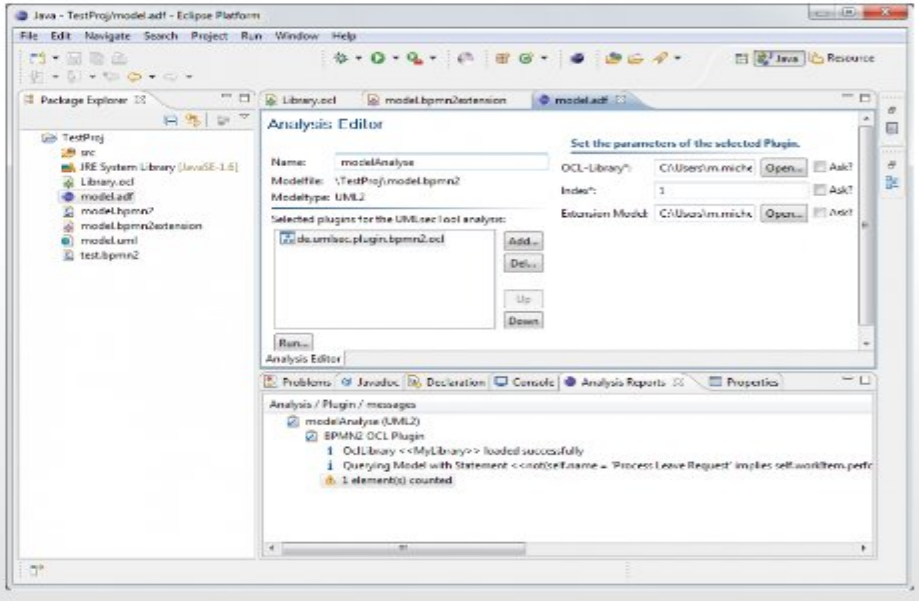
Werkzeugunterstützung

CARiSMA

Home - News - Team - Contact
Features - Checks - Docs - Installation

http://carisma.umlsec.de

2012-02-06: A new update of CARiSMA has been released!



Welcome to CARiSMA!

Modeling offers an unprecedented opportunity for high-quality critical systems development that is feasible in an industrial context. CARiSMA enables you to perform:

- **compliance** analyses,
- **risk** analyses, and
- **security** analyses

of software models.[1)]

Since CARiSMA is a reimplemented variant of the former UMLsec tool it natively supports UML models. Due to its EMF-based implementation CARiSMA can also support **domain-specific modeling languages** such as BPMN.

CARiSMA is fully **integrated into Eclipse** and can thus become part of the modeling tool of your choice including but not limited to TOPCASED, Papyrus MDT, IBM Rational Software Architect, and many others.

A flexible **plugin architecture** makes CARiSMA extensible for new languages and allows users to implement their own compliance, risk, or security checks.

Anwendungen in Unternehmen

- Digitaler Formularschrank [SAFECOMP 03]
- Internes Informationssystem [ICSE 07]
- Biometrisches Authentisierungssystem: mehrere Schwachstellen aufgedeckt [ACSAC 05, Models 09]
- Gesundheitskarte: Architektur mit UMLsec untersucht, Schwachstellen aufgedeckt [Jour. Meth. Inform. Medicine 08]
- Common Electronic Purse Specifications (Globaler Standard für elektr. Geldbörsen): mehrere Schwachstellen aufgedeckt [IFIPSEC 01, ASE 01]
- Gesundheitsinformationssysteme [Caise 09]
- Return-on-Security Investment Abschätzung
- Analyse Elektronische-Signatur-Architektur
- IT-Sicherheits-Risikomodellierung
- Smart-card Software-Update Plattform

Aktuell:

- Cloud-Anwender Compliance/Sicherheit
- Sicherheitsökonomische Analysen
- Cloud-Anbieter Compliance/Sicherheit

Weitere relevante Lehrveranstaltungen

LEHRSTUHL 14
SOFTWARE ENGINEERING

WiSe 2013/14:

- Vorlesung “Software-Konstruktion” (Bachelor-Wahlpflicht)
- **Vorlesung „Sicherheit: Fragen und Lösungsansätze“ (Bachelor-Wahlvorlesung)**
- Fachprojekt „Softwaretechniken für sichere Cloud-Computing-Systeme”
- Proseminar “Werkzeugunterstützung für sichere Software“
- Seminar „Software-Engineering und Sicherheit“

SoSe 2014 (**unter Vorbehalt**):

- **Fachprojekt „Softwaretechniken für sichere Cloud-Computing-Systeme”**
- **Proseminar “Werkzeugunterstützung für sichere Software“**
- **Vorlesung „Software-Engineering für langlebige Systeme“ (Bachelor-Wahlvorles.)**
- Vorlesung “Methodische Grundlagen des Software Engineering” (Master-Basismodul Software)
- Seminar „Software-Engineering und Sicherheit“

Forschungsbereich Master: Software, Sicherheit und Verifikation

Schwerpunktgebiete Diplom: Sicherheit und Verifikation, Software-Konstruktion

Informationen unter: http://www-secse.cs.tu-dortmund.de/secse/pages/teaching/index_de.shtml (oder http://jan.jurjens.de , Link: Lehre).

Einige Beispiel-Themen für Abschlussarbeiten

LEHRSTUHL 14
SOFTWARE ENGINEERING

- Mustererkennung und Identifizierung in Versionlogs
- Aspektorientierte Realisierung von UMLsec-Stereotypen am Beispiel des Secure Links-Stereotyps
- Adaption von UMLsec-Stereotypen zur Modellierung von Anforderungen an die Datensicherheit nach UML 2.x
- Evolutionen und Co-Evolutionen für ReL
- Differenz-basierte Sicherheitsverifikation
- Entwicklung eines Prüfdokument-Generators am Beispiel von Sicherheitszertifizierungen
- Sicherheit und Evolution von Informationssystemen
- Automatisiertes Mapping von Event-Log-Bezeichnern auf Aktivitäten zur Unterstützung von Compliance-Analysen

Informationen unter:

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

(oder: Vorlesungswebseite, linke Spalte: „Bachelor/Master/Diplomarbeiten “).

Abschlussarbeiten auch in Zusammenhang mit **Forschungsprojekten** am Fraunhofer ISST oder LS 14 / TUD möglich.

Bei Interesse **bitte bei mir melden** !

Hiwi-Tätigkeiten

Hiwi-Jobs am Fraunhofer ISST oder LS 14 / TUD:

- Unterstützung von **Forschungsprojekten** (z.B.: "Architectures for Auditable Business Process Execution (APEX)", Seconomics, SecVolution, ClouDAT):
  z.B. Java-Programmierung für UML-Analyse-Werkzeug, konzeptuelle Arbeiten zu modellbasierter Sicherheitsanalyse
- Unterstützung in der Lehre (Tutorien, Folienerstellung etc)

Weitere Informationen:

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

(oder: http://jan.jurjens.de ; rechte Spalte: HiWi-Stellen).

Auch Hiwi-Beschäftigung mit inhaltlichem Bezug zu **Abschlussarbeiten** möglich !

Bei Interesse **bitte bei mir melden** !

Auslandaufenthalte, Stipendien etc.

Vielfältige internationale Kontakte für Auslandsaufenthalte, z.B.:

- EU-Projekt Seconomics: Unis Trento (I), Aberdeen (UK), Madrid (S), Anadolu (TR); Firmen Atos Origin (F), National Grid (UK), Deep Blue (I), Barcelona Transport (S).

→ Bei Interesse **bitte bei mir melden**.

Als ehemaliger Geförderter: Vorschlagsrecht für Aufnahme in Förderung der Studienstiftung des deutschen Volkes.

→ Bei Interesse **bitte bei mir melden**.

Und danach ?

[http://www.fraunhofer.de/de/presse/presseinformationen/2013/Januar/fraunhofer-schafft-wieder-ueber-1000-neue-arbeitsplaetze---press.html]

"**Fraunhofer schafft wieder über 1000 neue Arbeitsplätze.** Fraunhofer wird auch 2013 wieder stark wachsen und über 1000 zusätzliche Stellen schaffen. ... Der weitere Ausbau erfolgt vor allem in den stark nachgefragten Forschungsgebieten, die von Energiewende, Elektromobilität, Produktionstechnik und digitalem Wandel angetrieben werden. »Unser Wachstum speist sich sowohl aus öffentlichen Förderprogrammen als auch aus Aufträgen mit der Wirtschaft. Das belegt, wie leistungsfähig und erfolgreich Fraunhofer am Forschungsmarkt agiert.«

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

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.

Agenda

- Organisatorisches
- Vorstellung des Fachgebietes
- **Vorlesungsinhalte**

Ziele dieser Vorlesung

- Vermittlung eines Überblicks über das Spektrum gängiger **Konzepte zur Spezifikation** und **zum Testen.**
- **Einbettung dieser Spezifikationskonzepte** in die Qualitätssicherung.
- **Modellbasiertes Software-Engineering**.
- Erläuterung verschiedener **Testtechniken.**
- **Fortgeschrittene Spezifikations-** und **Verifikationstechniken** (OCL, statisches Analyse)

Software Engineering is...

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

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

Defizite in der Praxis

Entwicklungsprozesse der Software-Industrie bei weiten nicht so planbar, zuverlässig, effektiv, effizient und flexibel wie die „althergebrachter“ Industrien:

- **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.

Probleme: Verspätung, Abbruch

LEHRSTUHL 14
SOFTWARE ENGINEERING

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]

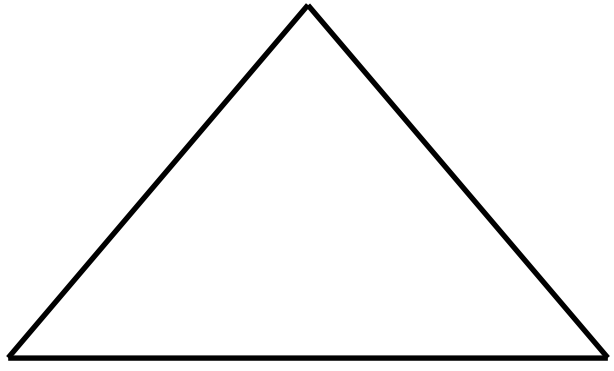
Andere Quelle: **Erfolgreich** durchgeführte **Software-Projekte**:

- 16% im Jahre 1994; 34% im Jahre 2003

1 Function Point (FP): ca. 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

Interessenkonflikte: Termine, Kosten, Qualität

Qualität: korrekt
(funktionsfähig)

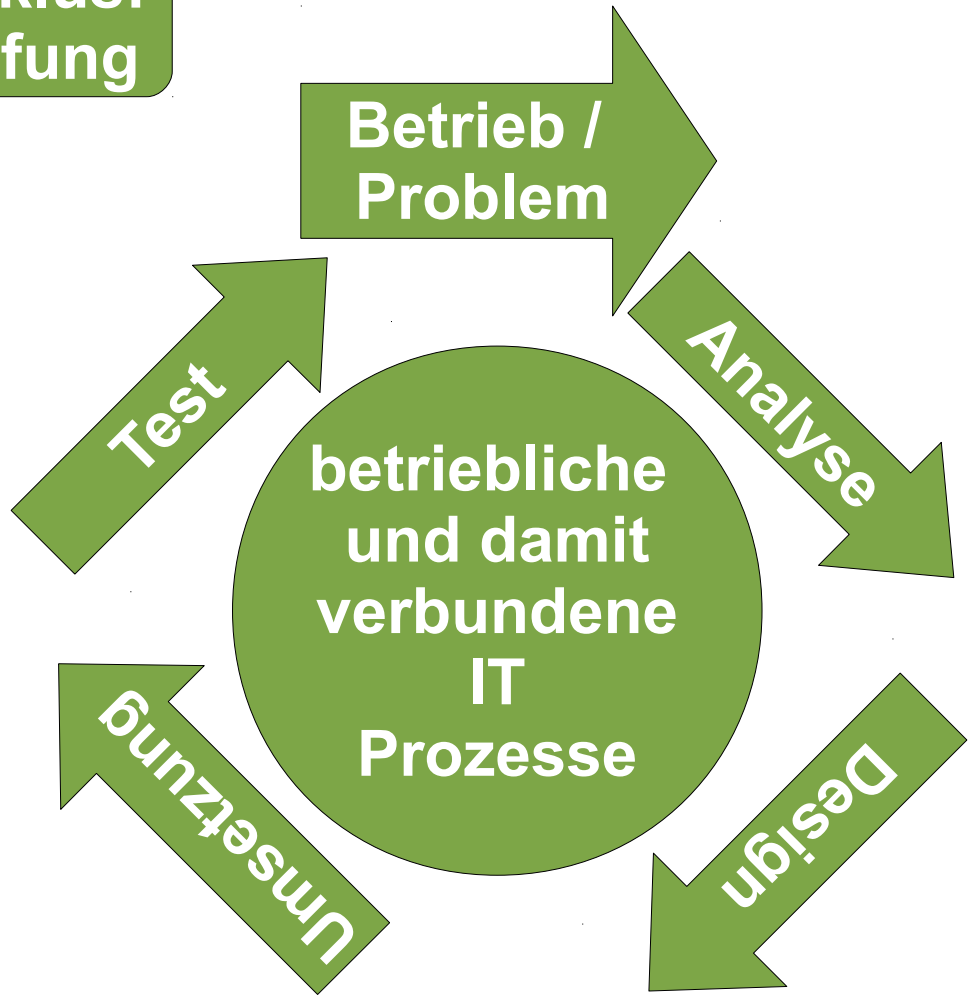


Termin:
rechtzeitig

Kosten:
im Budget

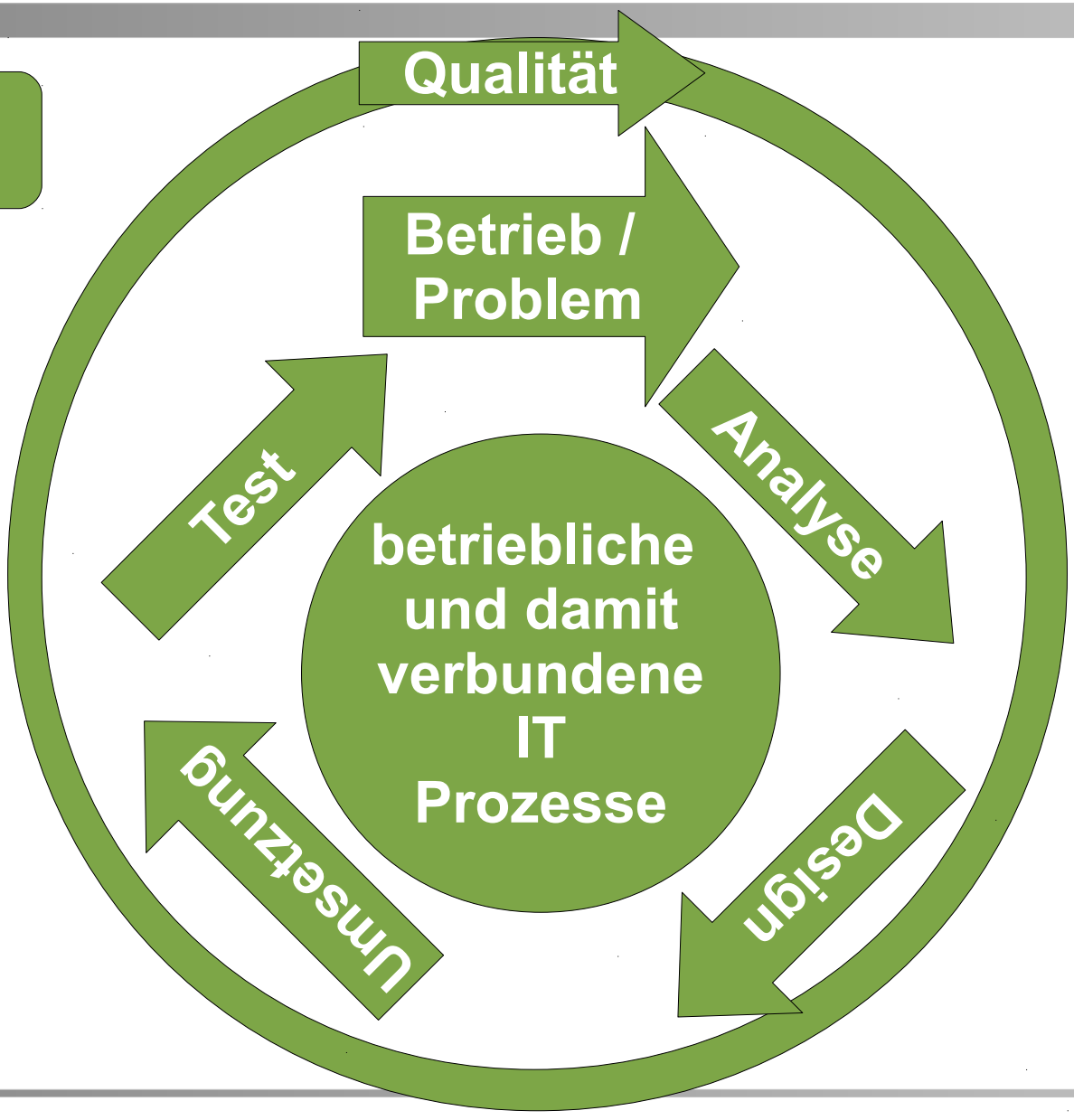
Vorlesungsüberblick
Inhaltlicher Zusammenhang

Der SE Lebenszyklus: Punktuelle Vertiefung



Vorlesungsüberblick
Inhaltlicher Zusammenhang

Umfassend: Qualität



Themen der Vorlesung

Kapitel 1: Modellbasierte Softwareentwicklung
* Teil 1.0: Modellbasierte Softwareentwicklung: Einführung
* Teil 1.1: Grundlagen: OCL
* Teil 1.2: Modellbasierte Softwareentwicklung
* Teil 1.3: EMF

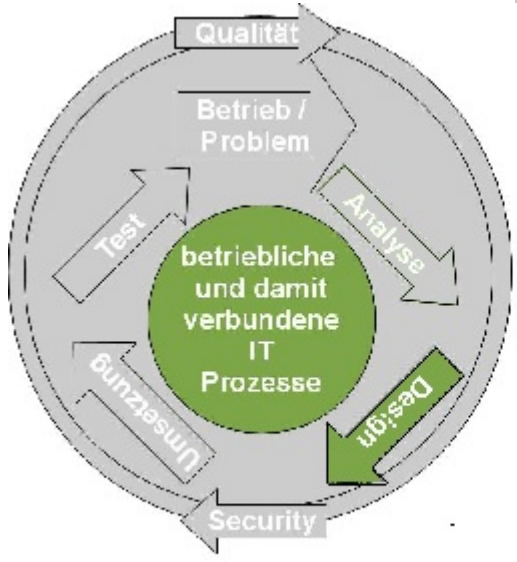
Kapitel 2: Qualitätsmanagement
* Teil 2.0: Qualität: Motivation
* Teil 2.1: Qualität: Grundlagen
* Teil 2.2: Prozess-Qualität

Kapitel 3: Softwareprüfung/Testen
* Teil 3.0: Grundlagen des Softwaretestens
* Teil 3.1: Testen im Softwarelebenszyklus
* Teil 3.2: Statischer Test
* Teil 3.3: Softwaremetriken
* Teil 3.4: Black Box Test
* Teil 3.5: White Box Test
* Teil 3.6: Testmanagement
* Teil 3.7: Testwerkzeuge

Kapitel 4: Modellbasierte Sicherheit
* Teil 4.0: Algebraische Spezifikation
* Teil 4.1: Modellbasierte Sicherheit mit UMLsec
* Teil 4.2: Werkzeugunterstützung
* Teil 4.3: Anwendungen

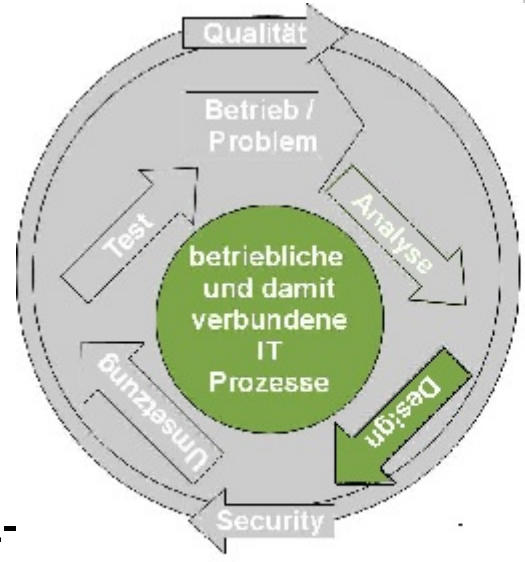
Einige Themen:

- **Modelle und Modellierung: Begriffe**
- **UML: Wiederholung**
- **Szenarien der Modellgetriebenen SW-Entwicklung**



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.

Beispielthema: Metamodellhierarchie der UML:



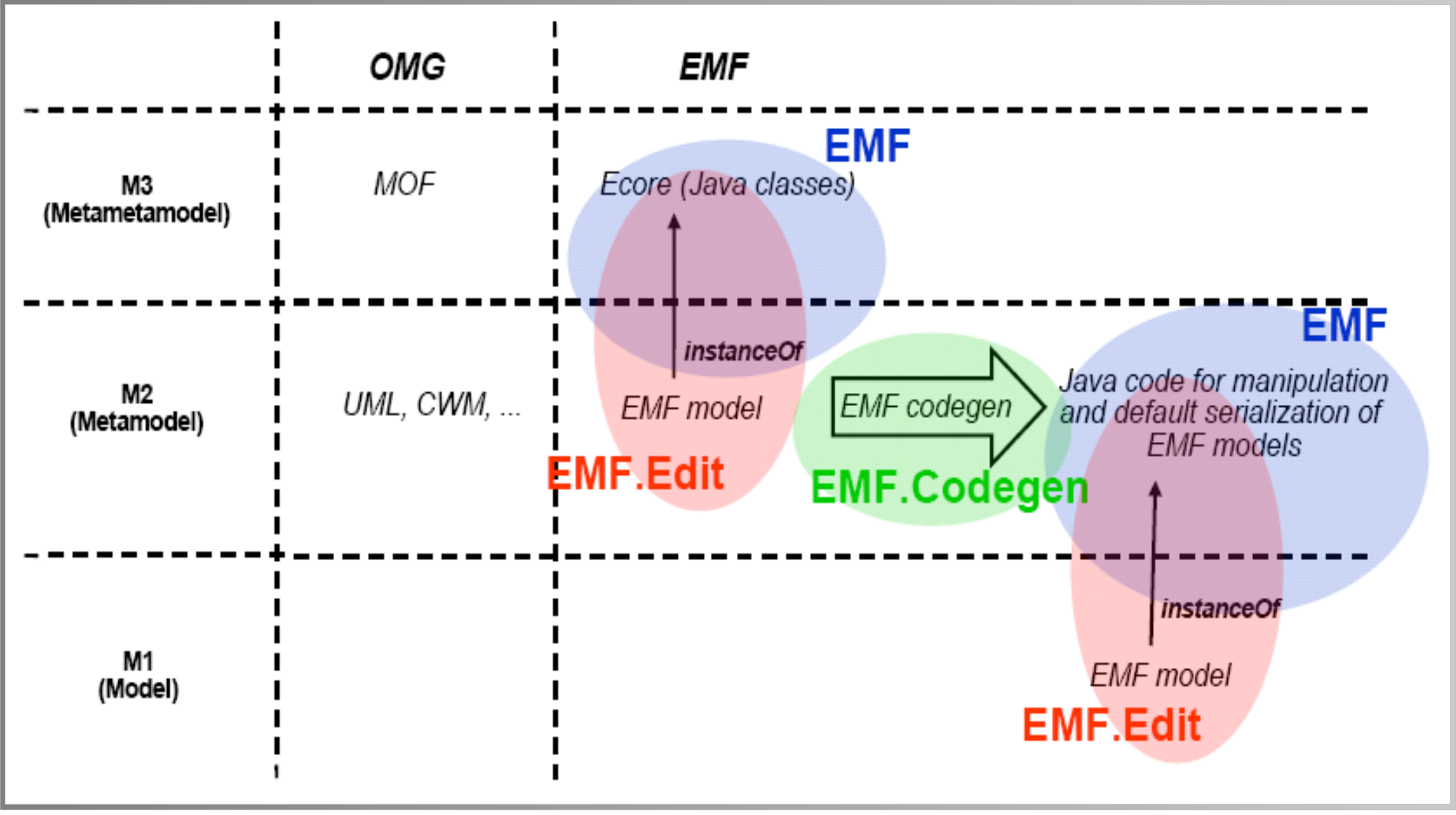
UML Infrastructure: Definition der Elemente, mit der UML-Diagrammtypen spezifizierbar sind.

UML Superstructure: Definition der 13 UML-

Modell eines konkreten Systems.

Instanzen eines modellierten konkreten Systems.

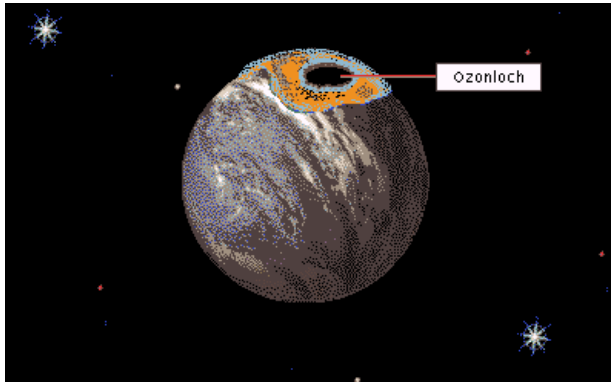
Teil 1.3: EMF


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

Teil 2.0: Qualität: Motivation

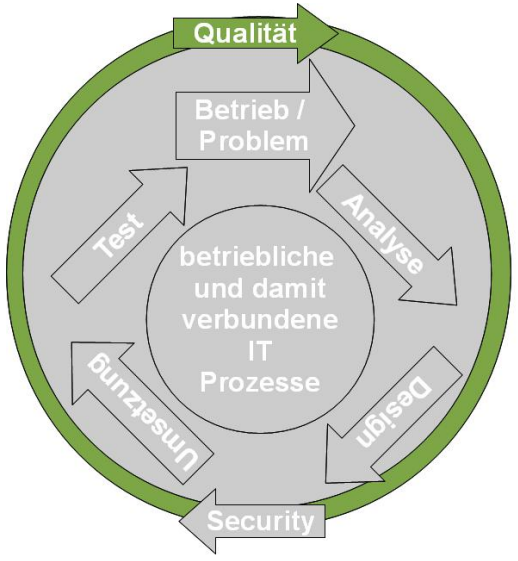
Beispiele für Auswirkungen von Software-Fehlern:

- **NASA - Erdbeobachtungssatelliten** 1979-1985: **Ozonloch** 7 Jahre (!) lang **nicht erkannt.**
- ESA, Kourou, Franz. Guyana, 4. Juni 1996: **Selbstzerstörung der Ariane** 5 beim Jungfernflug 39 Sekunden nach Start.
- Bemannte **NASA-Raumkapsel** Gemini V: **Verfehlte ihren Landeplatz** um ca. 160 Kilometer.

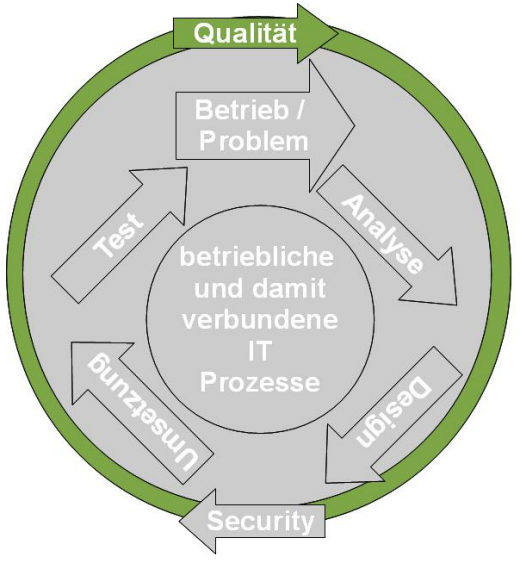


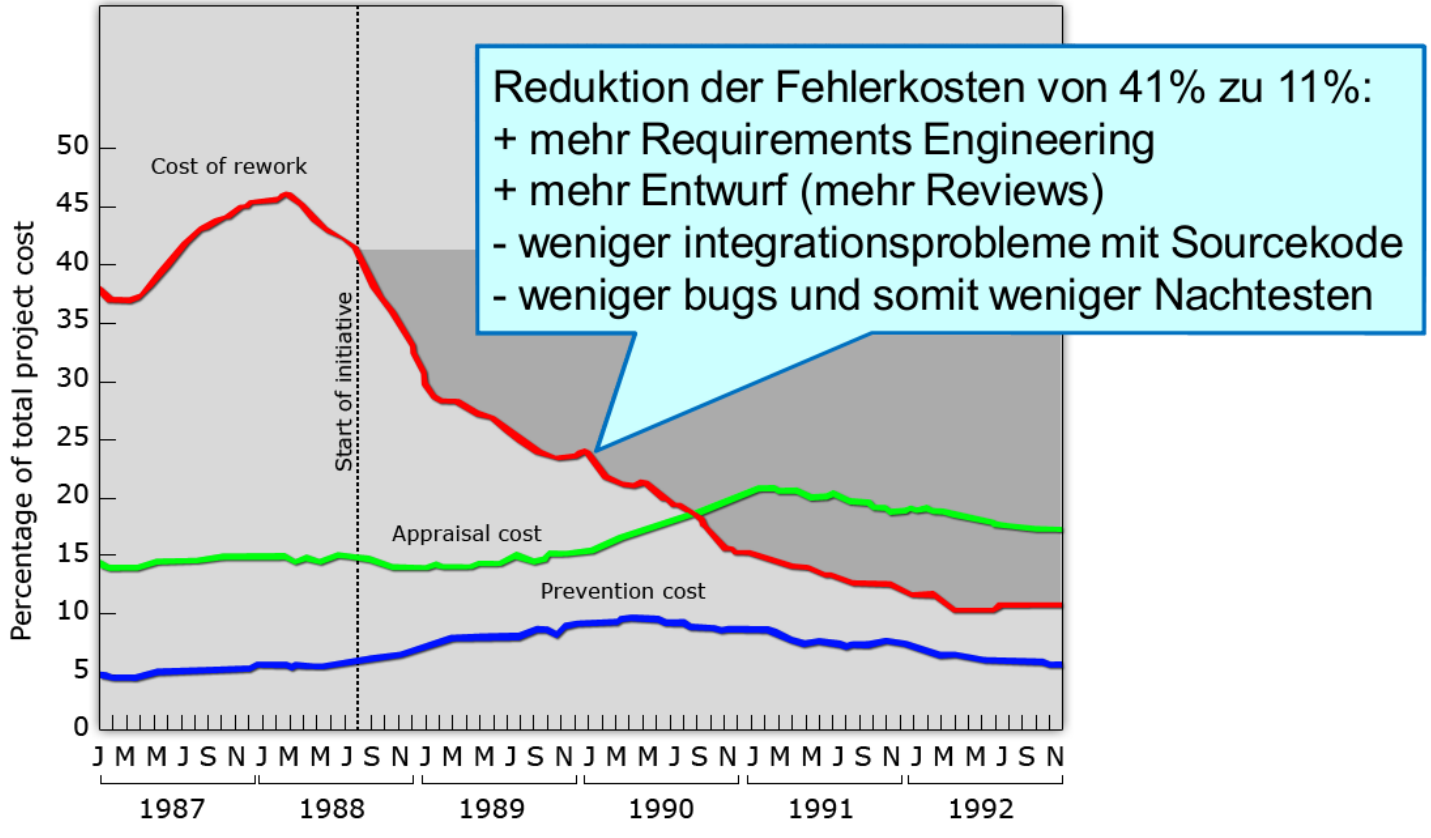
Teil 2.1: Qualität: Grundlagen

- Was ist Qualität?
- Qualitätsmerkmale
- Qualitätsmanagement
- Qualitätssicherungsprozesse

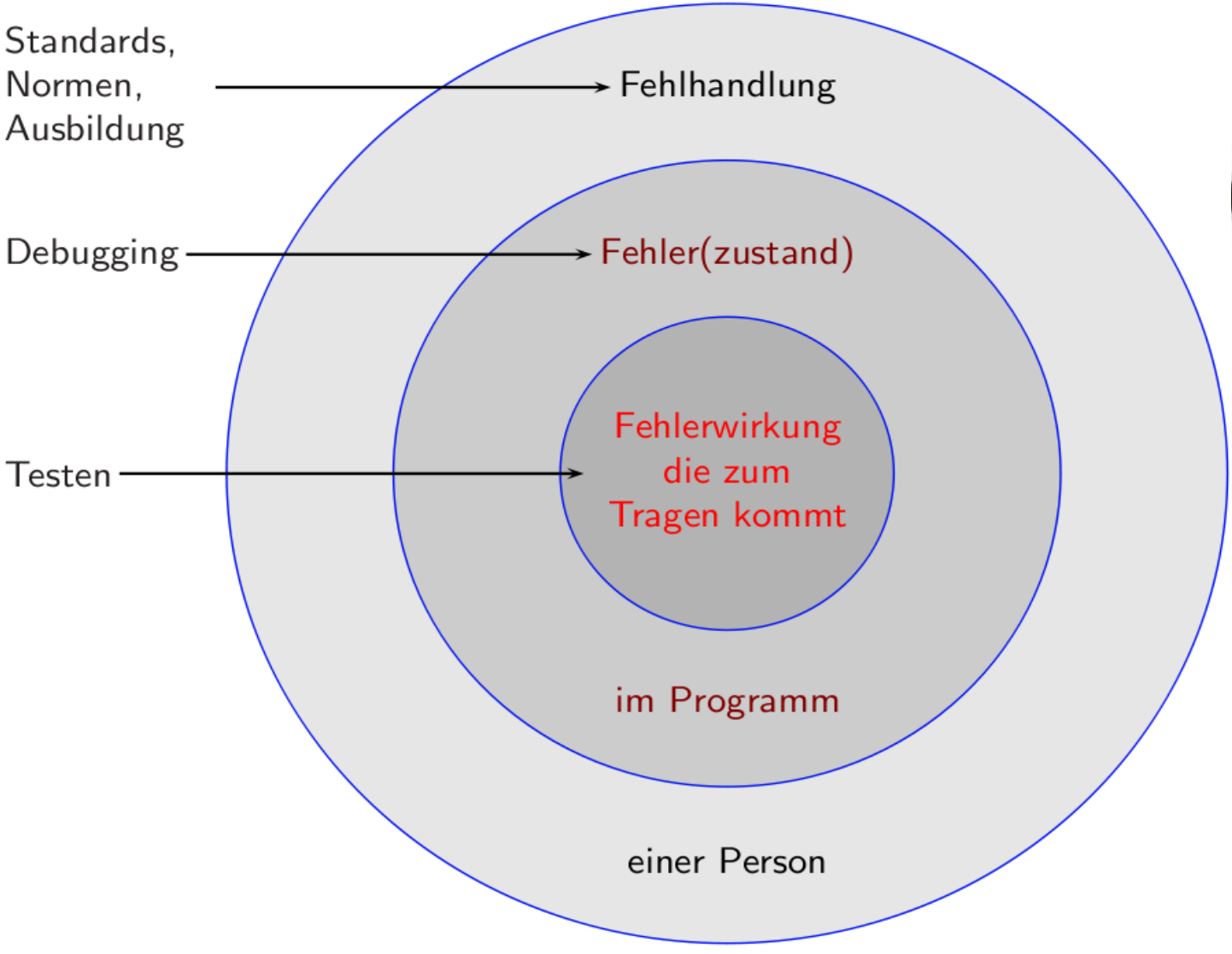


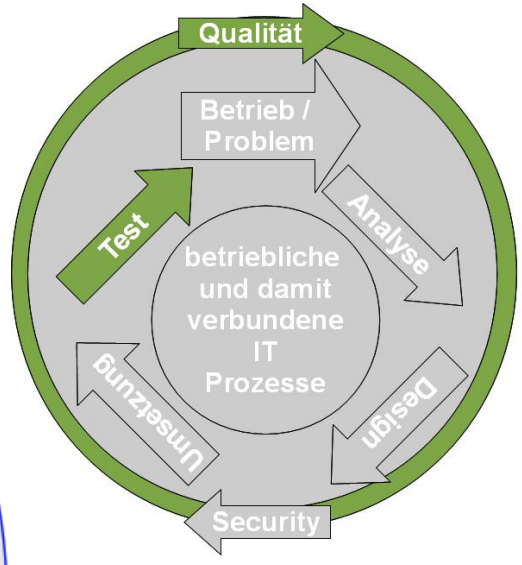
Teil 2.2: Prozess-Qualität





Teil 3.0: Grundlagen Softwaretesten

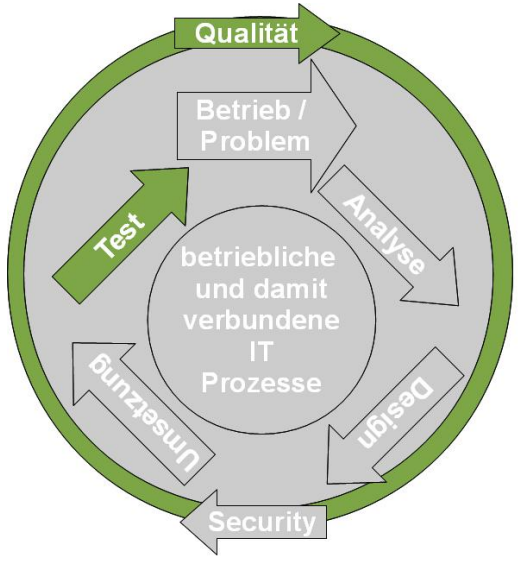




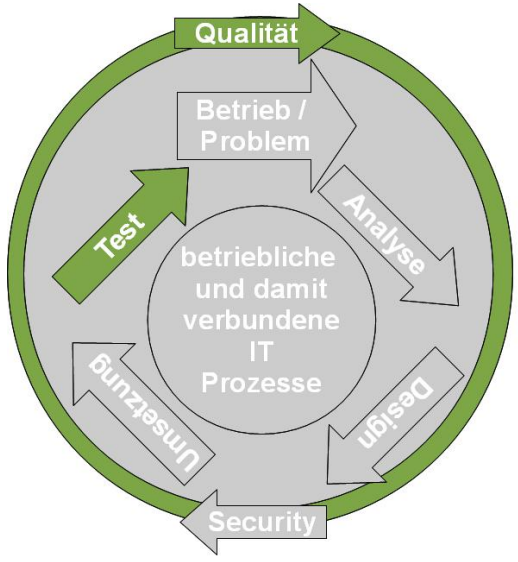
Teil 3.1: Testen im Softwarelebenszyklus

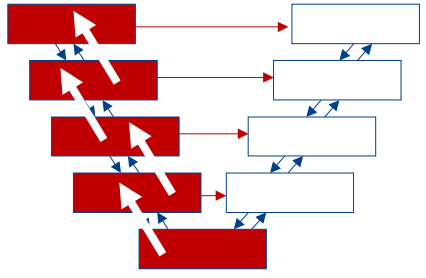
LEHRSTUHL 14
SOFTWARE ENGINEERING

- **Fehler in Softwaresystemen**
- Vorgehensmodelle
- **Testverfahren**
- Einleitende Begriffe (rund ums Testen)
  - Analysierende Prüfverfahren
  - Reviews, Inspektionen, Walkthroughs
  - statische und dynamische Tests



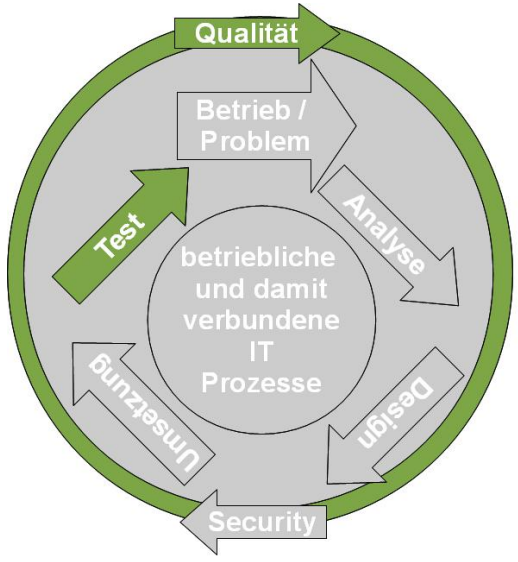
- **Testobjekt:** Gegen Dokument aus vorhergehender Konstruktionsphase „**verifizieren**“.
- Bei informellen Dokumente durch **Review.**
- Bei formalen Dokumenten durch **statische Analyse.**



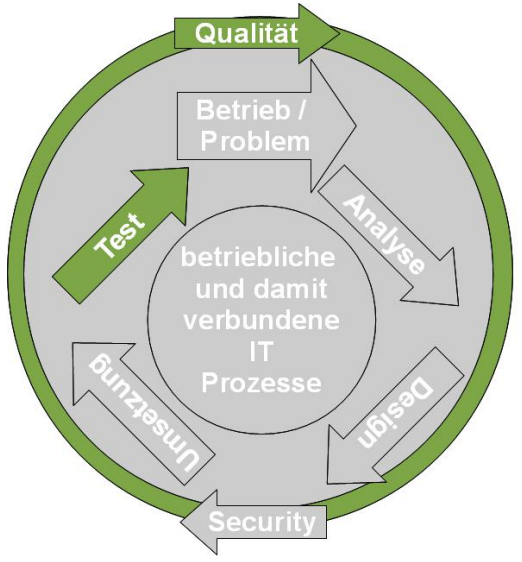


Teil 3.3: Softwaremetriken

- Metriken
- Direktes und indirektes Messen
- Vorgehensweisen
- Effekte



Teil 3.4: Black-Box-Test

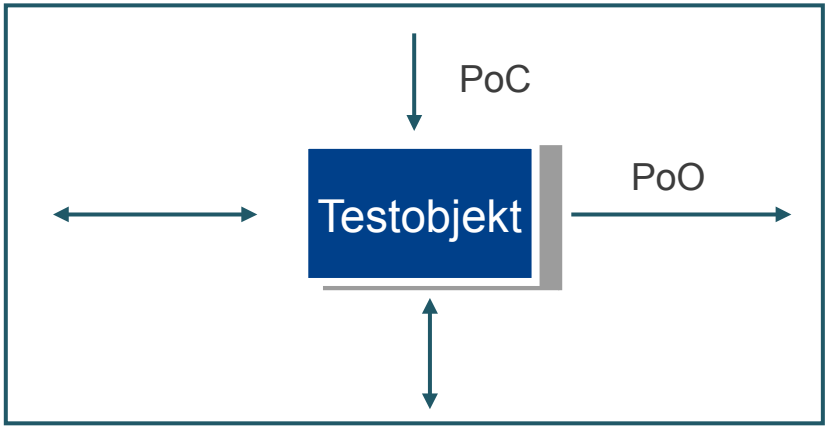


Eingabewerte

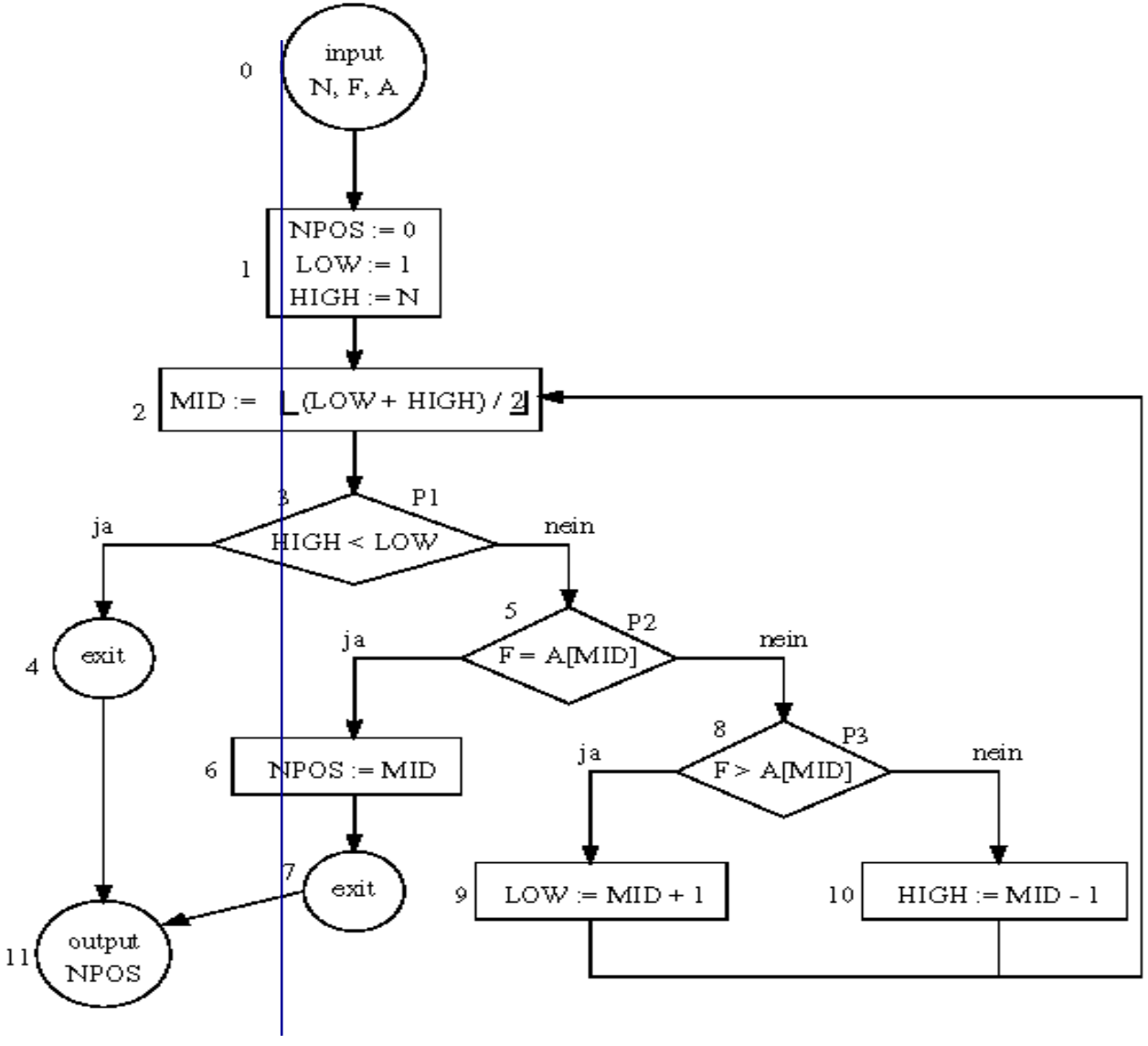
Testobjekt

Istergebnis

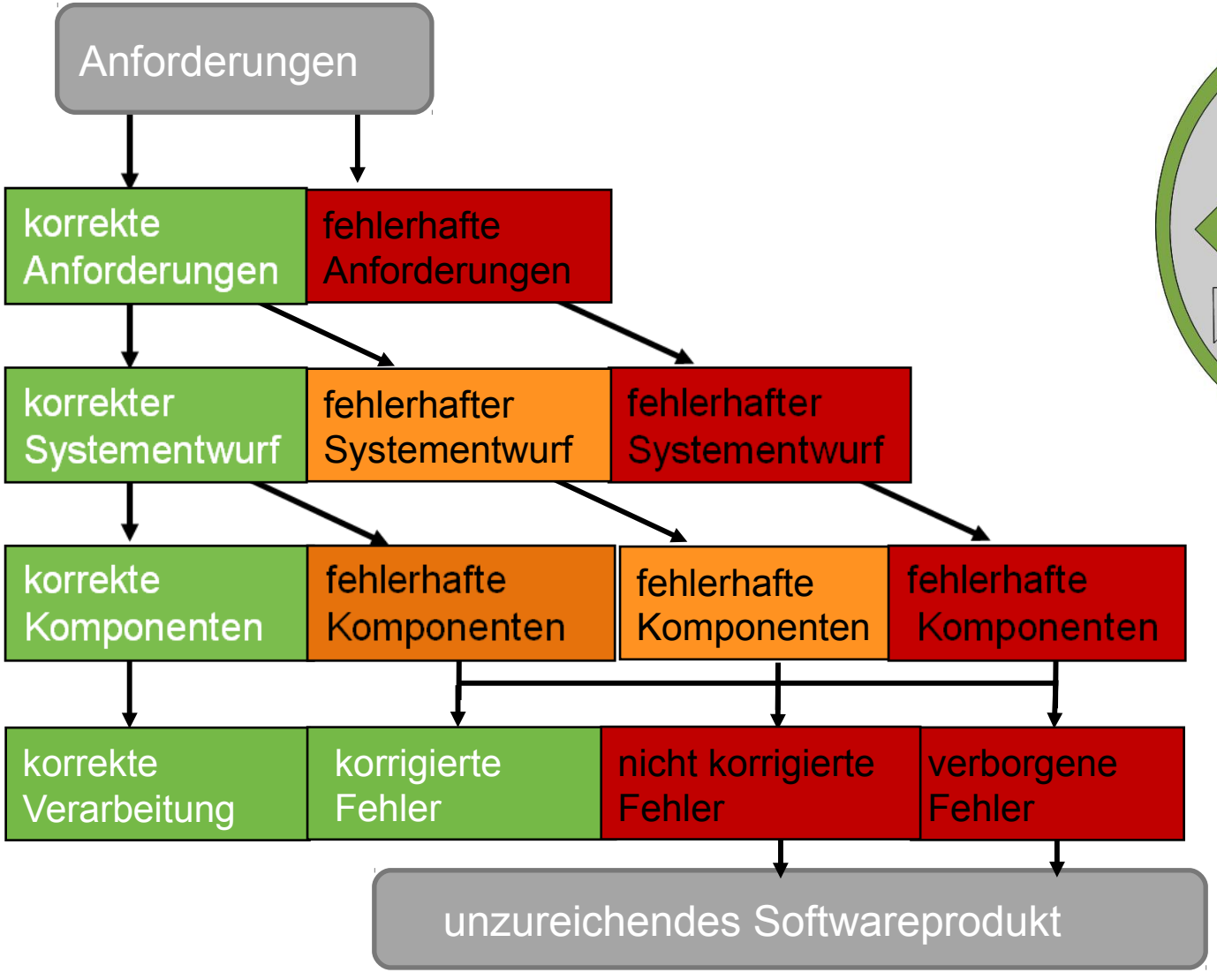
Ohne Kenntnis der Programmlogik abgeleitet

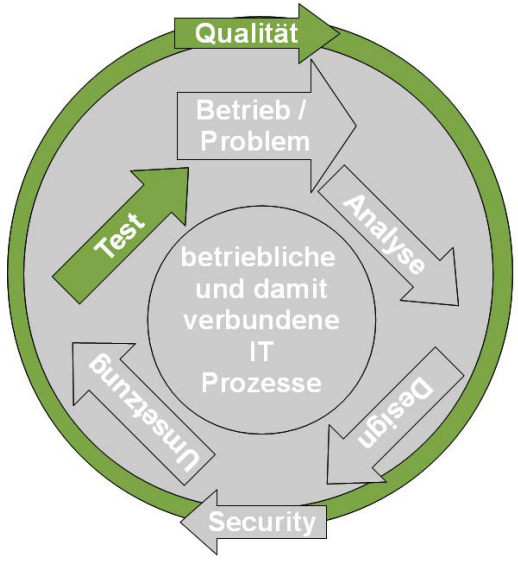


Daten- & Kontrollfluss-basiertes Testen

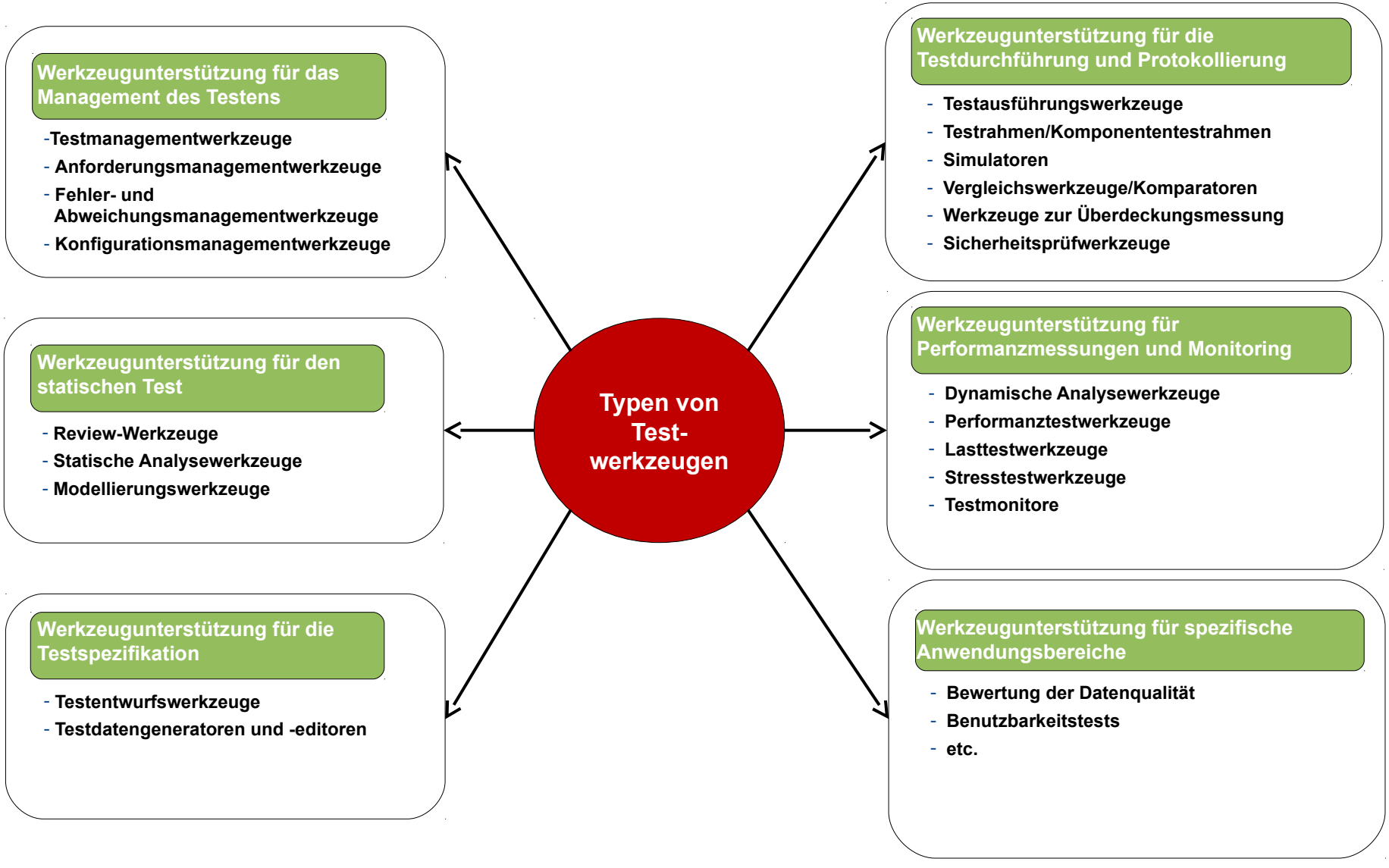


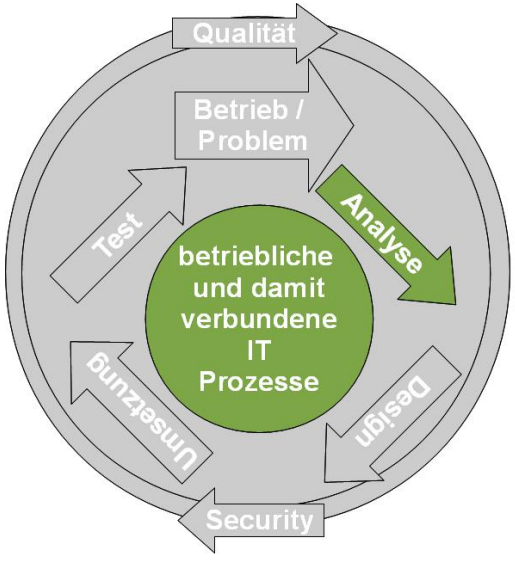
Teil 3.6: Testmanagement





Teil 3.7: Testwerkzeuge





- $\Sigma = (S,F)$ 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, 2, 4

- J. Ludewig, H. Lichter: **Software Engineering - Grundlagen, Menschen, Prozesse, Techniken**, dpunkt.verlag, 3. Auflage, 2013.
  https://www2.swc.rwth-aachen.de/se_buch .
  Ab 33 EUR (e-book):
  http://www.dpunkt.de/buecher/4501/software-engineering.html
  Unibibliothek (23 Exemplare, 3. Auflage):
  http://www.ub.tu-dortmund.de/katalog/titel/1416968 .

(Links s. a. Vorlesungswebseite)

Bei Engpässen in der Ausleihe kann **Kopiervorlage** der relevanten Ausschnitte zur Verfügung gestellt werden.

Literatur: Kapitel 3

- A. Spillner, T. Linz: **Basiswissen Softwaretest**. 4., überarbeitete Auflage, dpunkt.verlag, 2010, 308 Seiten, 39 Euro (D), ISBN 987-3-89864-642-0
  Unibibliothek (9 Exemplare): http://www.ub.tu-dortmund.de/katalog/titel/1287855 .
- Bei Engpässen in der Ausleihe alternativ ähnliche Inhalte als e-Book:
  A. Spillner, T. Linz, H. Schaefer: **Software Testing Foundations**, 3. Auflage, Rocky Nook, 2011, 296 Seiten, Print ISBN-13: 978-1-933952-78-9
  Unibibliothek (**e-Book**): http://www.ub.tu-dortmund.de/katalog/titel/1409780 .
- E. Riedemann: **Testmethoden für sequentielle und nebenläufige Software-Systeme**. Teubner, Stuttgart, 512 S., 1997
  Vollständig als PDF herunterladbar von Vorlesungs-Webseite.
  Unibibliothek (2 Exemplare): http://www.ub.tu-dortmund.de/katalog/titel/687299

(Links s. a. Vorlesungswebseite)

Agenda

- **Organisatorisches**
  - Studienordnung: Einordnung / Kompetenzen / Struktur / Prüfung
  - Vorlesung: Bildungsvertrag, Termine, Feedback
  - Übung: Konzept / Termine
  - Klausur
- **Vorstellung des Fachgebietes**
  - Forschung
  - Abschlussarbeiten, Hiwi-Jobs, weiteres Lehrangebot
- **Vorlesungsinhalte**

→ **NOCH FRAGEN ?**