

Software Engineering

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

IEEE Std. 610.12 (1990)

Methoden des Software Engineering

Nora Koch, Martin Wirsing

mit Beiträgen von Harald Störrle
in Zusammenarbeit mit
Florian Hacklinger

WS 2006/07

Vorlesung: Methoden des Software Engineering
Block R (Rahmen)

Einführung und Übersicht

Martin Wirsing

Einheit R1, 17.10.2006

Block R (Rahmen): Einführung

4. Juni 1996: Erster Start der "Ariane-5"



- Während des Flugs läuft ein unnötiges Kalibrierungsprogramm für die Trägheitssensoren. Die gemessenen Werte der Ariane-5 überschreiten die in der Ariane-4-Software vorgesehenen Bereiche.
- Die dadurch ausgelöste (Ada-)Exception wird durch Anhalten des Steuerungscomputers behandelt, um auf ein zweites redundantes System umzuschalten.
- Im zweiten System tritt der gleiche Software-Fehler auf und wird identisch behandelt.
- Kosten des Ariane-5-Programms bis 1996 ca. 7 Milliarden US-\$
- Wert des zerstörten Satelliten: ca. 500 Millionen US-\$

Block R (Rahmen): Einführung

Software-Katastrophe: Kein Einzelfall

- **Technik-Katastrophen:**
 - September 1999: Verlust der Sonde "Mars Climate Orbiter" wegen falscher Einheitenrechnung
 - 1985-1987 Therac 25 (Strahlengerät zur Krebsbehandlung): Fehlerhafte Programmierung führt zu Verbrennungen und Todesfällen
- **Finanzielle Katastrophen:**
 - 1990 AT&T Telefonverbindung zwischen Ost- und Westküste der USA wg eines SW-Fehlers für mehr als 24 Std unterbrochen: ca. 1 Mia US-\$
 - 1992: Integration des Reservierungssystems SABRE mit anderen Reservierungssystemen abgebrochen: 165 Mio. US-\$
 - 1996 Dtsche Telecom: "1. Januar 1996 ist kein Feiertag": >50 Mio DM
- **Terminkatastrophen:**
 - 1994: Eröffnung des Denver International Airport um 9 Monate verzögert wegen Softwareproblemen im Gepäcktransport-System
 - 2003: Einführung des LKW-Mautsystems in Deutschland verzögert sich um 18 Monate

Permanente Software-Krise?

- **1965: Der Begriff der *Softwarekrise* etabliert sich in Industrie und Wissenschaft.**
 - Fehler in Computersystemen sind fast immer auf Softwarefehler zurückzuführen
 - Software wird nicht termingerecht und/oder zu höheren Kosten als geschätzt fertiggestellt
 - Software entspricht oft nicht den Anforderungen ihrer Benutzer
- **Studie von 1979 zu Softwareprojekten (USA):**
 - 75% der Ergebnisse nie eingesetzt
 - 19% der Ergebnisse stark überarbeitet
 - 6% benutzbar.
- **Studie von 1994 zu Software-Großprojekten (IBM Consulting):**
 - 55% Kostenüberschreitung
 - 68% Terminüberschreitung
 - 88% Bedarf für starke Überarbeitung

Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Was ist Software?

- **Software ist eine Sammlung von Computerprogrammen, Prozeduren, Regeln, zugehöriger Dokumentation und Daten [IEEE Std. 729-83]**
- **Software ist eine Menge von Programmen oder Daten zusammen mit begleitenden Dokumenten, die für ihre Anwendung notwendig oder hilfreich sind [Hesse et al. 1984]**
- **Verschiedene Sichten auf Software:**
 - Innensicht: Software als System
 - Außensicht: Software als Produkt

Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Innensicht der Entwickler: Software als System

- **System**
Menge von Elementen, zwischen denen bestimmte Beziehungen bestehen [Duden 1974]
- **Softwaresystem**
 - Basissoftware: Schnittstelle zur Hardware (insbes. Betriebssysteme wie Unix, Windows, ...), Datenbanken, Kommunikationsprotokolle
 - Middleware: Plattform insbesondere für verteilte Anwendungen (Workflow-Management, CORBA, Grid-Plattform,...)
 - Anwendungssoftware zur Lösung kundenspezifischer Aufgaben
- **Software-intensives System**
System, das Softwarekomponenten enthält und in dem Software eine große Rolle spielt (eingebettete Systeme in Zügen, Autos; verteilte Anwendungen wie virtuelle Firmen, Geschäftsnetzwerke; mobile und ortsbezogene Informationsdienste; Webdienste, ...)

Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Außersicht der Anwender: Software als Produkt

- **Produkt**
Ein in sich abgeschlossenes, i.a. für einen Auftraggeber bestimmtes Ergebnis eines erfolgreich durchgeführten Projekts oder Herstellungsprozesses [Balzert 2000]
- **Softwareprodukt**
 - **Individualsoftware** entwickelt für einen Auftraggeber (z.B. Kundenverwaltung, Hubschraubersteuerung)
 - **Standardsoftware** entwickelt für Anwenderklasse, z.B. Büro (MS-Office), Logistik (SAP R/3),
 - **Typische Systemarten**
 - Informationssysteme (Sachbearbeitung)
 - Eingebettete Systeme (Steuerungen)
 - Produktionsplanung und -steuerung
 - Telekommunikation (Vermittlungssysteme)
 - E-Business (Web-Systeme)

Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Besonderheiten von Software

- Software ist **immateriell**.
- Software unterliegt **keinem Verschleiß**.
- Es gibt **keine Software-Ersatzteile**:
Defekte sind immer Konstruktionsfehler.
- Software ist **schwer zu vermessen** („Technische Daten“ von Software?).
- Software gilt als relativ **leicht änderbar**
(im Vergleich zu materiellen technischen Produkten).
- Software unterliegt einem **ständigen Anpassungsdruck**.
- Software **altert**.

Software is a hybrid, halfway between an abstract idea and a physical, tangible thing. Software is neither land nor sea, but swamp: a hybrid too thin for the (software engineering) and too thick for the navy (computer science).
Brad Cox

Software-Produktqualität (ISO 9126/DIN 66272)

- | | |
|---|--|
| <ul style="list-style-type: none"> ▪ Funktionalität <ul style="list-style-type: none"> ▪ Angemessenheit ▪ Sicherheit ▪ Genauigkeit der Berechnung ▪ Interoperabilität ▪ Konformanz zu Standards ▪ Zuverlässigkeit <ul style="list-style-type: none"> ▪ Reife ▪ Fehlertoleranz ▪ Wiederherstellbarkeit ▪ Benutzbarkeit <ul style="list-style-type: none"> ▪ Verständlichkeit ▪ Erlernbarkeit ▪ Bedienbarkeit | <ul style="list-style-type: none"> ▪ Effizienz <ul style="list-style-type: none"> ▪ Zeitverhalten ▪ Verbrauchsverhalten ▪ Änderbarkeit <ul style="list-style-type: none"> ▪ Analysierbarkeit ▪ Modifizierbarkeit ▪ Stabilität ▪ Prüfbarkeit ▪ Übertragbarkeit <ul style="list-style-type: none"> ▪ Anpassbarkeit ▪ Installierbarkeit ▪ Konformanz zu Standards ▪ Austauschbarkeit |
|---|--|

Was ist Software Engineering?

▪ Eine naive Sicht:

Problem Spezifikation → *Kodierung* → **Programm**

▪ Aber ...

- Wo kommt die **Spezifikation** her?
- Woher weiß man, dass die Spezifikation die **Nutzerbedürfnisse** erfüllt?
- Wie wurde die **Struktur des Programms** bestimmt?
- Woher weiß man, dass das Programm die **Spezifikation erfüllt**?
- Woher weiß man, dass das Programm **immer zuverlässig arbeiten** wird?
- Was macht man, wenn der Anwender **Änderungen benötigt**?
- Wie strukturiert man die Programmentwicklung in **Unteraufgaben**, wenn ein **Team von Entwicklern** zur Verfügung steht ?

Software Engineering (I)

Einige Definitionen und Aspekte

- *The establishment and use of sound engineering principles in order to obtain economically software that is reliable and runs on real machines.*
F.L. Bauer, NATO-Konferenz Software-Engineering 1968
- *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*
IEEE Std. 610.12 (1990)

Also:

- Bereitstellung und systematische Anwendung von Methoden, Verfahren und Werkzeugen zur Entwicklung, Betrieb und Wartung von Software.

Software Engineering (I)

“state of the art of developing quality software on time and within budget”

- Trade-off zwischen Perfektion und physischen Grenzen
 - SE muss sich mit Problemen der realen Welt auseinandersetzen
- Stand der Kunst!
 - Die Gemeinschaft entscheidet über „Beste Praxis“
 - Lebenslange Weiterbildung

Software Engineering (III)

“multi-person construction of multi-version software”

David L.Parnas

- Teamarbeit
 - Aspekt der Größenordnung (“Gut programmieren” ist nicht genug)
 - Kommunikationsaspekt
- Erfolgreiche Softwaresysteme müssen sich weiterentwickeln oder sie werden verschwinden
 - Änderung ist die Norm, nicht die Ausnahme

Software Engineering (IV)

“software engineering is different from other engineering disciplines”
Ian Sommerville

- Nicht durch physikalische Gesetze eingeschränkt
 - Die Grenze ist der menschliche Geist
- Aber beschränkt durch politische Kräfte
 - Interessen der Beteiligten abwägen und ausgleichen

Methoden des Software Engineering: WS 2003/04

Dozenten:

Dr. Nora Koch, Prof. Dr. Martin Wirsing

Übungsleitung:

Dipl.Inf. Florian Hacklinger

Vorlesung strukturiert in einzelne Moduln

Folien möglichst am Vorabend der Vorlesung im Netz

<http://www.pst.informatik.uni-muenchen.de/lehre/WS0607/mse/>

Feedback

Stellen Sie **viele Fragen** während der Vorlesung!

Ohne **Rückmeldung** können wir nichts verändern!

Durchführung: Am Besten **konstruktiv & offline**.

Übungen zur Vorlesung

Voraussetzungen:

Gute Programmierkenntnisse (möglichst Java)

Übungsbetrieb:

Wöchentliche Übungen, Mi 14-16 Uhr, Oettingenstr. 67, Raum 0.33, Beginn 2. Woche

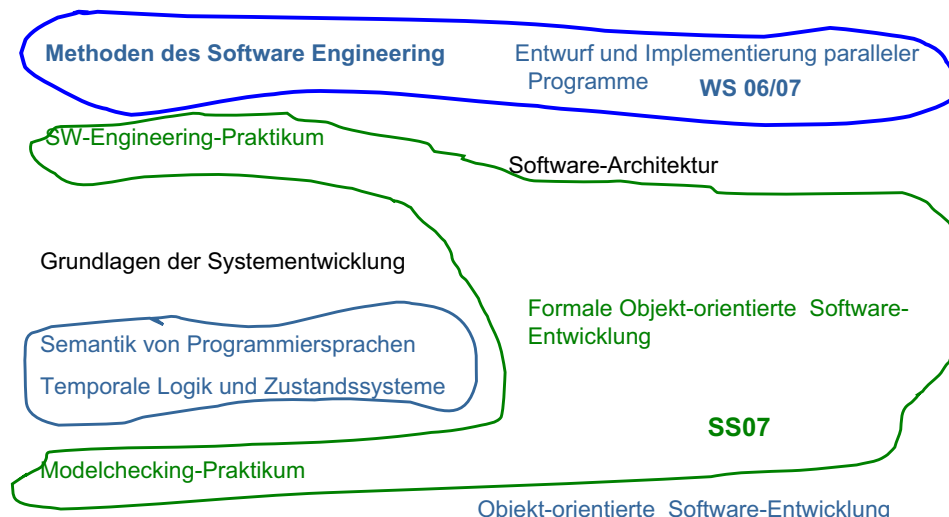
3 Hausaufgabenblätter

Klausur, 10. Februar 2007, 13:00 - 15:00 Uhr, Hauptgebäude, Raum B 201

Literatur

- Ian Sommerville: Software Engineering, 6th edition, Addison-Wesley 2000.
- Helmut Balzert: Lehrbuch der Software-Technik (2 Bände), Spektrum Akademischer Verlag 2000 und 1998.
- Weitere Literatur bei den einzelnen Moduln

Software-Engineering an der LMU

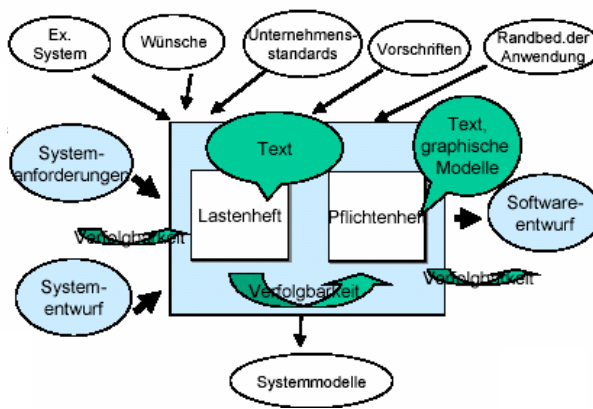


Inhalt der Vorlesung

- Block R:** Einführung und Übersicht (MW)
- Block A:** Anforderungsentwicklung (MW)
- Block B:** Software-Architektur (MW/HS)
- Block C:** Formale Methoden (MW)
- Block D:** Validierung & Verifikation (MW)
- Block E:** Software-Prozess & Projektmanagement (MW/HS)
- Block F:** Web-Engineering (NK)
- Block R:** Ausblick (MW)

Anforderungsanalyse

- Anforderungserwerb
- Konzeptuelle Modellierung
- Anforderungsvalidierung

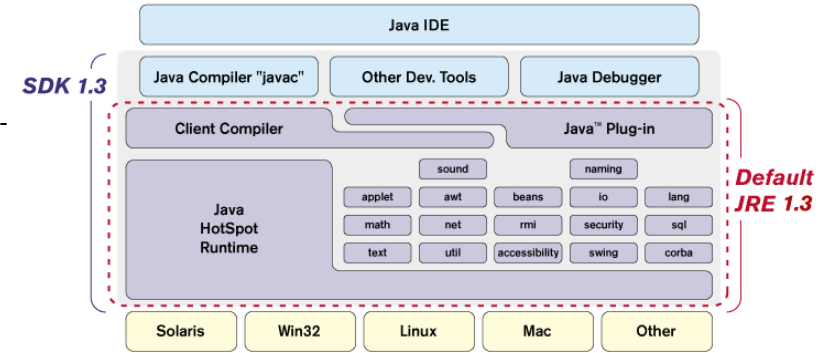


Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Software-Architektur

- Architekturentwurf
- Systemarchitektur
- Middleware
- Entwurfsmuster & Komponenten

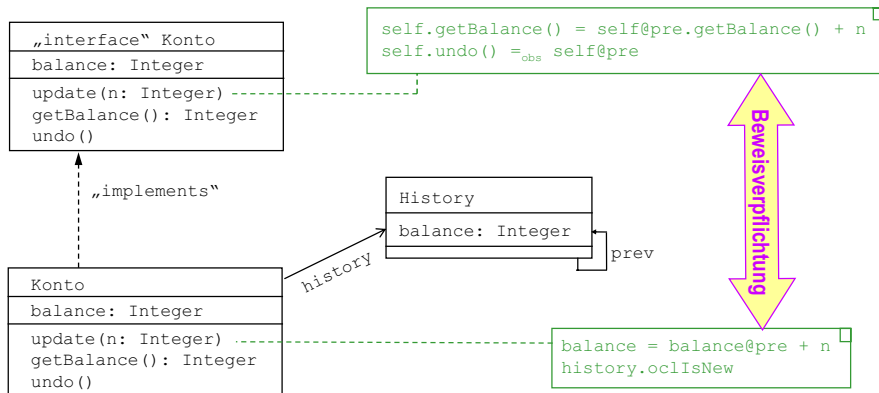
Beispiel:
Architektur einer Java-Entwicklungsplattform (informell)



Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Formale Methoden

- Funktionale Spezifikation
- Zustandsautomaten
- Eigenschaften dynamischer Systeme
- Petrinetze

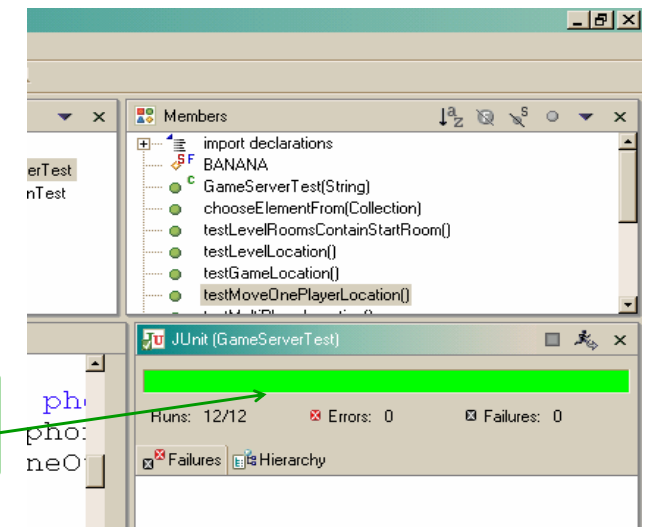


Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Validierung und Test

- Testplanung
- Blackbox- und Whiteboxtest
- Modultest mit JUnit
- Verifikation durch Modelchecking

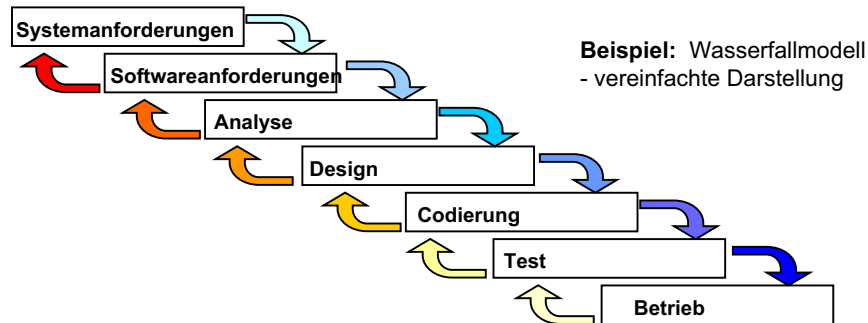
Alle Tests erfolgreich!



Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Prozess und Projektmanagement

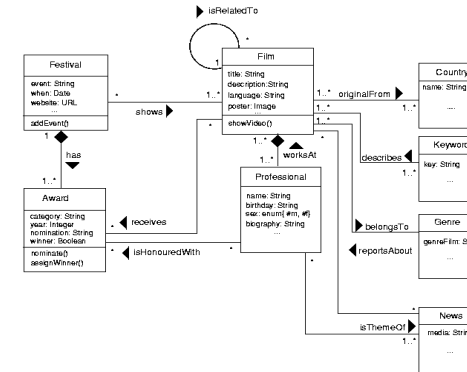
- Vorgehensmodelle
- Projektmanagement
- Prozessverbesserung
- Empirisches Software-Engineering



Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Web-Engineering

- Modellierung von Web Anwendungen
- Web-Architektur, -Implementierung und Testen
- Web-Prozessmodelle und Projektmanagement



Konzeptuelles Modell



Implementierung des Präsentationsmodells

Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München

Zusammenfassung

- Software-Produktqualität umfasst die Aspekte Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit, Übertragbarkeit
- Software Engineering bedeutet die Bereitstellung und systematische Verwendung von Methoden, Verfahren und Werkzeugen zur Entwicklung, Betrieb und Wartung von Software

Methoden des Software Engineering WS 06/07, N. Koch, M. Wirsing, LMU München