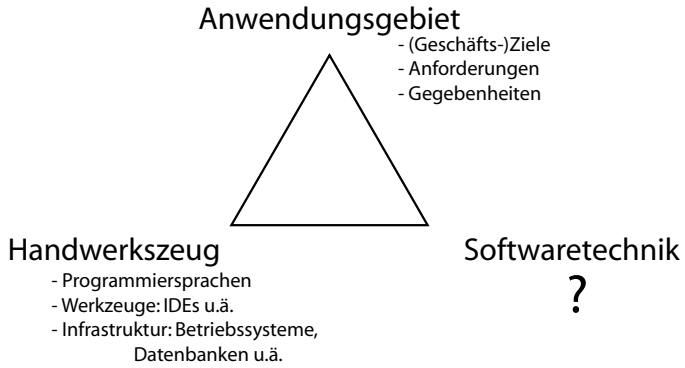
Was ist Softwaretechnik?

Prälimenaria

Worum geht es in der Vorlesung?



Literatur

- Jochen Ludewig, Horst Lichter:
  Software Engineering: Grundlagen, Menschen, Prozesse, Techniken
  dpunkt.Verlag 2007
- Wolfgang Zuser, Thomas Grechenig, Monika Köhle:
  Software Engineering mit UML und dem Unified Process
  2. Auflage Pearson Studium 2004
- Barbara Liskow, John Guttag:
  Program Development in Java
  Addison-Wesley 2001
- Hans van Vliet:
  Software Engineering: Principles and Practice
  Wiley 2000
- Ein Muss für jeden, der sich ernsthaft mit Softwaretechnik befassen möchte:
  Michael Jackson:
  Software Requirements & Specifications: a lexicon of practice, principles and prejudices
  Addison-Wesley 1995

- Jedes Lehrbuch zur Softwaretechnik

Software als industrielles Produkt

Situation der Softwareentwicklung in Deutschland

„Analyse und Evaluation der Softwareentwicklung in Deutschland“ Eine Studie für das Bundesministerium für Bildung und Forschung durchgeführt von GfK Marktforschung GmbH, Fraunhofer-Institut für Experimentelles Software Engineering IESE, Fraunhofer-Institut für Systemtechnik und Innovationsforschung ISI im Dezember 2000

Kernaussagen zu

- Marktstruktur der Softwarebranche
- Personalsituation in der Softwareentwicklung
- Technologischer Innovationsstand
- Prognosen und Empfehlungen

Marktstruktur der Softwarebranche (2000)

- Zahl der Unternehmen 19.200
  - Primärbranche 10.550,
    überwiegend kleine, junge Unternehmen
  - Untersuchte Sekundärbranchen 8.650,
    überwiegend mittlere und große Unternehmen
- Umsatz und Wertschöpfung
  - In der Primärbranche Umsatz 55,5 Mrd. DM im Jahr 1999.
  - Sekundärbranche nicht erfassbar
  - Errechnete Wertschöpfung aus der Studie: ca. 50 Mrd.
  - Anteil an der gesamten Wertschöpfung: 1,4%
  - Zum Vergleich: Landwirtschaft 42 Mrd.
- Art der Produkte nach Verwendungszweck (Anteil der Unternehmen)

Art	Gesamt	Primärbranche	Sekundärbranche
Produkt	77%	86%	53%
Teilprodukt	22%	24%	18%
embedded	24%	15%	47%

- Produkttypen in der Primärbranche
  - Betriebswirtschaftliche Software bei 55% der Unternehmen

  - Multimedia und Internet 46%
  - Technische Systeme 33%
  - Systemsoftware 16%
- Individualsoftware/Produkte (Anteil der Unternehmen)

Art	Gesamt	Primärbranche	Sekundärbranche
Individual	53%	45%	74%
Kleinserie	30%	32%	26%
Standardsoftware	30%	38%	6%

Personalsituation in der Softwareentwicklung

Trends in der nachgefragten Qualifikation:

- Analyse, Architektur, Anwendungswissen und Informatik auf hohem Abstraktionsniveau (insbesondere große Unternehmen)
- Programmierung und Anpassung von Anwendungen
- Trend zur Professionalisierung/Akademisierung

Technologischer Innovationsstand

- Dominanz von Kosten- und Terminanforderungen über Qualität
- Fokus auf frühzeitige Fehlererkennung durch Reviews
- Trend zu risikomindernden (z.B. inkrementellen) Entwicklungsprozessen
- Komponententechnologie
- Internet
- Wachsender Bestand an „Legacy Software“
- Nur 22% der Unternehmen forschen
- Kooperationen mit Hochschulen und Forschungseinrichtungen unter dem Niveau der USA

Prognosen und Empfehlungen der Studie

Wichtige Entwicklungsgebiete für Software

- Software als Bestandteil von Produkten (eingebettete Software, Maschinenbau, etc.)
- Software zur Unterstützung von Dienstleistungen (Banken, Versicherungen, etc.)
- Software für Infrastruktur (Verkehrsleitsysteme, Telekommunikation, etc.)

Aufgabenfelder für Softwaretechnik

- Verbesserung der Kompetenz der Unternehmen: Prozesse, Verfahren, Werkzeuge und Methoden
- Grundlagen der Softwaretechnik als Ingenieursdisziplin
- Innovative Techniken

Thematische Schwerpunkte für die Zukunft

- Wiederverwendbarer Softwarekomponenten („Componentware“) auf Basis langlebiger Softwarearchitekturen
- Hohe Software-Qualität
- Software-Varianten
- Räumlich verteilte Entwicklung
- Konfigurierbarkeit und Skalierbarkeit in heterogenen Systemen
- „Knowledge Management“
- „Human Centered Engenieering“
- Netzinfrastrukturen
- Anforderungsanalyse, „Requirements Engineering“
- Innovative Softwaretechnik wie Extreme Programming, etc.

Die Studie ist unter folgender Adresse zu finden: www.isi.fhg.de/publ/downloads/isi00b69/software.pdf

Aktuelle Zahlen aus der Statistik von BITKOM

Siehe Tabelle 1 und Tabelle 2.

Jahr	**Umsatz in Mrd. Euro**
2008	14,8
2009	14,3
2010	15,4
2011	16,2
2012	17,0

Tabelle 1: Umsatz Softwarebranche (Quelle: BITKOM)

Jahr	Beschäftige in Tsd.
2007	524,3
2008	551,3
2009	563,5
2010	588,2
2011	605,1

Tabelle 2: Beschäftigte in Software und IT-Services (Quelle: BITKOM)

Qualität von Software

Qualitätsmerkmale von Software

Funktionalität Die Software erfüllt die Aufgaben, derentwegen sie konstruiert wurde.

Benutzbarkeit (*usability*) Erlernbarkeit, Angemessenheit, Softwareergonomie

Verfügbarkeit(*availability*) Zuverlässigkeit, Ausfallsicherheit

Leistungsfähigkeit (*performance*) Durchsatz und Antwortzeit, Skalierbarkeit

Sicherheit (*security*) Vertraulichkeit, Unverletzlichkeit

Änderbarkeit (*modifiability*) Wartbarkeit, Erweiterbarkeit, Portierbarkeit

Testbarkeit (*testability*) Überprüfbarkeit der korrekten Funktionsweise

Spektakuläre Softwarefehler

Therac-25 1985-87 Mindestens 6 Personen wurden einer überhöhten Strahlendosis ausgesetzt, weil das Bestrahlungsgerät Therac-25 wegen Softwarefehlern falsch dosierte. 3 Opfer starben wegen dieser Überdosis.

AT&T-Telefonnetz 1990 70 Millionen Ferngespräche innerhalb der USA konnten 9 Stunden lang nicht vermittelt werden.

Stellwerk Altona 1995 Nach Umstellung auf ein elektronisches Stellwerk kam es immer wieder zu Sicherheitsabschaltungen mit erheblichen Beeinträchtigungen des Verkehrs: etwa 3 Tage je 50.000 bis 100.000 Reisende betroffen.

Ariane 5 1996 Der Jungfernflug am 4.6.1996 endete mit der Sprengung der Rakete etwa 40 Sekunden nach dem Start. Ursache: Softwarefehler.

Oft wird in der Literatur von der „Software-Krise“ gesprochen. Der Begriff kam gegen Ende der 60er Jahre des letzten Jahrhunderts auf. Diese sogenannte Krise hält also schon 50 Jahre an! Und so gesehen hat sich eine von der Krise andauernd befallene Branche recht rasant entwickelt.

Man bedenke: Auch heute scheitern Software-Projekte, auch heute wird Anwendern Software zugemutet, die grundlegende Qualitätsmerkmale vermissen lässt – aber: der Leistungsumfang und die Komplexität der Software hat in den letzten 40 Jahren enorm zugenommen! Also ist „Software-Krise“ keine tatsächlich treffende Charakterisierung der Situation in der Software-Entwicklung.

(Als Beleg für die sogenannte Software-Krise wird oft der *The Chaos Report* der Standish Group von 1994 zitiert. Dieser Bericht beruht nicht wirklich auf einer soliden Untersuuchung, berichtet Robert L. Glass in *The Standish Report: Does It Really Describe a Software Crisis?* in: Communications of the ACM, August 2006.)

Spektakuläre Projektverläufe

Taurus 1993 System zur automatischen Transaktionsabwicklung an der Londoner Börse. 1993 eingestellt, 5 Jahre Entwicklungszeit. Projektkosten 75 Mio Pfund, Schaden für Kunden ca. 450 Mio Pfund

Denver 1995 Probleme mit der Software der Gepäckverteilungsanlage verzögern die Eröffnung des neuen Flughafens in Denver. Kosten 5,2 Mrd $, geplant waren 2 Mrd $.

VW/Audi 1999 VW und Audi konnten wegen Problemen bei der Einführung von SAP/R3 im zentralen Ersatzteillager in Kassel ihren Händlern nicht ausreichend Ersatzteile liefern.

Inpol-Neu 2001 Polizeiinformationssystem; Betrieb im Dezember 2001 nach 10 Minuten gestoppt, völlig unzureichende Performance; August 2003: Erfolgreicher Start eines modifizierten Systems; geschätzte Kosten durch Fehlstart: 60 Mio Euro.

LUSD 2007 Lehrer- und Schüler-Datenbank des Landes Hessen. Entwicklung bei CSC seit 1.6.06. Einsatz ab Oktober(!) 2006 bei am Ende 2000 hessischen Schulen; Inhaltliche Fehler, Datenverluste, extrem lange Antwortzeiten. Folge: erhebliche Mehrarbeit in den Schulen. Kosten (nach Angaben der Opposition): 20 Mio Euro.

CityTime 2010 Zeiterfassungssystem für die Arbeiter und Angestellten von New York City. Geplante Entwicklung: 1998 - 2003. 2010 ist das Projekt noch nicht fertig und hat 722 Millionen $ gekostet, 10 mal so viel wie geplant.

Es gibt auch andere Beispiele

Paris Métro Ligne 14

- Die fahrerlose Linie 14 der Métro in Paris verbindet Saint Lazare und Olympiades, die Strecke durchquert die Innenstadt von Paris.
- Die Software für die Zugsteuerung wurde von Siemens Transportation Systems entwickelt.
- Die Software wurde mit B, einer von Jean-Raymond Abrial entwickelten formalen Methode, entworfen. B basiert auf AMN (Abstract Machine Notation) und hat Werkzeuge für die Spezifikation, den Entwurf, die Generierung von Code (in ADA) und die Verifikation.
- Mit B wird jeder Schritt der Entwicklung des Systems *mathematisch* bewiesen.
- In der Métro Paris ist seit 1998 die Version 1.0 der damals entwickelten Steuerungssoftware im Einsatz.

Stand der Softwaretechnik

In den USA wird seit längerem diskutiert, ob die Berufsbezeichnung „Software Engineer“ staatlich lizensiert werden soll – ähnlich wie bei Bauingenieuren, Architekten etc.

Die Position der ACM (Association for Computing Machinery) ist:

> ... a software engineering license would be interpreted as an authoritative statement that the licensed engineer is capable of producing software systems of consistent reliability, dependability, and usability. The ACM Council concluded that our state of knowledge and practice is too immature to give such assurances. (Stellungnahme vom November 2002)

Gründe?

Als Gründe für Fehler in Software werden oft genannt:

- Fehler können immer passieren
- Software ist viel komplexer als jedes andere Produkt

- Softwaretechnik ist eine junge Disziplin, andere Ingenieursdisziplinen sind viel älter und reifer
- Es ist unmöglich, alle Möglichkeiten zu testen, die im laufenden Betrieb des Systems auftreten können
- Aus wirtschaftlichen Gründen muss Software unter Zeit- und Kostendruck hergestellt werden, also passieren auch Fehler – hohe Qualität wäre unbezahlbar und solch ein Projekt würde ewig dauern

Was ist von solchen „Argumenten“ zu halten?

Alles *Ausreden*!

(Die Aussage „Alles Ausreden!“ ist natürlich polemisch. Jeder der genannten Gründe trifft immer wieder zu. Nur: eine Mentalität, die davon ausgeht und akzeptiert, dass Software Fehler hat, ist nicht okay, wiewohl weit verbreitet!)

Auswege?

Häufige Auswege, geglaubte und oft zitierte Irrtümer — oder: was Softwaretechnik *nicht* ist:

- *Mehr Management, bessere Abläufe im Entwicklungsprozess führen zu besserer Qualität von Software*

  In Wahrheit: Management kann eine Kultur fördern, in der Qualität gebaut wird. Aber Qualität herbei managen kann man nicht!
- *Das Werkzeug, die Methode — dann klappen die Projekte*

  In Wahrheit: Eine Methode, die für *alles* taugt, taugt für ein spezielles Problem nicht besonders gut, also für *nichts*. Denn jedes Problem ist ein spezielles Problem!
- *Wenn genug Leute, wie bei Open-Source-Projekten, auf den Code schauen, dann werden die Fehler schon gefunden werden*

  In Wahrheit: Es liegt schon am Code selbst und am Design, ob man ihn überprüfen kann. Obskure Systeme können auch von noch so viel Augen nicht durchschaut werden. Das gilt auch für Open Source!

Fragen wir also:

Was ist Softwaretechnik?

Definitionen

Definition der IEEE Computer Society

> (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
>
> (2) The study of approaches as in (1)
>
> (IEEE Standard Glossary of Software EngineeringTerminology)

Definition von Jochen Ludewig und Horst Lichter

> Software Engineering ist jede Aktivität, bei der es um die Erstellung oder Veränderung von Software geht, soweit mit der Software Ziele verfolgt werden, die über die Software selbst hinausgehen.

Etwas polemisch zusammengefasst: Jede Art Erstellen oder Pflegen von Software, die nicht gerade eine Übungsaufgabe für die Veranstaltung Softwaretechnik oder Programmieren I ist, also eine in der wirklichen Welt nutzbare Funktionalität bietet, ist Software Engineering. (Nebeneinsicht: Hochschulveranstaltungen sind nicht Bestandteil der wirklichen Welt.)

Definition von Helmut Balzert

> Zielorientierte Bereitstellung und systematische Verwendung von
>
> - Prinzipien
> - Methoden
> - Konzepten
> - Notationen und
> - Werkzeugen
>
> für die arbeitsteilige, ingenieursmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen. Zielorientiert bedeutet die Berücksichtigung z.B. von Kosten, Zeit und Qualität.

In dieser Definition werden als die Mittel des systematischen Vorgehens der Softwaretechnik genannt:

Prinzipien sind Grundsätze, die man seinem Handeln zugrundelegt.
Beispiel: Abstraktion, Modularisierung etc.

Methoden sind planmäßig angewandte, begründete Vorgehensweisen zur Erreichung von festgelegten Zielen (im allgemeinen im Rahmen festgelegter Prinzipien).
Beispiel: Vorgehen nach der Methode der OO Analyse.

Verfahren sind ausführbare Vorschriften oder Anweisungen zum gezielten Einsatz von Methoden. (Balzert verwendet Methoden und Verfahren synonym.)
Beispiel: Rational Unified Process (RUP)

Konzepte sind die Grundvorstellung eines Systems, einer Software, einer Technik. Sie erlauben es, Sachverhalte unter einem oder mehreren Gesichtspunkten zu modellieren.

Notationen werden zur Beschreibung von Konzepten verwendet. Sie haben eine festgelegte Syntax und Semantik (Bedeutung der Symbole). Beispiel: UML (die freilich mehrere Semantiken hat!)

Werkzeuge dienen der automatischen Unterstützung von Methoden, Verfahren, Konzepten und Notationen
Beispiel: Together, Rational Rose, Eclipse

Was bedeutet „ingenieursmäßig“?

Darunter versteht man drei Merkmale:

1. Systematische und kreative Anwendung wissenschaftlicher Grundlagen, um einen nützlichen Effekt zu erreichen (Unterschied zu Wissenschaft)
2. Das Einbeziehen wirtschaftlicher Abwägungen
3. Berechenbare und garantierbare Qualitätseigenschaften (auch bezogen auf die Wirtschaftlichkeit)

Was gehört alles zum Thema Softwaretechnik

- Software-Anforderungsanalyse
- Softwaredesign
- Softwarekonstruktion (Implementierung, Codierung)
- Testen
- Softwarewartung
- Konfigurationsmanagement
- Softwarequalitätsanalyse
- Management des Softwareengineerings
- Infrastruktur des Softewareengineerings
- Softwareengineeringprozess

Plan der Vorlesung

1. Was ist Softwaretechnik?

2. Elementare Softwaretechnik
3. Softwareabstraktionen
4. Objektorientierte Abstraktionen
5. Vorgehensmodelle für den Softwareentwicklungsprozess
6. Domänenanalyse
7. Analysemuster
8. Anforderungsanalyse
9. Entwurf
10. Entwurfsmuster
11. Test

Lektüre:

Martin Glinz: *Software-Entwicklung und -Pflege als Problem*
https://files.ifi.uzh.ch/rerg/amadeus/teaching/courses/software_engineering_hs10/skript/Kapitel_01.pdf

Jochen Ludewig, Horst Lichter: *Kapitel 2 - 5 des Buches*

Wolfgang Zuser u.a.: *Kapitel 1 des Buches*

Burkhardt Renz
Technische Hochschule Mittelhessen
Fachbereich MNI
Wiesenstr. 14
D-35390 Gießen

Rev 3.0 – 12. März 2012