Softwaretechnik

Prof. Dr. Wolfgang Weitz

Organisatorisches

- **Prüfungsmodalitäten:**
  - Klausur (PL) über gesamtes Modul
  - Praktikum (SL): Projekt mit Zwischenabgaben
- **Sprechstunde:**
  - Raum 22, einfach vorbeikommen (oder vereinbaren)
  - E-Mail: wolfgang.weitz@hs-rm.de
- **Folien-PDFs** etc. im read.MI (nach der Vorlesung)
- **Laptops** falten.

Organisatorisches zum Praktikum

- Praktikum:
  - Übungsaufgaben zum Einstieg, dann
  - Kleingruppen-Projekt mit Zwischenabgaben
  - Gruppenbildung innerhalb der Praktikumsgruppen (nicht übergreifend)

- Heute: Git-Einführung
  - Jeder sollte einen persönlichen Git-Ordner (Informatik-Loginname) haben unter `https://scm.mi.hs-rm.de/rhodecode/stud/jbiff017`
  - Falls nicht: Bitte persönlichen Git-Ordner bestellen mit `https://www.mi.hs-rm.de/~weitz/cgi-bin/persrepo.cgi`

Literatur (1)

Chris Rupp,
„Requirements-Engineering und -Management
Aus der Praxis von klassisch bis agil“,
Hanser-Verlag 2014

Gernot Starke,
„Effektive Softwarearchitekturen
Ein praktischer Leitfaden“,
Hanser-Verlag 2020

Literatur (2)

Chris Rupp et al.,

„UML2 glasklar - Praxiswissen für die UML Modellierung“

4. Auflage 2012, Hanser

Erich Gamma et al.,

„Entwurfsmuster“

mitp, 2014

Literatur (3)

Tom DeMarco et al.,

„Wien wartet auf Dich!“

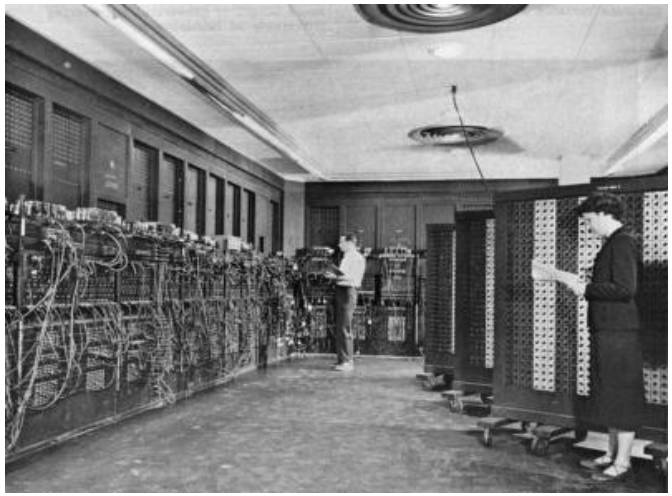
Hanser 2014, 3.Auflage

Don't

Was ist Softwaretechnik?

- Was verstehen Sie unter „Softwaretechnik“?
- Geht es um...
  - neue **Programmiersprachen**?
  - clevere Programmier**tricks**?
  - „Kochrezepte“, mit denen man die Programmieren I / II-Aufgaben (noch) schneller gelöst hätte?
  - das Finden / den Entwurf effizienter **Algorithmen**?
  - Methoden, wie man **ohne Nachdenken** von der Programmieraufgabe zum Programmcode kommt?

Woher kommt der Begriff?



- **Frühe Computersysteme:**
  - sehr teure Hardware
  - kleiner Speicher
  - → "überschaubare" Software
- **60er Jahre: "Softwarekrise"**
  - **Leistungsfähigere** Hardware ermöglicht
  - Entwicklung **komplexerer** Softwaresysteme, dabei gab es...
    - verspätete Fertigstellung von Software
    - schlechte Qualität
    - mangelnde Wartbarkeit
    - Entwicklungskosten zu hoch
- **1968: NATO Science Committee Tagung** in Garmisch
  - Ziel: "Ingenieurmäßige Softwareentwicklung"
  - Begriff "Software Engineering" geprägt (dt: Softwaretechnik)

http://homepages.cs.ncl.ac.uk/brian.randell/NATO/index.html

Softwaretechnik ist ...

*The Establishment and use of sound **engineering principles** in order to obtain **economically** software that is **reliable** and works **efficiently** on real machines.*

- Friedrich Ludwig Bauer, deutscher Informatiker (1968)

Softwaretechnik ist ...

*Software engineering is the application of a **systematic**, **disciplined**, **quantifiable** approach to the development, operation, maintenance, and retirement of software, that is, the application of **engineering** to software.*

IEEE Standard Glossary of Software Engineering Terminology (1990)

Softwaretechnik ist ...

Software-Technik: *Zielorientierte Bereitstellung und systematische Verwendung von* ***Prinzipien***, ***Methoden***, ***Konzepten***, ***Notationen*** *und* ***Werkzeugen*** *für die arbeitsteilige*, ***ingenieurmäßige*** *Entwicklung und Anwendung von umfangreichen Software-Systemen.* ***Zielorientiert*** *bedeutet die Berücksichtigung von Kosten, Zeit, Qualität.*

Helmut Balzert (1996)

Schlüsselbegriffe

- **"ingenieurmäßig"**
  - Methoden, Konzepte und Werkzeuge
  - systematisch, diszipliniert, umfangreiche Projekte
- **Kosten- / Zeit- / Qualitäts-Ziele**
  - Messbarkeit, Planbarkeit; ökonomisch
  - Zuverlässigkeit, Benutzerfreundlichkeit, ...
- **arbeitsteilig**
  - Koordination / Management von Teams
  - Berücksichtigung / Einbeziehung von Nicht-Technikern (z.B. *fachliche* Experten des Auftraggebers)
  - Dokumentation und Schulung

50 Jahre später

Seit 1968 ist viel Zeit vergangen, heute haben wir das alles im Griff!

Haben wir?

Flughafen Denver (1992)

- Automatisches **Gepäcktransportsystem**
  - 4000 Transportwagen
  - 100 vernetzte Rechner
  - Kosten: 193 Mio US$
  - Ziel: Eröffnung Oktober 1993

- **Verzögerung** der Flughafeneröffnung, weil die Softwarefertigstellung sich verspätete
- Im Testbetrieb ging **Gepäck verloren**, wurde **fehlgeleitet** oder **beschädigt**
- Beschluss August 1994: Konventionelles System zusätzlich (51 Mio US$)
- **Resultat:**
  - Flughafeneröffnung um ca 2 Jahre verzögert
  - **Verlust: 1.1 Mio US$ pro Tag**

Paul Smith, U Cambridge

Krankenwagen London ('92)

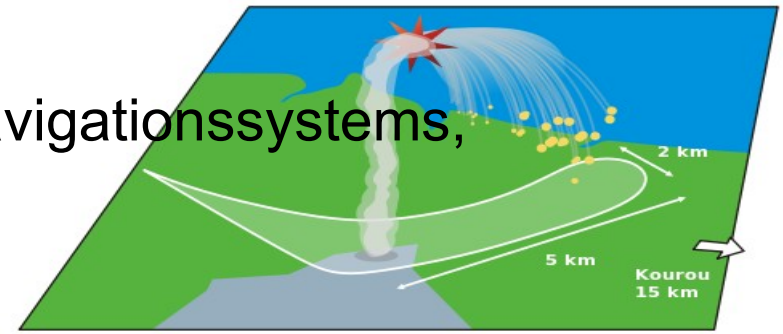
- Neues System zur **Einsatz-Koordination** von **Krankenwagen** bei Notrufen
- Ergebnis: Nach wenigen Tagen im Betrieb Abbruch:
  - **falsche Anzahl** Krankenwagen wurde losgeschickt (zu viele, keiner...) ohne Möglichkeit der Überprüfung
  - **Meldungen** des Systems konnten nach "Wegscrollen" aus dem Bildschirmfenster nicht zurückgeholt werden
  - **Systemverlangsamung** bis zur Unbrauchbarkeit (→Neustart)
  - System-**Crash**, aber
  - das **Reserve**-System funktionierte auch nicht (nie getestet)
  - angeblich um die 20 **Tote**

http://www0.cs.ucl.ac.uk/staff/A.Finkelstein/las/lascase0.9.pdf

Ariane 5 (1996)

- Explosion bei Jungfernflug Juni 1996
- Grund: Navigationssystem war von Ariane 4 übernommen worden (ohne ausreichende Tests)
- Ariane 5 war **stärker** als Ariane 4, daher konnten gewisse Variablen **größere Werte** annehmen
- **Überlauf bei Konvertierung** einer 64-Bit-Float-Zahl in einen 16 Bit Integer.
- Das führte zur **Abschaltung** des Trägheitsnavigationssystems, was den **Absturz zur Folge** hatte.
- Schade: Der fehlerhafte Programmteil wurde während des Flugs **nicht benötigt**

Von DLR German Aerospace Center - Raumfrachter ATV-4 "Albert Einstein" Ariane 5ES Rollout_5, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=26533880



Von Phrd - Eigenes Werk, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=6376804

http://java.sun.com/people/jag/Ariane5.html

September 1983

Stanislaw Petrow und das Geheimnis des roten Knopfs

Markus Kompa 20.06.2009

Was geschah wirklich im September 1983?

Historiker in Ost und West sind sich heute weitgehend darüber einig, dass die **riskanteste Phase des Kalten Kriegs** der Herbst 1983 markierte. Während dieser zwischen den Supermächten denkbar gespannten Situation ereignete sich im russischen Kontrollzentrum zur Früherkennung amerikanischer Angriffe ein Vorfall, der dem amerikanischen Experten Bruce Blair zufolge die Menschheit am nähesten an einen Atomkrieg gebracht hatte: So hatte eine **Computermeldung über anfliegende Interkontinentalraketen** binnen Minuten eine Entscheidung über einen Gegenschlag erforderlich gemacht. Der diensthabende Offizier Stanislaw Petrow bewahrte Nerven und bewertete die plausible Information des als zuverlässig geltenden Systems aus einem Bauchgefühl heraus als **Fehlalarm**. (...)

http://www.heise.de/tp/artikel/30/30488/1.html

F-16

- F-16 Mehrzweck-Kampfjet
- „*fly-by-wire*“, der Bordcomputer verhindert gefährlich abrupte Lenkbewegungen

- ...bei Flug über den Äquator dreht sich das Flugzeug schlagartig auf den Rücken
  (vorab bei Simulation entdeckt)

- Testpilot probierte, auf dem Boden im Stand das Fahrwerk einzufahren
  - ...hat funktioniert :-)

ACM Forum on Risks to the Public
in Computers and Related Systems
http://catless.ncl.ac.uk/Risks/3.44.html
(Bill Janssen / Aug 1986)

USS Yorktown (1998)

- Lenkwaffenkreuzer USS Yorktown
- Irrtümliche **Eingabe einer Null**
- führte zu einer "**Division durch Null**", die
- letztlich die **Abschaltung des Antriebssystems** auslöste.
- Das Schiff lag mehrere Stunden **bewegungsunfähig** im Wasser
- ... wegen einer fehlenden Eingabedaten-Überprüfung.

https://www.wired.com/1998/07/sunk-by-windows-nt/
„Sunk by Windows NT“, wired 07.24.98
http://archive.wired.com/science/discoveries/news/1998/07/13987

Hartz IV, Anfang 2005

„Vom Start weg sorgte das Verwaltungsprogramm für das Arbeitslosengeld II für Ärger bei Bedienern und Kunden. Wie jetzt bekannt wurde, überweist die Software zuviel Geld an die Krankenkassen. Experten rufen bereits nach einem totalen Neubeginn.“

http://www.dradio.de/dlf/sendungen/computer/417244/

- 25 Mio Euro zu viel an Krankenkassen überwiesen - und zwar jeden Monat
- Herstellerfirma Mai 2005 aus Verträgen ausgestiegen, Aufnahme der Arbeit durch Nachfolger schwierig,
- allein 6300 Manntage zum Beheben von Fehlern im aktuellen Stand geschätzt.
- Beispiel: Nutzung „hinterlegter Festwerte“ (= hardcoded), keine zentrale Änderbarkeit wichtiger Werte.

Warum?

Autsch!

Einzelfälle?

- Standish Group CHAOS Report (seit 1994):
- >25.000 Projekte untersucht, ca 5000 pro Jahr

- Stand 2015:
  - **19%** der Projekte werden vor Abschluss **abgebrochen**
  - **52%** waren (erheblich) **teurer** als geplant, **verspätet** oder **mit reduziertem Funktionsumfang**
  - **29%** innerhalb des Zeit-/Kostenrahmen mit geplanter Funktionalität
- Kleine Unternehmen waren etwas besser als große.

- Hinweis: Es gibt unterschiedliche Meinungen zur Methodik und Aussagekraft dieser Untersuchung.

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Einzelfälle?

- **Ursachen?**
  - **Unrealistische** Planung (zu groß)
  - **Unklare Anforderungen**, Änderungen/Neuplanung
  - Mangelnde **Einbeziehung** der Endanwender
  - **Termindruck**
  - Probleme mit **Management**, Entwurf und Programmierung

http://www.cs.nmt.edu/~cs328/reading/Standish.pdf

„Softwareentwicklung im Großen“

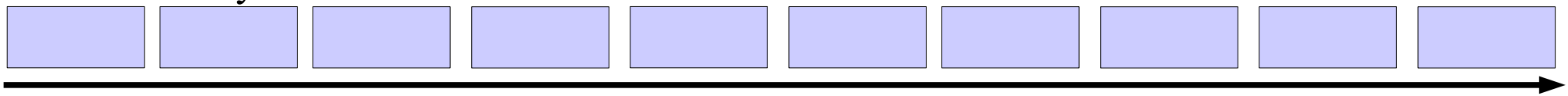
- Viele Entwickler, großer Codeumfang, lange Lebensdauer
- Damit höhere Anforderungen an...
  - (Projekt-)Management, **Planung**
  - **Benutzerfreundlichkeit** / Ergonomie
    - Leichte Handhabung, aufgabengerechte Gestaltung
  - **Zuverlässigkeit**
    - Keine Fehler/Ausfälle, auch bei Fehlbedienung (Robustheit)
  - **Wartbarkeit**
    - z.B.: Fehler ohne große Betriebsstörung beseitigbar?
  - Pflege / **Anpassbarkeit**
    - Anpassungen an geänderte Bedingungen möglich?
  - **Portabilität**
    - Übertragbarkeit der Software auf andere Rechnersysteme
  - **Dokumentation**

Lebensdauer Software/Hardware

Anwendungssoftware: Zykluszeit 10-15 Jahre

Systemsoftware: Zykluszeit 6 Jahre

Hardware: Zykluszeit 3 Jahre



[Balzert 96]

- Anwendungssystem „erlebt“ mindestens einen Systemsoftware- und mehr als zwei **Hardware-Wechsel**
- **Zielsysteme** existieren zum Zeitpunkt der Softwareerstellung noch nicht
- Anfangs **nicht vorhersehbare** Anpassungen während Betriebsdauer nötig
  z.B. Euro-Einführung, Gesetzesänderungen, technologische Innovationen (Web? Mobile? Wireless Mind Reader? ...)

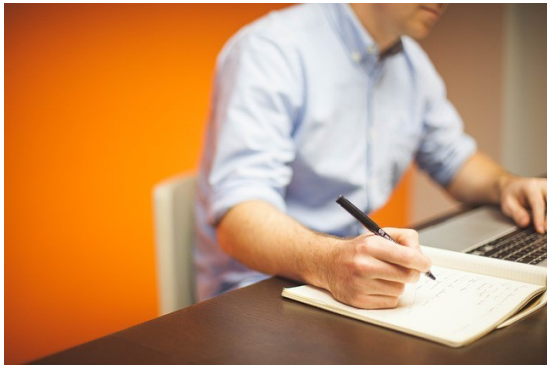
Phasen im Software-Lebenszyklus

- Von der Planung bis zur Stilllegung eines Softwaresystems werden verschiedene **Phasen** (teilweise mehrfach) durchlaufen:
- **Anforderung**saufnahme und Anforderungsanalyse
- **Entwurf**
- **Implementierung**, Test
- **Abnahme** und **Einführung**
- **Betrieb** (Wartung und Pflege)

Anforderungsaufnahme/-Analyse

- Vom Kunden oder in Gesprächen ihm werden dessen Anforderungen an das System
  - ermittelt, konkretisiert und
  - mit Quellenangaben dokumentiert;
  - **fachlich** vollständig/korrekt,
  - verständlich gerade auch für Kunden

Entwurf (Design)



https://pixabay.com/photo-594132/ - StartupStockPhotos - CC0

- Entwicklung einer Software-**Architektur** **aus den Anforderungen**, die
  - die funktionalen und nichtfunktionalen **Anforderungen** erfüllt
  - nötige **Schnittstellen** zur Systemumgebung enthält
  - allgemeine und spezifische **Qualitätsanforderungen** erfüllt

- Eine **Software-Architektur** beschreibt
  - die **Struktur** des Softwaresystems als Menge von
  - klar voneinander abgegrenzten **Komponenten** und
  - **deren Zuständigkeiten** und **Beziehungen** untereinander.

- **Bis hier** (Analyse und Design) passieren **64% aller Fehler**

Implementierung

- **Realisierung** der einzelnen Systemkomponenten in Form von (Teil-)Programmen (Modulen, Klassen, ...)
  - Algorithmen und Datenstrukturen konzipieren
  - Strukturieren (schrittweise Verfeinerung)
  - Umsetzung in einer Programmiersprache
  - Einhaltung von Programmier-Richtlinien
  - Tests (Teil-/Gesamtsystem)
  - Dokumentation

- **Ergebnis:**
  - lauffähiges (Teil-)Programm
  - Programm-Quelltexte
  - Dokumentation
  - Testprotokolle

https://pixabay.com/photo-336376/ - Free-Photos - CC0 Creative Commons

Abnahme

- **Übergabe** des Produkts (incl. Dokumentation) an den Auftraggeber
- Durchführung von **Abnahmetests**, incl. Lasttests
- Dokumentation in einem **Abnahmeprotokoll**
- Bei Erfolg: Schriftliche **Abnahmeerklärung**

http://www.publicdomainpictures.net/pictures/60000/velka/handshake.jpg

Einführung / Inbetriebnahme

- **Installation** des Produkts auf Produktivsystem des Kunden
- evtl. **Datenübernahme** aus Altsystemen
- **Schulung** der künftigen Nutzer, Systemverwalter usw.
- **Inbetriebnahme** als
  - Parallelbetrieb zu einem Altsystem oder (mutiger)
  - direkter Inbetriebnahme mit Abschaltung des Altsystems

Betrieb (Wartung und Pflege)

- Nach Inbetriebnahme
  - können **Fehler** auftreten,
  - sich das **Systemumfeld** ändern (SW/HW, Organisation)
  - oder **neue Anforderungen** auftreten (z.B. neue Funktionen)

- **Wartung** & **Pflege** sorgen für
  - **Stabilisierung** des Systems und Korrektur von Fehlern
  - **Optimierung** / Leistungsverbesserung
  - Anpassungen / **Änderungen**
  - **Erweiterungen**

- Der **Aufwand** für Wartung & Pflege ist **ca 2x-4x so hoch** wie der Entwicklungsaufwand (für ein größeres, länger laufendes Produkt)

SWT-Themenbereiche gemäß SWEBOK

Kapitel des „Software Engineering Book of Knowledge“ (SWTBOK) der IEEE, https://www.sebokwiki.org/wiki/An_Overview_of_the_SWEBOK_Guide