



# 51. Über die Herstellung großer Softwaresysteme

---

---

Prof. Dr. rer. nat. habil. Uwe Aßmann  
Institut für Software- und  
Multimediatechnik  
Lehrstuhl Softwaretechnologie  
Fakultät für Informatik  
TU Dresden  
Version 15-0.1, 20.07.15

- 1) Software-Produkte
- 2) Große Systeme
- 3) Was heißt “Software Engineering”?
- 4) Geschichte
- 5) Ihre Zukunft

# Obligatorisches Lesen

- ▶ Zuser Kap. 1-3 *oder*
- ▶ Ghezzi Chapter 1 *oder*
- ▶ Pfleeger Chapter 1; Chap 8.1
  
- ▶ <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/> The first International Conference on Software Engineering (ICSE) 1968.
- ▶ S. Garfunkel Die schönsten Software-Fehler  
<http://www.wired.com/news/technology/bugs/0,2924,69355,00.html>
- ▶ Peter G. Neumann <http://www.risks.org> Das *Risk Digest* sammelt voller Akribie Softwarefehler
- ▶ IEEE ACM code of ethics for Software Engineers  
<http://www.computer.org/cms/Computer.org/Publications/code-of-ethics.pdf>

Für Hängebrücken  
gibt es keine  
Bastler-Bausätze.

Michael Jackson, IEEE Software 1/1998





# 51.1. Herstellung Großer Softwaresysteme

---

---





# 51.1. Software-Produkte

How to Earn Money with Products

Ensure that programs run only finitely many times

# Was heißt hier "groß"?

- ▶ Klassifikation nach W. Hesse:

Klasse	Anzahl Code-Zeilen	Personenjahre zur Entwicklung
sehr klein	bis 1.000	bis 0,2
klein	1.000 - 10.000	0,2 - 2
mittel	10.000 - 100.000	2 - 20
groß	100.000 - 1 Mio.	20 - 200
sehr groß	über 1 Mio.	über 200

# Riesige Systeme

- ▶ Telefonvermittlungssoftware EWSD (Version 8.1):
  - 12,5 Mio. Code-Zeilen
  - ca. 6000 Personenjahre
- ▶ ERP-Software SAP R/3 (Version 4.0)
  - ca. 50 Mio. Code-Zeilen
- ▶ Gesamtumfang der verwendeten Software (Anfang 2000):
  - Credit Suisse                                   25 Mio. Code-Zeilen
  - Chase Manhattan Bank:                   200 Mio. Code-Zeilen
  - Citicorp Bank:                                400 Mio. Code-Zeilen
  - AT&T:   500 Mio. Code-Zeilen
  - General Motors:                             2 Mrd. Code-Zeilen

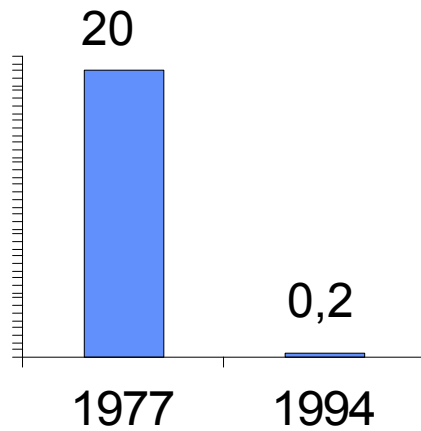
## Abkürzungen:

EWSD = Elektronisches Wählsystem Digital (Siemens-Produkt)

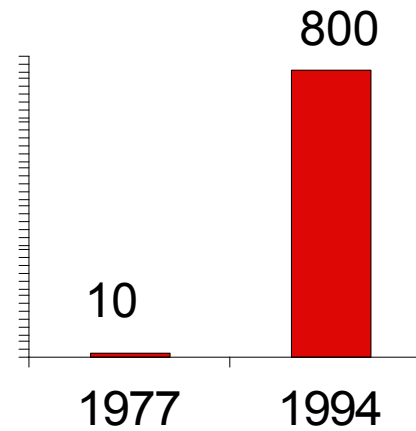
ERP = Enterprise Resource Planning

SAP: Deutscher Software-Konzern

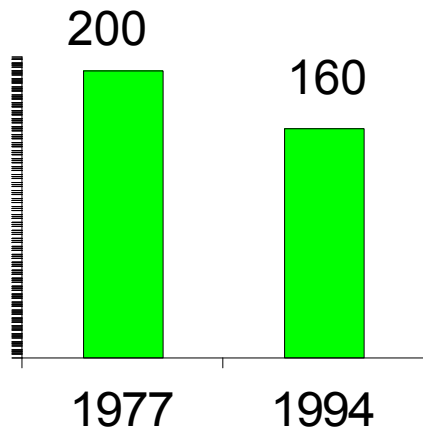
# Komplexitätswachstum und Fehlerrate



Anzahl Fehler auf 1000 LOC



Programmgröße (1000 LOC)



Resultierende absolute Fehleranzahl

Echte Qualitätsverbesserungen sind nur möglich, wenn die Steigerung der Programmkomplexität **überkompensiert** wird !



# Human Problems

- ▶ Programmers are not educated well
  - To develop
  - To communicate
- ▶ Software construction is a social process
  - It's hard to organize people
- ▶ Software stays, the people go
  - Software evolves, many versions
- ▶ Projects run out of time
  - How to control?

# Structural Problems

- ▶ The following figures are taken from the Goose Reengineering Tool, analysing a Java class system [Goose, FZI Karlsruhe, <http://www.fzi.de>]

# Permanente Softwarekrise ?

- ▶ "Software-Krise":
  - Fehler in Computersystemen sind fast immer Softwarefehler.
  - Software wird nicht termingerecht und/oder zu höheren Kosten als geschätzt fertiggestellt.
  - Software entspricht oft nicht den Anforderungen ihrer Benutzer.
- ▶ Begriff der Software-Krise existiert seit 1968 bis heute !
  - 70er Jahre:
    - ◆ Mangelhafte Beherrschung der Basistechnologie
  - 90er Jahre und Millennium:
    - ◆ Grosse (software-)technologische Fortschritte
    - ◆ Mangelhafte Beherrschung des Wachstumstempos und des Anpassungsdrucks

We undoubtedly produce software by backward techniques.

D. McIlroy, ICSE 1968

# Life Time of Software

- ▶ Average: 5 – 15y
  - max > 35 y (control software, certified systems, data bases)
  - Development time: 1 - 3y
- ▶ Microsoft (1994):
  - 50% turnaround from products of the last 12 months
  - 50 Products/y
  - new version every 18-24 months
  - 2/3 of a new version are new

# Programmer Productivity – Rules of Thumb

- ▶ System software: 1000-2000 loc/y
- ▶ Application software: 5-10000 loc/y
- ▶ Master's thesis: 10-20000 loc/thesis
- ▶ Individual differences up to factor 5
  - Has not changed in the last 30 years
- ▶ Differences by programming language and reuse mechanisms

## 51.2 "Software-Engineering"

Softwaretechnologie  $\leq$  Software-Engineering = Softwareingenieurswesen  
Softwaretechnik: Unterbegriff von Softwaretechnologie

*software engineering*: Die Entdeckung und Anwendung solider Ingenieur-Prinzipien mit dem Ziel, auf wirtschaftliche Weise Software zu bekommen, die zuverlässig ist und auf realen Rechnern läuft.

(F.L. Bauer, NATO-Konferenz Software-Engineering 1968)



# What is Software Engineering?

- ▶ It teaches the production of software with engineering techniques (the engineer's toolkit)
  - Analysis
  - Construction (development, invention)
  - Validation
  - Improvement
- ▶ in a structured software development process

Engineers measure, build, validate, and improve

# The (Software) Engineer's Toolkit

- ▶ Analyze (measure)
  - Measuring
  - Searching and finding
  - Controlling
  - Describe the world with *descriptive* models
- ▶ Construct (design, development, invent, build)
  - Modelling: prescribe the form of a system with a *prescriptive* model
  - Describing the infinite and the unknown with finite descriptions
  - Structuring (making the model more clear)
  - Refinement (making the model more precise and detailed)
  - Elaboration (adding more details)
  - DomainTransformation (changing representation of model)



# The (Software) Engineer's Toolkit

- ▶ Form hypotheses and validate them
  - Experimentation (empirical software engineering)
  - Checking (consistency, completeness, soundness)
  - Testing
  - Proving (formal software engineering)
  - Statistics
- ▶ Improve
  - Reverse engineer
  - Optimize with regard to a value model

# Der Softwareingenieur als Problemlöser

- ▶ Probleme identifizieren (beobachten)
  - ▶ Theorie und Verbesserungshypothesen entwickeln
  - ▶ Lösungen entwickeln (Techniken, Methoden u. Werkzeuge)
  - ▶ Empirisches Testen (danach Iteration)
- 
- ▶ Merke: Fehlt eine Aktivität, liegt kein ingenieurmässiges Arbeiten vor!

# Was ist eine Software-Entwicklungsmethode?

- ▶ Beschrieben in Lehrbüchern und Handbüchern
- ▶ Zweck: Hilfe bei der Erstellung von Software
  - Bessere Planbarkeit der Entwicklung
  - Bessere Struktur des Produkts



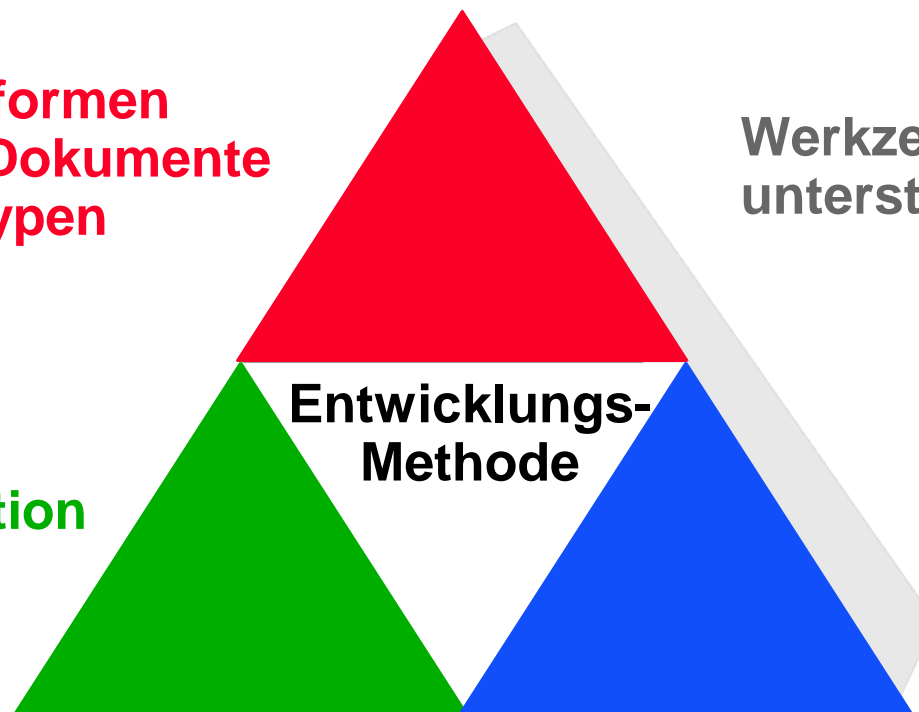
## Darstellungsformen

- Syntax für Dokumente
- Diagrammtypen

## Verfahren

- Analyse
- Transformation

Werkzeug-  
unterstützung



## Vorgehensmodell

- Schritte
- Reihenfolge
- Ergebnisse

# Definitions

- ▶ **Program:** Sources, object files, libraries (also called *bananaware*)
- ▶ **Testable system/program:** Reliable Program with extensive test suite
- ▶ **Software:** Testable program with user and developer documentation, requirement specification, design descriptions, implementation description
- ▶ **Product:** Mature software. Good, simple, and pedagogic documentation. Simple Installation. Support guaranteed
  - Scalable for many users and scenarios
  - Reliable
  - Note: Companies like products
- ▶ **Product line (product family):** A group of products, having a common framework and product-specific extensions.
  - Note: every product is sold independently

# Themengebiet "Software-Engineering"

Projektmanagement

Vorgehensmodellierung

Software-Entwicklungsmethoden

Requirements  
Engineering

Software-  
Architektur  
und -Entwurf

Software-  
Wartung

Re-  
Engineering  
(Sanierung)

Qualitätssicherung (incl. Testverfahren)

Werkzeugunterstützung (CASE)

# 51.3 Geschichte



# A Little History

- ▶ NATO Conference on Software Engineering in Garmisch-Patenkirchen. Oct 7-10, 1968
- ▶ *"The whole trouble comes from the fact that there is so much tinkering with software. It is not made in a clean fabrication process. What we need is Software Engineering."* Friedrich L. Bauer, 1968
- ▶ Hence the conference was called "on Software Engineering"
- ▶ [in Thayer&McGettrick IEEE Press]
- ▶ "Software Crisis"
- ▶ "Component Software"



# Photos of Famous People [Courtesy Brian Randell's Web Site]



E. Dijkstra



K. Samuelson  
(Stack)



D. McIlroy  
"Mass-produced  
Software Components"



W. van der Poel



B. Randell



D. Gries



T. Hoare



3<sup>rd</sup> from right: G. Goos



# Photos of Famous People [Courtesy Brian Randell's Web Site]



A. Perlis



J. Feldman



C. Strachey



N. Wirth



P. Brinch Hansen





# 51.4 Karriere



# Your Career

- ▶ First, you will be a designer and programmer in a team
  - You will need design skills most urgently for your own and small-size projects
  - In the software process, design flaws are most costly
- ▶ Afterwards, you will be project leader
  - Without good knowledge in design, you will not be a good developer nor project leader
- ▶ And then manager
  - But neither a good manager
  - Basic Microsoft strategy: every manager must be able to program
- ▶ .. but some gamble instead [Bill Gates, The Way Ahead]...

# Your Career (II)

- ▶ Some become entrepreneurs
- ▶ What is an entrepreneur?
  - [Prof. R. Würth: Lecture notes on entrepreneurship  
[http://www.iep.uni-karlsruhe.de/seite\\_260.php](http://www.iep.uni-karlsruhe.de/seite_260.php)]

Ein Unternehmer ist ein Problemlöser.  
Insbesondere sind ein Unternehmer und ein Kapitalist zweierlei. Der Kapitalist sieht den Gewinn im Mittelpunkt, aber der Unternehmer findet seine Befriedigung nur im Lösen von Problemen seiner Kunden und seiner Mitarbeiter. Damit kann er zwar auch Geld verdienen, im Wesentlichen lebt er aber nur einen grundlegenden Zug des Menschen aus: für Probleme befriedigende Lösungen zu finden.

# Your Career (III) - Consultant

- ▶ A consultant teaches and employs modern technology at a customer site
  - Must lead in knowledge and technology
  - Individual software
- ▶ Outsourcing
  - to Asia or MOE (Mittelosteuropa) is a trend
  - how to manage distributed projects?

# Was haben wir gelernt?

- ▶ Der Ingenieur misst, entwirft, validiert und verbessert
- ▶ Kein Ingenieur bleibt Programmierer, sondern muss sich entwickeln

# Referenz

- ▶ Die deutschen Folien der Softwaretechnologie-Vorlesung stammen zu grossen Teilen aus den Jahren 2000-2003 und wurden von Prof. Dr. Heinrich Hussmann, jetzt LMU München, erstellt. Used by permission.