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 21-0.2, 16.07.21

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

Obligatorische Literatur

- JDK Tutorial für J2SE oder J2EE, www.java.sun.com
- Dokumentation der Jgrapht library http://www.jgrapht.org/
  - Javadoc http://www.jgrapht.org/javadoc
  - http://sourceforge.net/apps/mediawiki/jgrapht/index.php?title=jgrapht:Docs
- Dokumentation der Library für verteilte Graphen GELLY (Teil von Apache Flink)
  - http://ci.apache.org/projects/flink/flink-docs-master/gelly_guide.html

Softwaretechnologie

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

How to Ensure that Programs Run Finitely Many Times

- Run as applet
- run as a webservice
- Cloud-based registration of run
- Checking a local memory file in the library data
- Timed licence:
  - Read by reflection the day of installation
  - After the deadline, show advertisement
- Buy uses (pay-per-use)

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

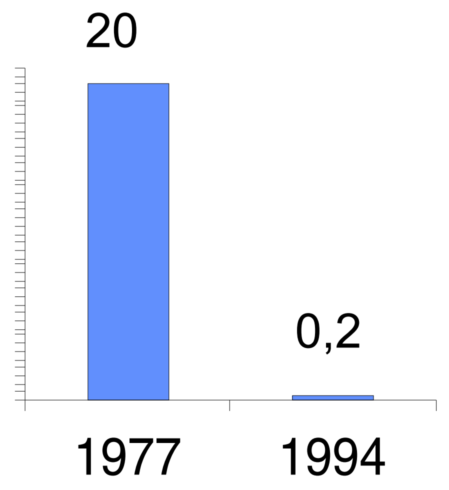
Riesige Systeme

- 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):

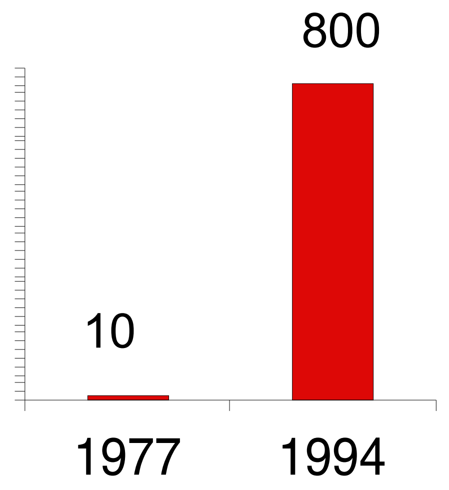
Credit Suisse	25 Mio. Code-Zeilen
Chase Manhattan Bank:	200 Mio. Code-Zeilen
Citicorp Bank:	400 Mio. Code-Zeilen
AT&T:	500 Mio. Code-Zeilen
General Motors:	2 Mrd. Code-Zeilen

Abkürzungen:
EWSD = Elektronisches Wählsystem Digital (Siemens-Produkt)
ERP = Enterprise Resource Planning
SAP: Deutscher Software-Konzern

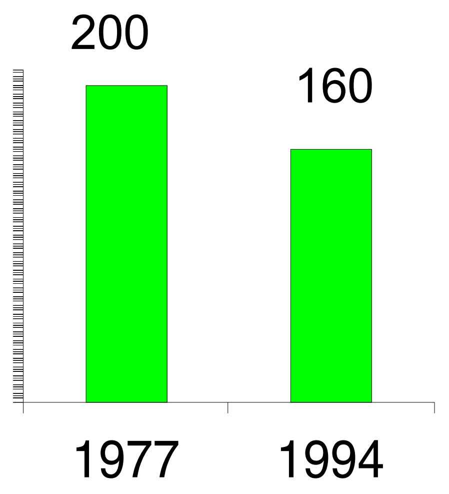
Komplexitätswachstum und Fehlerrate



Anzahl Fehler auf 1000 LOC



Programmgröße (1000 LOC)



Resultierende absolute Fehleranzahl

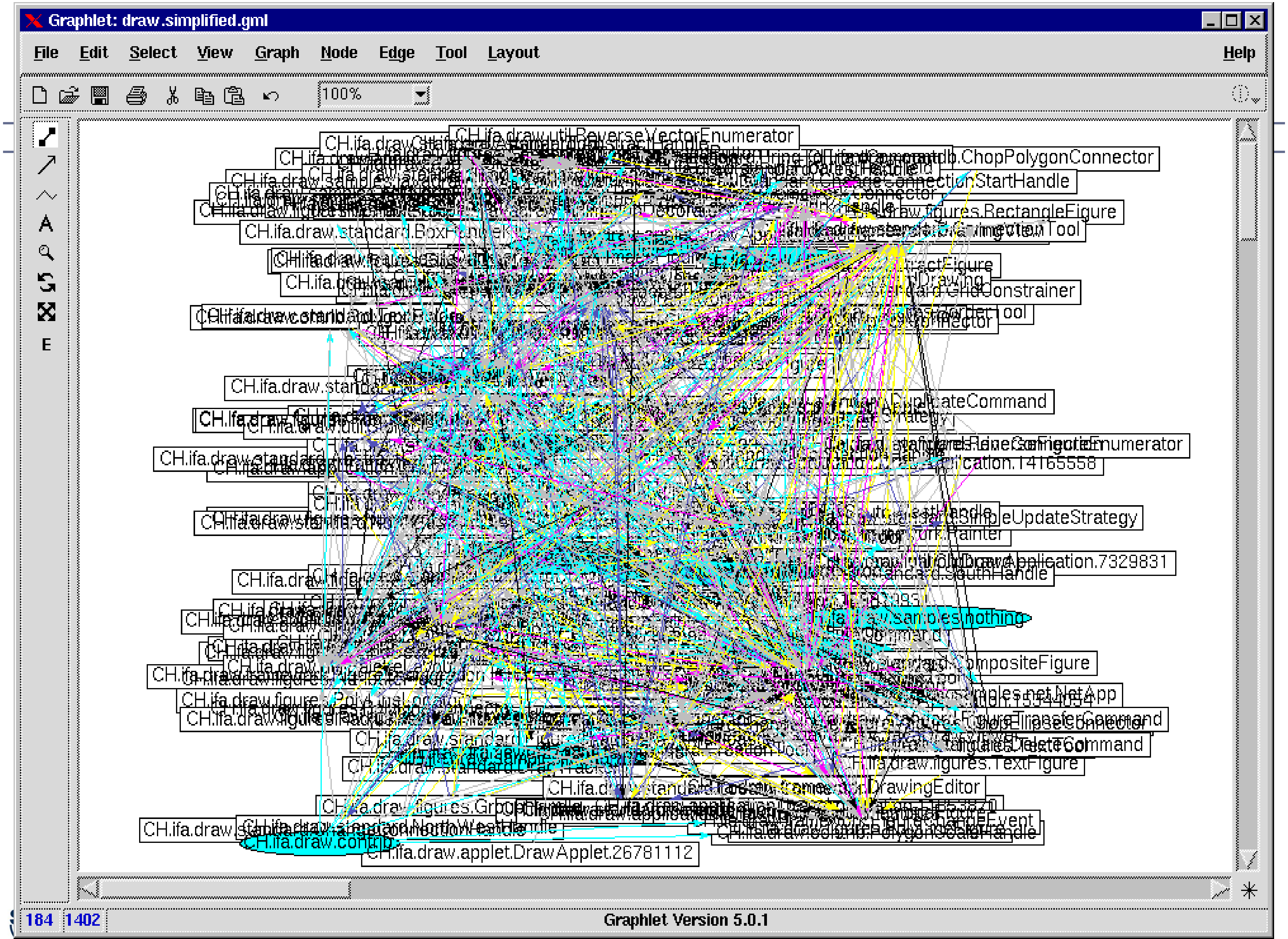
Echte Qualitätsverbesserungen sind nur möglich, wenn die Steigerung der Programmkomplexität **überkompensiert** wird !

Human Problems

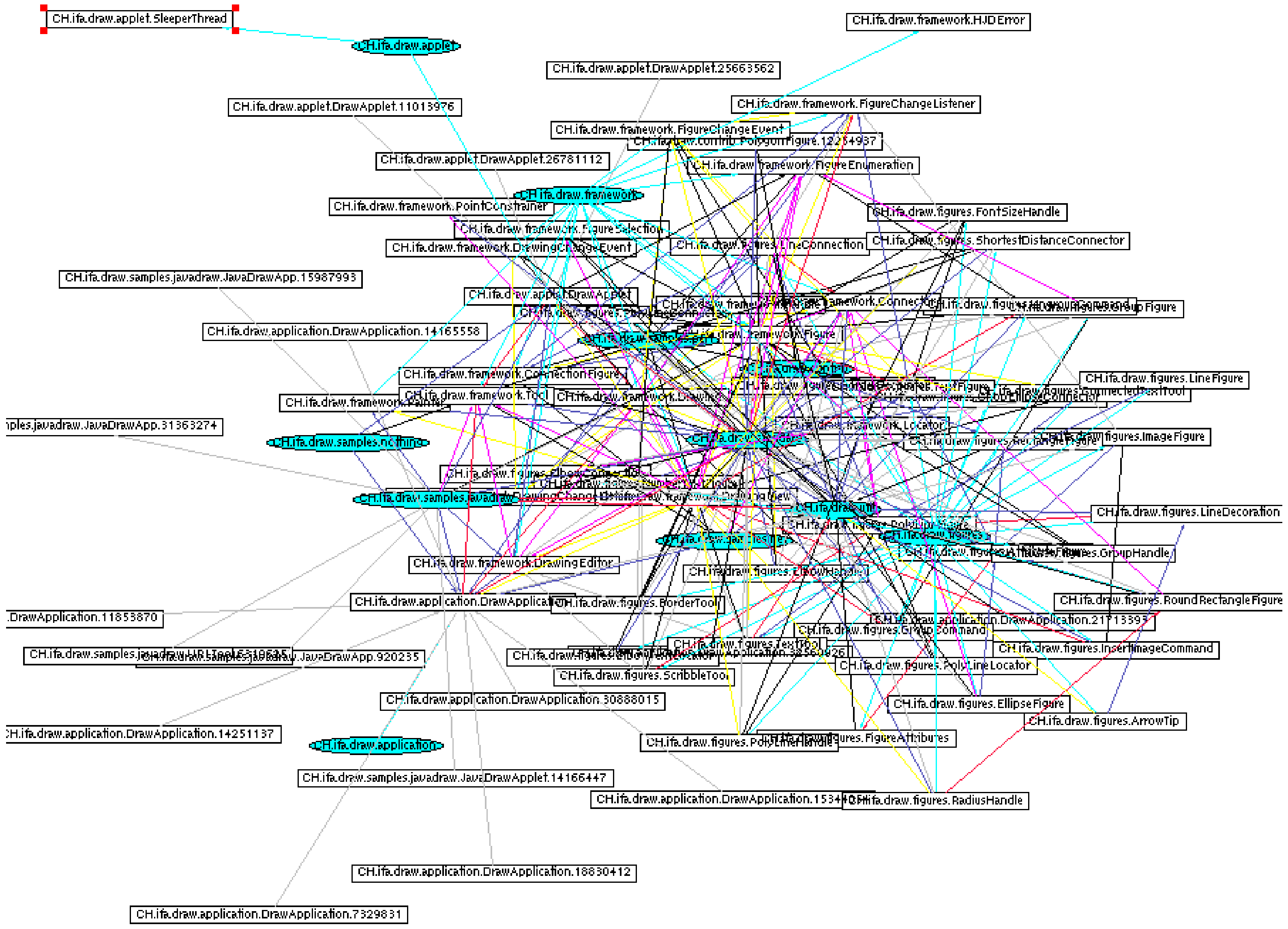
- Programmers are not educated well
  - To develop
  - To communicate
- Software construction is a social process
  - It’s hard to organize people
- Software stays, the people go
  - Software evolves, many versions
- Projects run out of time
  - How to control?

Structural Problems

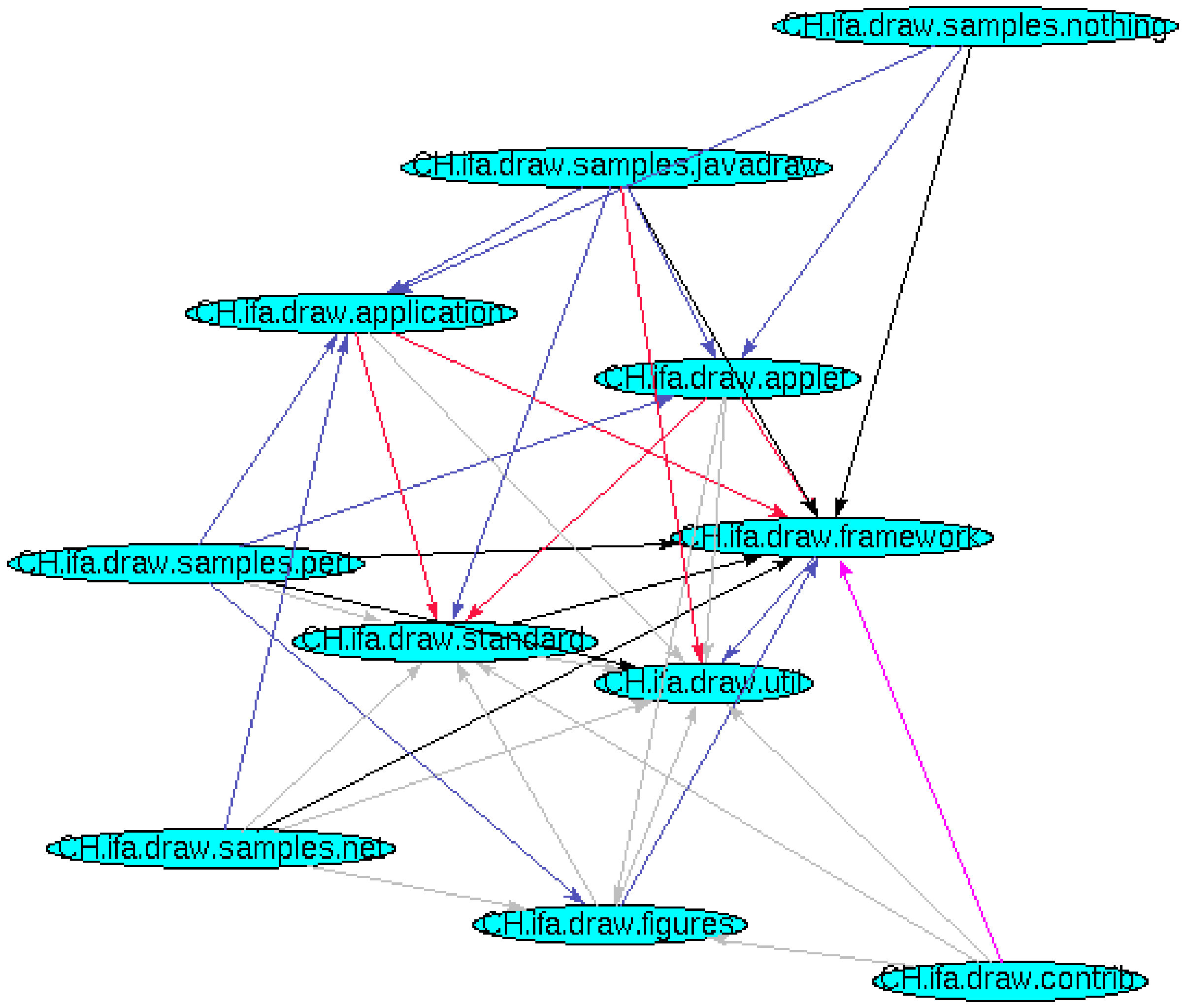
- The following figures are taken from the Goose Reengineering Tool, analysing a Java class system [Goose, FZI Karlsruhe, http://www.fzi.de]


Graphlet: draw.simplified.gml
File
Edit
Select
View
Graph
Node
Edge
Tool
Layout
Help
100%
CH.ifa.draw.util.ReverseVectorEnumerator
ChopPolygonConnector
ConnectionStartHandle
RectangleFigure
CH.ifa.draw.standard.BoxHandleKit
GridConstrainer
DuplicateCommand
SimpleUpdateStrategy
DrawApplication.7329831
SouthHandle
CH.ifa.draw.samples.nothing
CompositeFigure
CH.ifa.draw.figures.TextFigure
DrawingEditor
CH.ifa.draw.contrib
CH.ifa.draw.applet.DrawApplet.26781112
184
1402
Graphlet Version 5.0.1

Partially Collapsed


CH.ifa.draw.applet.SleeperThread
CH.ifa.draw.framework.HJDError
CH.ifa.draw.applet
CH.ifa.draw.applet.DrawApplet.25663562
CH.ifa.draw.applet.DrawApplet.11013976
CH.ifa.draw.framework.FigureChangeListener
CH.ifa.draw.framework.FigureChangeEvent
CH.ifa.draw.contrib.PolygonFigure.12214937
CH.ifa.draw.applet.DrawApplet.26781112
CH.ifa.draw.framework.FigureEnumeration
CH.ifa.draw.framework
CH.ifa.draw.framework.PointConstrainer
CH.ifa.draw.figures.FontSizeHandle
CH.ifa.draw.framework.FigureSelection
CH.ifa.draw.framework.DrawingChangeEvent
CH.ifa.draw.figures.LineConnection
CH.ifa.draw.figures.ShortestDistanceConnector
CH.ifa.draw.samples.javadraw.JavaDrawApp.15987993
CH.ifa.draw.applet.DrawApplet
CH.ifa.draw.framework.Connector
CH.ifa.draw.figures.GroupFigure
CH.ifa.draw.application.DrawApplication.14165558
CH.ifa.draw.framework.Figure
CH.ifa.draw.framework.ConnectionFigure
CH.ifa.draw.figures.LineFigure
CH.ifa.draw.framework.Tool
CH.ifa.draw.framework.Painter
CH.ifa.draw.figures.ConnectedTextTool
CH.ifa.draw.figures.ChopEllipseConnector
samples.javadraw.JavaDrawApp.31363274
CH.ifa.draw.framework.Locator
CH.ifa.draw.samples.nothing
CH.ifa.draw.figures.ImageFigure
CH.ifa.draw.figures.ElbowConnection
CH.ifa.draw.samples.javadraw
CH.ifa.draw.framework.DrawingView
CH.ifa.draw.figures.LineDecoration
CH.ifa.draw.figures.PolyLineFigure
CH.ifa.draw.figures.GroupHandle
CH.ifa.draw.framework.DrawingEditor
CH.ifa.draw.figures.ElbowHandle
DrawApplication.11853870
CH.ifa.draw.application.DrawApplication
CH.ifa.draw.figures.BorderTool
CH.ifa.draw.figures.RoundRectangleFigure
CH.ifa.draw.application.DrawApplication.21713393
CH.ifa.draw.figures.GroupCommand
CH.ifa.draw.samples.javadraw.URLTool
CH.ifa.draw.samples.javadraw.JavaDrawApp.920235
CH.ifa.draw.figures.TextTool
CH.ifa.draw.figures.InsertImageCommand
CH.ifa.draw.figures.PolyLineLocator
CH.ifa.draw.figures.ScribbleTool
CH.ifa.draw.application.DrawApplication.30888015
CH.ifa.draw.figures.EllipseFigure
CH.ifa.draw.figures.ArrowTip
CH.ifa.draw.application.DrawApplication.14251137
CH.ifa.draw.application
CH.ifa.draw.figures.PolyLineHandle
CH.ifa.draw.figures.FigureAttributes
CH.ifa.draw.samples.javadraw.JavaDrawApplet.14166447
CH.ifa.draw.application.DrawApplication.15344
CH.ifa.draw.figures.RadiusHandle
CH.ifa.draw.application.DrawApplication.18830412
CH.ifa.draw.application.DrawApplication.7329831

Totally Collapsed



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 Milennium:
    - Grosse (software-)technologische Fortschritte
    - Mangelhafte Beherrschung des Wachstumstempos und des Anpassungsdrucks

We undoubtedly produce software by backward techniques.
D. McIlroy, ICSE 1968

The Industry

- Top Players: IBM, Microsoft, Google, HP, Hitachi, Computer Associates, Oracle, Fujitsu, SAP, Bull HN, Novell
- About 10% of the national turnaround of the US
  - 5% of employees, about the size of car industry
- PC-game equates Hollywood film ($50 mill. in first week)
- 2/3 standard software : 1/3 individual software (with growing rate)
- Im 'Quartal erwirtschaftet Microsoft
  - Umsatz im Bereich von 10-15 Milliarden US-Dollar.
  - operativer Gewinn: 3-6 Milliarden US-Dollar bewegen.
- Atlassian-Ecosystem grows 50% every year! www. atlassian.com

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

Costs

- Contrary to Grosch's Law,
  - Hardware speed doubles every 2 years
- Software productivity increases only about < 5% /y.
- Costs
  - acad. Prototype / acad. Product / Product = 1 : 3 : 3
  - Commercialization is rather difficult
- Relation of development and maintenance 40:60 up to 20:80
  - Development and maintenance are done by different departments

Costs: Extreme Requirements

- Telecommunication: Failure < 1 h./40 y., working rate 99.999%
  - One second failure may cost $5Mio
- Certification
  - Insurance: certified software must be executable after 40 years
  - German pension rules of the 50s must be processed today
    - Nobody knows the details anymore!
    - Solution: write an interpreter for the old assembler
    - This has happened twice..

Cost Example: The Year 2000 Problem

- COBOL programmers saved space and stored only the last two digits of the year
  - In the 70s, programs should only live 20 years
- In 2000, catastrophes were prophesied
  - Power plants?
  - Pension insurances (birth dates)
- From 1996 on, the industry panicked
  - Spent enormous amounts to update software
- New systems got installed
  - SAP R/3 with date data type
- Rewriting didn't work
  - Programmers didn't trust the rewrites
  - Solution: sliding window technique

Cost Example: The Euro Introduction in 2001

- End of 2001, many countries introduced the Euro
- Too bad: on paper, the Euro was introduced 2 years before
  - Some companies had to maintain double booking for 2 years
  - At least for some months in 2002
  - Double booking was very costly: accounts had to be printed in two currencies
- How to test the transition?
  - In May 2001, the Dresdner bank ran a test
  - Which failed,.. And produced many wrong money transfers!
- Many people worked day and night…
  - One of my friends was responsible for the ATM machines…

Cost Example: Telecommunication

- Telecommunication software product line
  - 20-30 000 Module of 1000 loc (lines of code)
- Single product has 2-8000 modules
- Necessary: 5000 persons/7years.
- Costs ca. 7 billion €.
- Size of world market 50 billion €
- How many suppliers can exist?

Programmer Productivity – Rules of Thumb

- System software: 1000-2000 loc/y
- System like Software: 5000 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 <= 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

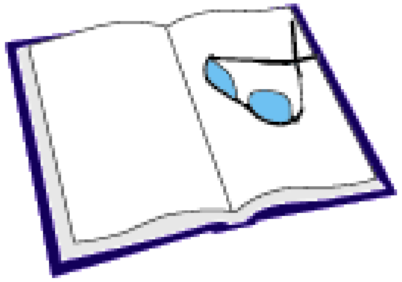
Der Softwareingenieur als Problemlöser

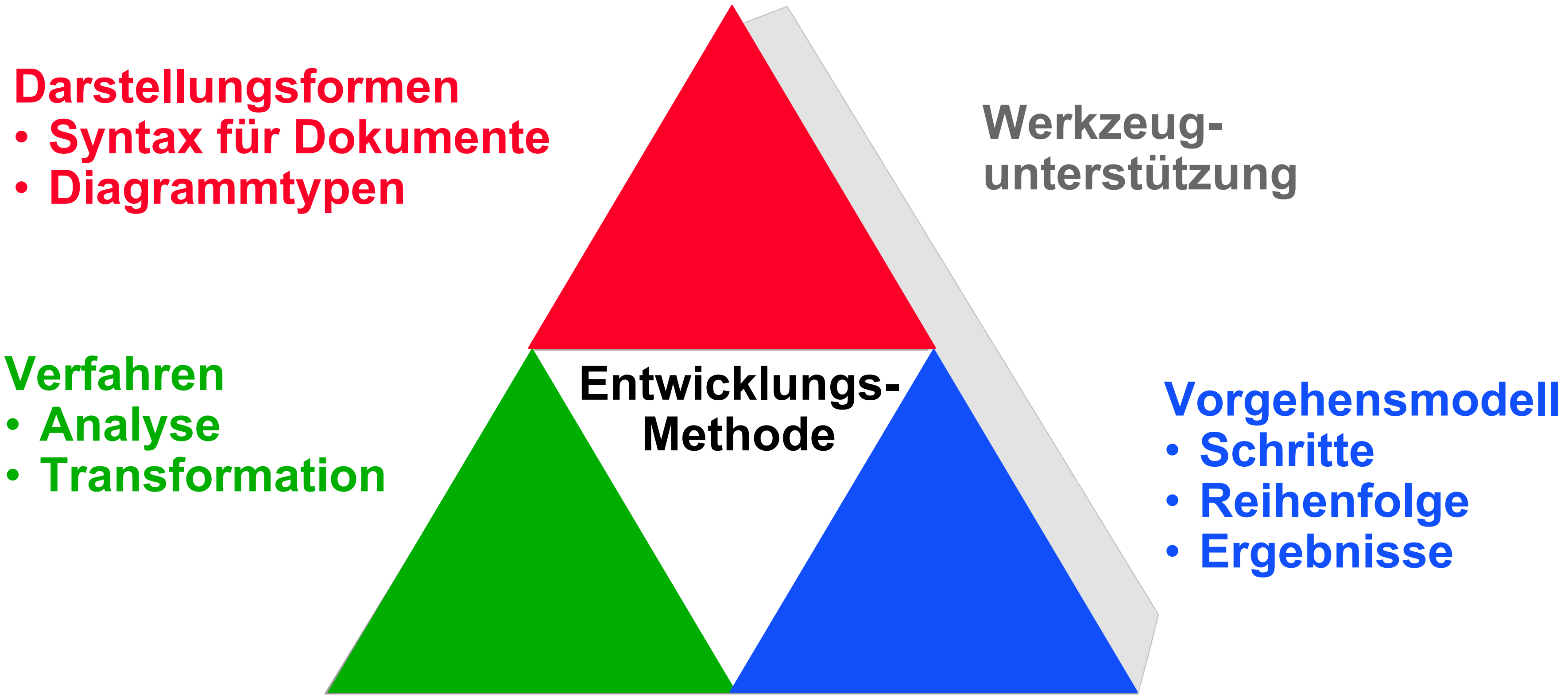
- Probleme identifizieren (beobachten)
- Theorie und Verbesserungshypothesen entwickeln
- Lösungen entwickeln (Techniken, Methoden u. Werkzeuge)
- Empirisches Testen (danach Iteration)

- Merke: Fehlt eine Aktivität, liegt kein ingenieursmässiges 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





Another Important Aspect of Software Engineering

- It teaches the production of software with *languages* for
  - Analysis
  - Design
  - Validating and experimenting
  - Improvement
- On different abstraction levels
  - Imprecise (sketching information)
  - Precise (such that code can be generated from the specifications)
- In different syntaxes
  - Informal, formal, diagrammatic, high-level
- Both on the technical and on the process (management) level

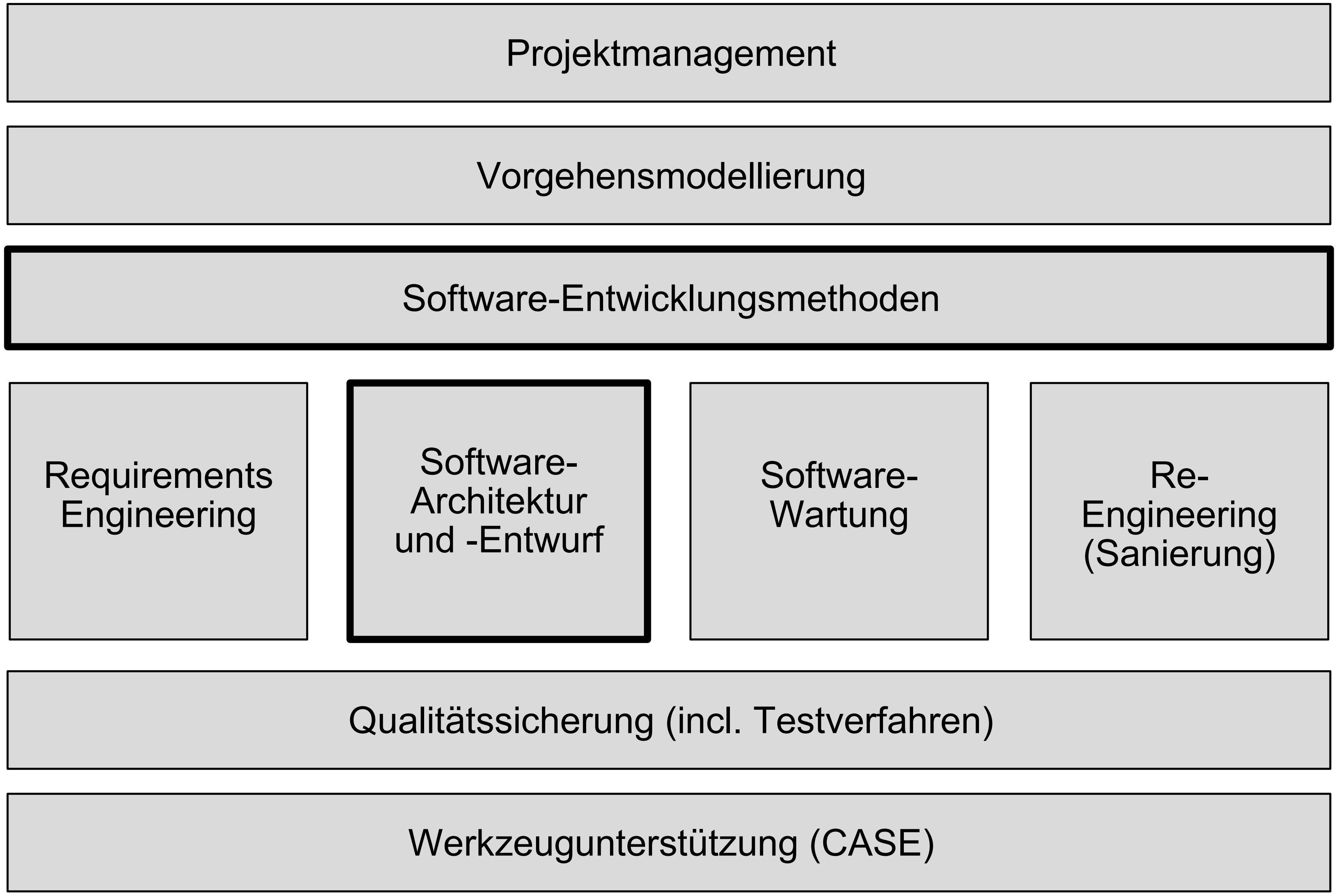
Definitions

- ***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

Lehman's Classes of Programs

- Specification programs (S-programs)
  - A formal problem specification exists, describing problem and solution
  - The specification allows for checking the solution on validity (formal checks or formal proofs)
- Problem solving programs (P-programs)
  - Can be formalized and checked
  - Have requirements for usability and appropriate
- Embedded programs (E-programs)
  - Embedded in a social context
  - The specification is a social process; the functionality depends on the involved people
  - No correctness proofs possible

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

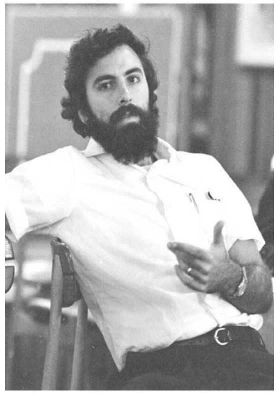
D. Gries

T. Hoare

3rd 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]

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
- Keiner 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.