# Software Engineering

## Motivation für Software Engineering

Prof. Dr. Bodo Kraft

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

# Agenda der gesamten Vorlesung
## Motivation für Software-Engineering



1. Einstieg ins Software-Engineering
2. Vorgehensmodelle
3. Anforderungsanalyse
4. Entwurf
5. Implementierung
6. Qualitätssicherung
7. Umfeld der Software-Entwicklung

# Agenda für heute / diese Woche
Motivation für Software-Engineering



## Präsenzvorlesung

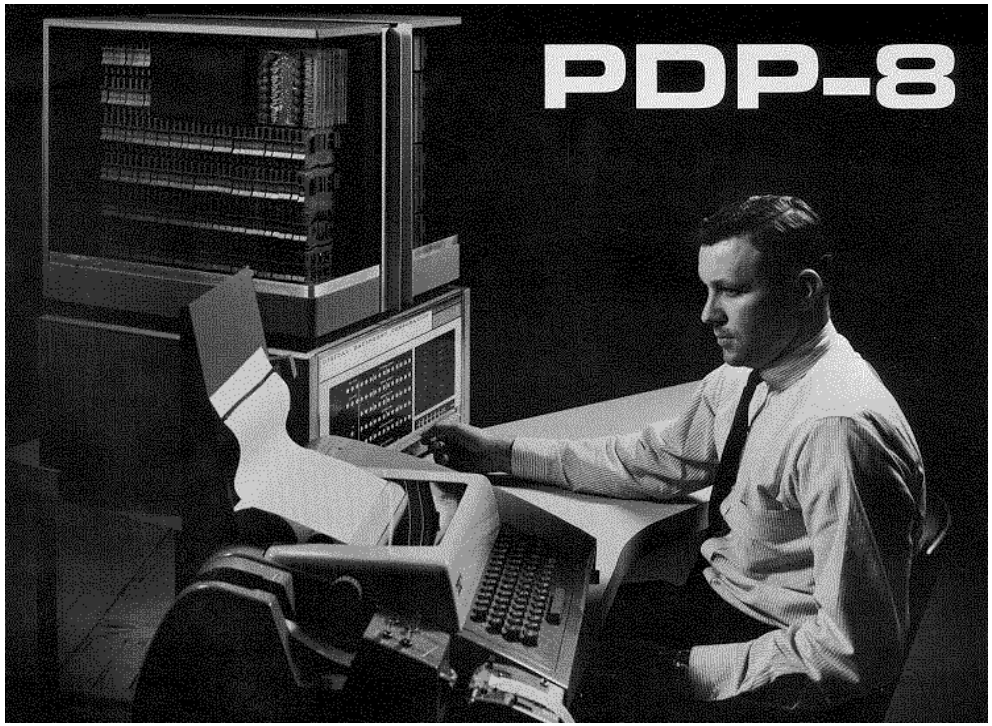
- Historie des SW-Engineering
- Warum ist SW-Engineering wichtig?
- Warum scheitern SW-Projekte?
- SW-Krise und die Antwort: SW-Engineering
- Definitionen für SW-Engineering

## Selbststudium

- Einstieg in GIT (Teil 1 ist unbedingt nötig, Teil 2 freiwillig)
- Details zur Organisation der Softwaretechnik Projekte

# Softwaretechnik im Jahr 1972 (1)

## Historie des SW-Engineering


000 - AND - undiere Operanden mit AC.
001 - TAD - addiere Operanden zu <L,AC> (ein 13 Bit-Wert).
010 - ISZ - inkrementiere Operanden und skippe wenn Ergebnis 0.
011 - DCA - deponiere AC im Speicher und lösche AC.
100 - JMS - jump to subroutine, Springe zum Unterprogramm.
101 - JMP - jump, Springe.
110 - IOT - input/output transfer, Ein-Ausgabe Operation.
111 - OPR - Microkodierte Operationen.

# Softwaretechnik im Jahr 1972 (2)

Historie des SW-Engineering



Welche Antworten gab es damals auf die Frage:

„Wie soll Software entwickelt werden?“

**PROGRAMMING RULES**

The most successful method of programming is to begin a program as simply as possible, test it, and then add to the program until it performs the required job. Before beginning the programming, the programmer should become familiar with the programs that he will be using. Refer to Chapter 4 for a description of the standard programs and refer to Appendix A for a complete list of the PDP-8/E compatible programs. For best results, the programmer should avoid the use of the following device codes:

1. Devide code 0 (reserved for processor)
2. Device code 3

# Entwicklung von Sprachen und Paradigmen (2)
Historie des SW-Engineering



## Ende 60er

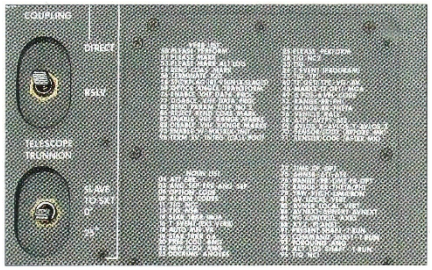
- Bedarf für Softwaretechnik neben der reinen Programmierung erstmals voll erkannt
- Vorher sind zahlreiche große Programmentwicklungen (möglich durch verbesserte Hardwareeigenschaften) gescheitert
- Arbeiten von Dijkstra 1968 (u.a. gegen Verwendung von GOTO) und Royce 1970 (Software-Lebenszyklus),
- Top-Down-Entwurf, graphische Veranschaulichungen (Nassi-Shneiderman Diagramme)

## Mitte 70er

- Top-Down-Entwurf für große Programme nicht ausreichend, zusätzlich Modularisierung erforderlich
- Entwicklung der Begriffe *Abstrakter Datentyp, Datenkapselung* und *Information Hiding*

# - Einschub -
## Historie des SW-Engineering






**Apollo Guidance Computer (AGC)**
Bordrechner der Apollo Mission

**Margaret Hamilton** next to a stack of Apollo Guidance Computer source code.
Photo Credit: Courtesy MIT Museum

# Entwicklung von Sprachen und Paradigmen (3)
Historie des SW-Engineering



## Ende 70er

- Bedarf für präzise Definition der Anforderungen an ein Softwaresystem, Entstehen von Vorgehensmodellen, z. B. *Structured Analysis Design Technique* (SADT)

## 80er Jahre

- Vom Compiler zur Entwicklungsumgebung (Editor, Compiler, Linker, symbolischer Debugger, Source Code Control Systems)
- Weiterentwicklung der Modularisierung und der Datenkapselung zur objektorientierten Programmierung

## 90er Jahre

- Objektorientierte Programmierung nimmt zu
- Neue Programmiersprache Java (ab Mitte 80er C++)
- Anwendungs-Rahmenwerke (*Application Frameworks*) zur Vereinfachung von Design und – vor allem – Programmierung

# Entwicklung von Sprachen und Paradigmen (4)
Historie des SW-Engineering



## 90er Jahre

- Geeignete Analyse- und Entwurfsmethoden entstehen (*Coad/Yourdon*, *Rumbaugh, Booch, Jacobson* und andere)

## 1995

- Vereinigung mehrerer Ansätze zunächst als *Unified Method* (UM) von Booch und Rumbaugh, dann kommt Jacobson hinzu (Use Cases).
- 3 Amigos definieren die *Unified Modeling Language* (UML) als Quasi-Standard.

## 1997

- UML in der Version 1.1 bei der OMG (Object Management Group) zur Standardisierung eingereicht und angenommen
- UML ist jedoch keine Entwicklungsmethode (Phasenmodell), *nur* eine Beschreibungssprache

# Entwicklung von Sprachen und Paradigmen (5)
Historie des SW-Engineering



## 1999

- Entwicklungsmethode: *Unified Process* (UP) und *Rational Unified Process* (RUP) (erste Version)

## 2010 bis heute

- Starker Fokus auf Agile Entwicklungsmethoden, eXtreme Programming, Scrum, Kanban
- Leistungsfähige Skript-Sprachen und Frameworks minimieren „Boilerplate-Code“
- Verbesserte Werkzeuge zur Kollaboration, Qualitätssicherung, Entwicklung etc.
- Container-Technologie, Continuous Integration, Continuous Delivery

# Warum ist Software-Engineering so wichtig?
Motivation für Software-Engineering



## Gute Softwareentwicklung ist wichtig

- Die Wirtschaft aller industrialisierten Länder hängt von Software ab.
- Immer mehr Systeme werden durch Software gesteuert.
- Die Aufwendungen für Software repräsentieren einen enormen Faktor im Bruttosozialprodukt aller Länder.

Jahr **2017** betrug das **Bruttoinlandsprodukt** Deutschlands rund 3,28 Billionen Euro.

| ITK-Markt Deutschland* | Marktvolumen (in Mrd. Euro) | | | | Wachstumsraten | | |
|---|---|---|---|---|---|---|---|
| | 2015 | 2016 | 2017 | 2018 | 16/15 | 17/16 | 18/17 |
| **Summe ITK + CE** | **157,6** | **157,8** | **161,3** | **164,0** | **0,1%** | **2,2%** | **1,7%** |
| Consumer Electronics | 9,6 | 9,2 | 9,4 | 9,3 | -4,4% | 2,6% | -1,9% |
| Summe ITK | 148,0 | 148,6 | 151,8 | 154,7 | 0,4% | 2,2% | 1,9% |
| **Informationstechnik** | **80,9** | **83,0** | **86,2** | **88,8** | **2,6%** | **3,9%** | **3,1%** |
| IT-Hardware | 23,4 | 23,2 | 24,2 | 24,4 | -0,7% | 4,2% | 0,9% |
| Software | 20,4 | 21,6 | 23,0 | 24,4 | 6,2% | 6,3% | 6,3% |
| IT-Services | 37,2 | 38,1 | 39,0 | 40,0 | 2,7% | 2,3% | 2,6% |
| **Telekommunikation** | **67,1** | **65,6** | **65,7** | **65,9** | **-2,2%** | **0,1%** | **0,4%** |
| TK-Endgeräte | 11,3 | 10,1 | 10,5 | 10,7 | -11,0% | 4,7% | 1,3% |
| TK-Infrastruktur | 6,5 | 6,6 | 6,6 | 6,7 | 1,7% | 0,5% | 1,4% |
| Telekommunikationsdienste | 49,3 | 49,0 | 48,5 | 48,5 | -0,7% | -1,0% | 0,1% |

* Für detaillierte Zahlen zum deutschen ITK-Markt sowie zu anderen europ./internat. Märkten siehe: www.eito.com Abweichend von den EITO-Definitionen werden hier im Segment IT-Hardware auch Halbleiter berücksichtigt.

Quelle: Bitkom Research GmbH

Wirtschaftliche Bedeutung von Software

# Anteil der Kosten für Software
Motivation für Software-Engineering

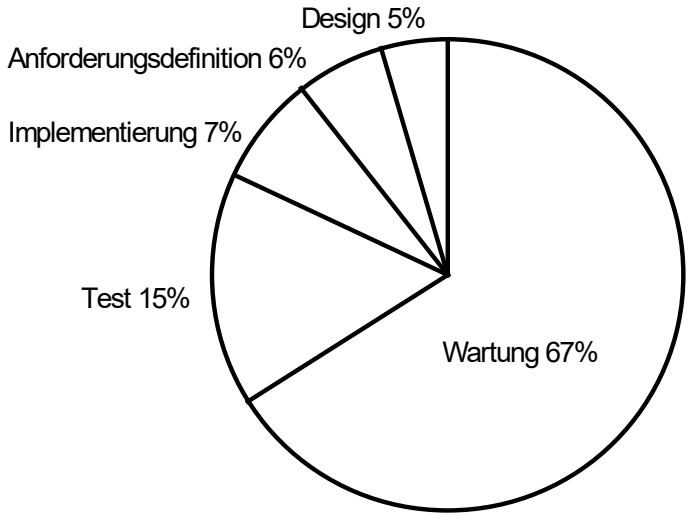
## Software-Kosten

- **Softwarekosten dominieren oft die Systemkosten**

  Die Kosten der Software eines PCs sind oft höher als die Hardwarekosten.

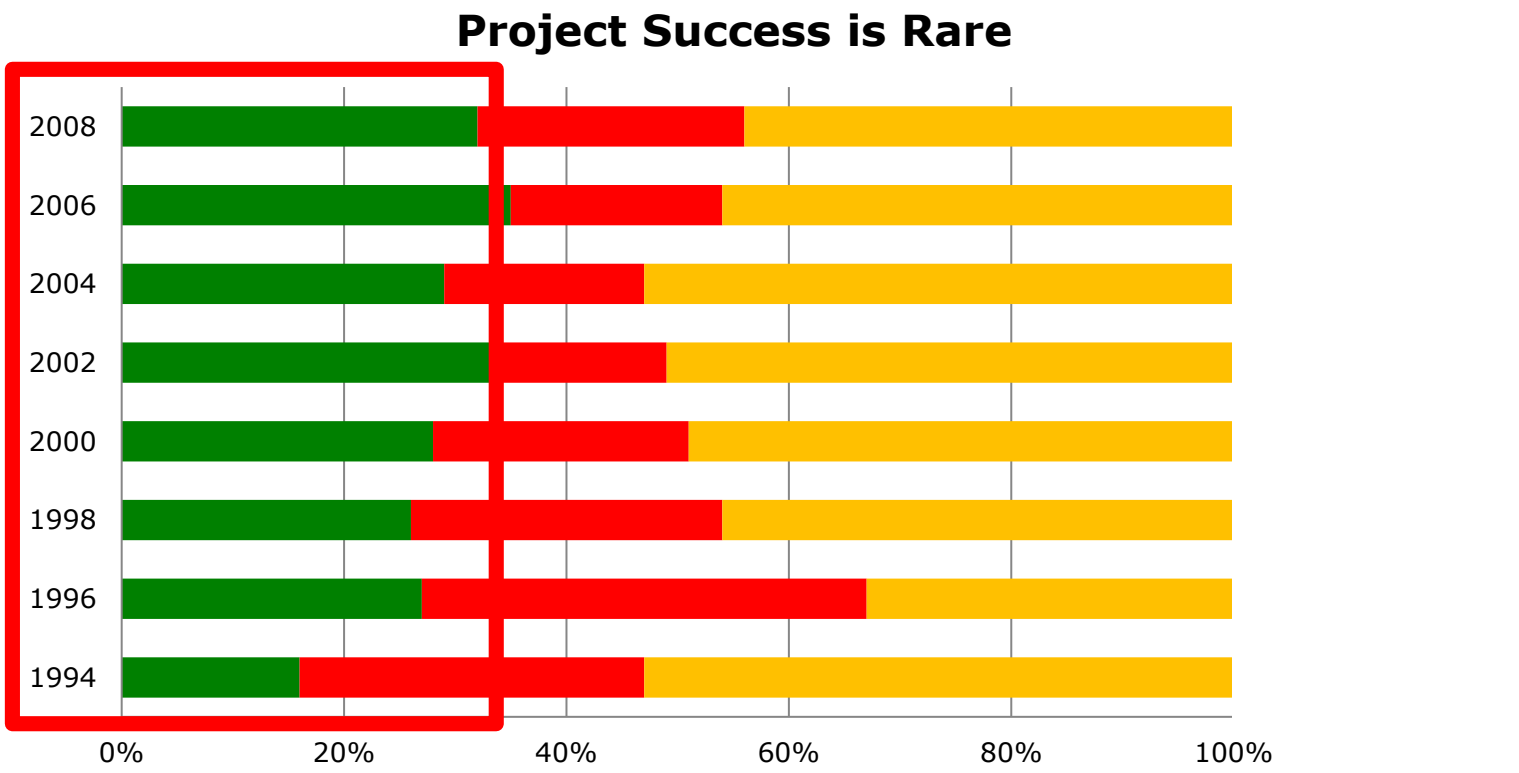
- **Die Wartung der Software ist teurer als die Entwicklung**

  Für Systeme mit einer langen Lebensdauer sind die Wartungskosten um ein Vielfaches höher als die Entwicklungskosten.



# Schlechtes Image von Softwareprojekten

## Warum scheitern SW-Projekte?




| | 1994 | 1996 | 1998 | 2000 | 2002 | 2004 | 2006 | 2008 |
|---|---|---|---|---|---|---|---|---|
| Successful | 16% | 27% | 26% | 28% | 34% | 29% | 35% | 32% |
| Failed | 31% | 40% | 28% | 23% | 15% | 18% | 19% | 24% |
| Challenged | 53% | 33% | 46% | 49% | 51% | 53% | 46% | 44% |

*Source: The Standish Group (2009)*

Diskutieren Sie gerne im Chat mit

# Typische Gründe für das Scheitern von SW-Projekten

Warum scheitern SW-Projekte?

- Die Software wurde wesentlich zu spät geliefert
- Die Software erfüllt nicht die Wünsche des Kunden
- Die Software läuft nicht auf den vereinbarten Rechnersystemen, sie ist zu langsam oder kommt mit dem Speicher nicht aus
- Die Software kann nicht erweitert werden oder mit anderer Software zusammenarbeiten
- …

# Immer wieder scheitern IT-Projekte
## Beispiele für spektakuläre Softwarefehler



### Fehler bei Arbeitslosengeld-II-Auszahlung (2005)

- Bei Kontonummern von ALG-II-Empfängern mit weniger als 10 Stellen: rechtsbündiges statt linksbündiges Auffüllen mit 0
  --> Empfänger konnten nicht zugeordnet werden

  Kontonummer: 123 456 78 → 123 456 78 **00** falsch
  Kontonummer: 123 456 78 → **00**1 234 56 78 richtig

- Hunderttausende ALG-II-Empfänger erhalten kein Geld

### Mehrere Toll-Collect-Probleme (2005)

- Fehler in Funktionen der On-Board-Unit
- Ca. 17.000 LKW benutzen mautpflichtige Strecken ohne korrekt abgerechnet zu werden

### Verlust des Mars Global Surveyors (2006)

- 2 falsche Speicheradressen bei Parameterupdate eingegeben
- Satellit richtete Sollarzellen auf Sonne aus
- Batterie überhitzte und zerstörte sich dadurch

# Absturz Ariane 5 (1)

## Beispiele für spektakuläre Softwarefehler

Europäische Trägerrakete zum Transport von kommerziellen Nutzlasten (z. B. Kommunikationssatelliten, usw.) in die Erdumlaufbahn

Nachfolger der erfolgreichen Ariane 4

Ariane 5 hat eine höhere Nutzlast als Ariane 4

Jungfernflug: 4. Juni 1996

Totalausfall der Ariane 5 auf ihrem Jungfernflug

Ariane 5 video

# Absturz Ariane 5 (2)
Beispiele für spektakuläre Softwarefehler



## Startfehler

- Ungefähr 37 Sekunden nach dem erfolgreichen Abheben, verlor Ariane 5 die Kontrolle
- Falsche Steuersignale wurden an die Triebwerke gesendet
- Untragbare Spannungen an der Rakete entstanden
- Ariane 5 brach auseinander und zerstörte sich selbst
- Der Systemfehler war eine direkte Folge eines **Softwarefehlers**

# Absturz Ariane 5 (3)
Beispiele für spektakuläre Softwarefehler

## Ariane 5 – Softwarefehler

- Der Softwarefehler trat auf bei dem Versuch, eine **64-bit floating point Zahl** in eine **16-bit signed Integer-Zahl** zu konvertieren, was zu einem Überlauf führte

- Es gab **keine Ausnahmebehandlung** für die Konvertierung, so wurde das System Exception Management aktiviert. Dieses schaltete die Software ab

- Die **Backup-Software war eine Kopie** und verhielt sich identisch

# So geht es nicht weiter…
## Warum scheitern SW-Projekte?

# Prägung der Disziplin Software-Engineering

## SW-Krise und die Antwort: SW-Engineering




- Mitte der 60-er Jahre: Erstmals überstiegen die Kosten für die Software die Kosten für die Hardware. In der Folge kam es zu den ersten großen gescheiterten Software-Projekten

- NATO-Tagung **1968** in Garmisch-Partenkirchen: Als Reaktion wurde der Begriff *Software Engineering* geprägt.

- [The major cause of the software crisis is] that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem." (Dijkstra, 1972)

# Fokus auf Standardisierung, Planung und Qualität
SW-Krise und die Antwort: SW-Engineering

**Werkzeuge**: Effektive Entwicklung:

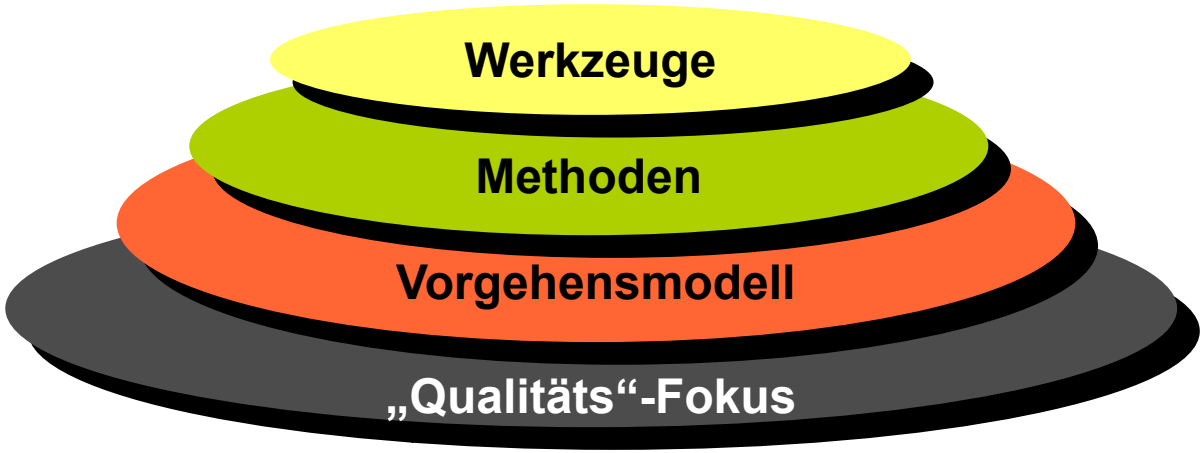
SEU, Build, Test, Quality, VCS, ...

**Methoden**: Beschreiben bewährte Verfahren:

OOA, OOD, TDD, Coding Conventions, CI/CD...

**Vorgehensmodelle**: Standardisierte Prozessbeschreibungen:
Scrum, Kanban, V-Modell, RUP

**Dokumentation**: Einheitliche Notationen für Entwicklungsergebnisse:
UML, Templates, Tracebility im Prozess



# Ursachen für schlechte Software-Qualität
Warum scheitern SW-Projekte?



## Organisation und Management

- unklare Zielvorstellung
- unklare Verantwortlichkeit
- nicht adäquate Projektplanung und Projektsteuerung
- unzulängliche Projektkostenschätzung
- abhängige Qualitätssicherung

## Technologie

- keine Verwendung von Vorgehensmodellen
- keine Strategie für Entwicklungsprozessverbesserungen
- geringe Nutzung von CASE-Tools
- kein systematisches Testen

## Methodik

- fehlende / ungenügende Anforderungserfassung
- keine Nutzung von Entwicklungsmethoden
- unvollständige Dokumentation
- keine Nutzung von Standards
- keine Qualitätssicherung

# Ursachen für schlechte Software-Qualität (2)
Warum scheitern SW-Projekte?



## Vergleich zu anderen Produkten

- Software ist immateriell
- Software wird nicht durch physikalische Gesetze begrenzt
- Software ist im Allgemeinen leichter und schneller zu ändern als ein technisches Produkt

- Für Software gibt es keine Ersatzteile
- Software unterliegt keinem Verschleiß
- Software altert

- Herstellung vieler Exemplare trivial

# Attribute guter Software
## SW-Krise und die Antwort: SW-Engineering



### Performance

- Gute Software muss die geforderte Funktionalität so schnell wie möglich erbringen

### Effizienz

- Gute Software muss ressourcenschonend sein

### Wartbarkeit

- Gute Software muss leicht gewartet werden können, um den Änderungsbedarf zu erfüllen

### Zuverlässigkeit

- Gute Software muss stabil funktionieren

### Benutzbarkeit

- Gute Software muss für den Benutzer benutzbar sein, für den sie entworfen wurde

# Was ist Softwaretechnik?
## Definitionen für SW-Engineering

**Definition 1: Was gehört zur Softwaretechnik?**

Softwaretechnik ist die Anwendung von Prinzipien, Fähigkeiten und Kunstfertigkeiten beim Entwurf und der Erstellung von Programmen und Systemen von Programmen

**Künstler, Einzelkämpfer**

[Dennis, 1975]

**Definition 2: Äußere Umstände**

Softwaretechnik ist Programmierung unter mindestens einer der folgenden Bedingungen:

- mehr als eine Person befasst sich mit der Erstellung und/oder dem Gebrauch des Programms
- mehr als eine Version des Programms wird erstellt werden

**Teamarbeit, Pflege von Software**

[Parnas, 1985]

**Definition 3: Ziele der Softwaretechnik**

Die Ziele der Softwaretechnik sind die ökonomische Erstellung von Software, die zuverlässig ist und effizient auf realen Maschinen arbeitet.

**Wirtschaftlichkeit, Ingenieurdisziplin**

[Bauer, 1990]

# Was ist modernes Software-Engineering?
## Definitionen für SW-Engineering



Software-Engineering ist der Einsatz

qualifizierter Methoden, Werkzeuge und Vorgehensmodelle zum Erstellen und Betreiben von Software mit dem Ziel,

- einerseits die Softwarekosten bei der Entwicklung, Wartung und Erweiterung von Programmsystemen zu senken und
- andererseits eine höheren Systemqualität zu erreichen.

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik

Prof. Dr. Bodo Kraft

Heinrich-Mußmann-Straße 1
52428 Jülich
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Kraft@fh-aachen.de
http://www.fh-aachen.de/menschen/kraft/