



Vorlesung Softwaretechnik I (SS 2024)

1. Einführung

Prof. Dr. Jens Grabowski

Tel. 39 172022

grabowski@informatik.uni-goettingen.de

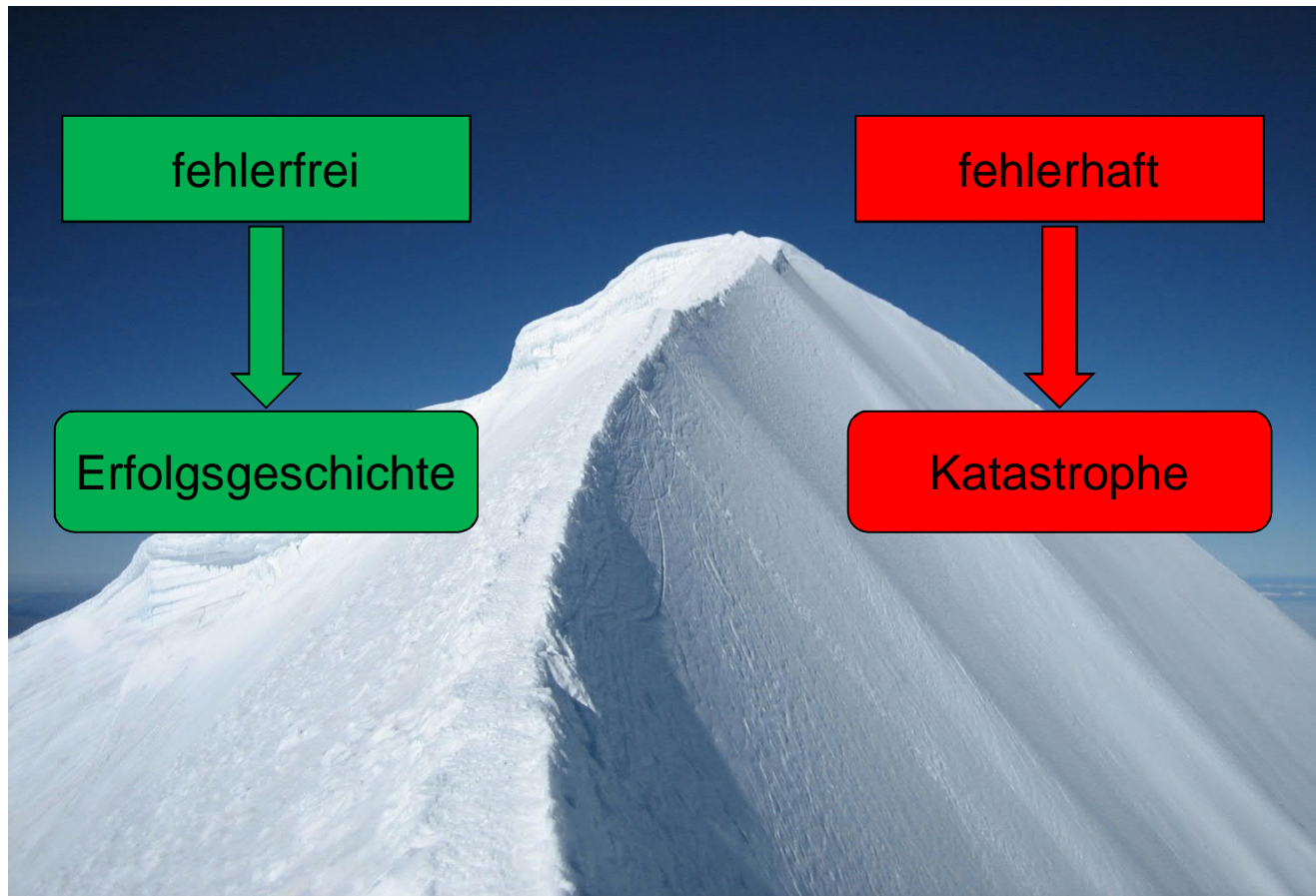


Inhalt

- Softwaretechnik = Gratwanderung?
- Softwarekrise
- Software Engineering
- Software Ingenieur
- Wissensgebiete des Software Engineering
- Lernziele



Softwaretechnik = Gratwanderung?



<http://www.summitpost.org/the-grat/559212>

Erfolgsgeschichte 1: Automotive Software

- **Mercedes
GLE 500e (Hybrid)**



www.mobilegeeks.de

- **Software-gesteuerte Funktionen:**

- ☐ Hybridoptimierung (Elektro/BenzinV6)
- ☐ Elektronisches Stabilitätsprogramm
- ☐ Verschiedene Assistenten (Spurhalten, Totwinkel, Abstand, ...)

1-4 ☐ ...

Einführung

Erfolgsgeschichte 1: Automotive Software



www.mercedes-benz.com

■ Autonomes Langstreckenfahren

- August 2013: Das “S 500 INTELLIGENT DRIVE” Versuchsfahrzeug fährt autonom 100 Kilometer von Mannheim nach Pforzheim

Erfolgsgeschichte1: Automotive Software



Roborace wird die weltweit erste Rennserie für autonome Elektrofahrzeuge werden.

■ Beta-Saison 2016/17

- Im Rahmen des Buenos Aires ePrix 2017 fuhren erstmals zwei DevBot-Fahrzeuge gegeneinander auf einer Rennstrecke. Demonstrationsfahrten gab es zudem in Marrakesch, Berlin, New York City und Montreal.



<https://www.e-formel.de/roborace.html>

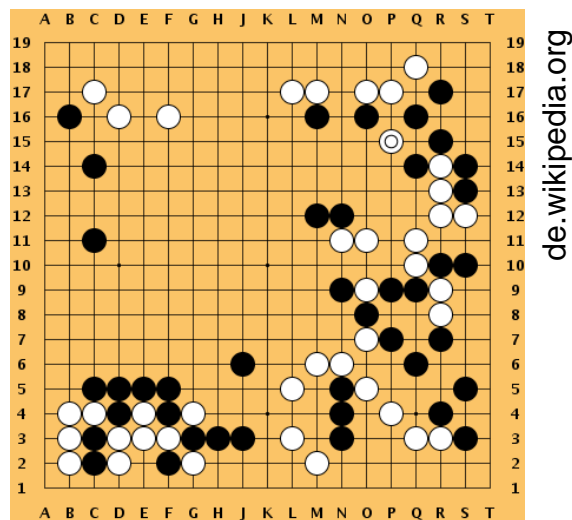
Erfolgsgeschichte 2:

KI spielt «GO»

- «GO» ist ein Strategie-Brettspiel, das vor ca. 2`500 Jahren in China erfunden wurde.
- Spielbrett: 19 x 19 Linien
- unbeschränkte Anzahl schwarzer und weißer Steine



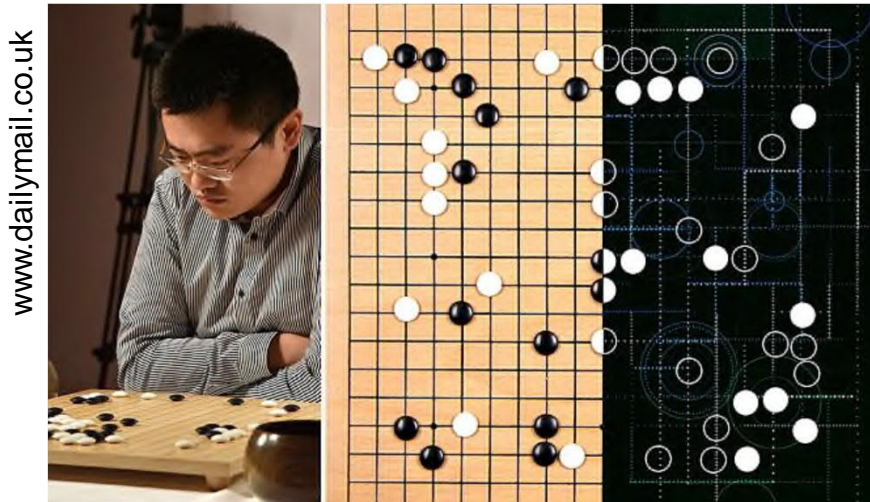
de.wikipedia.org



de.wikipedia.org

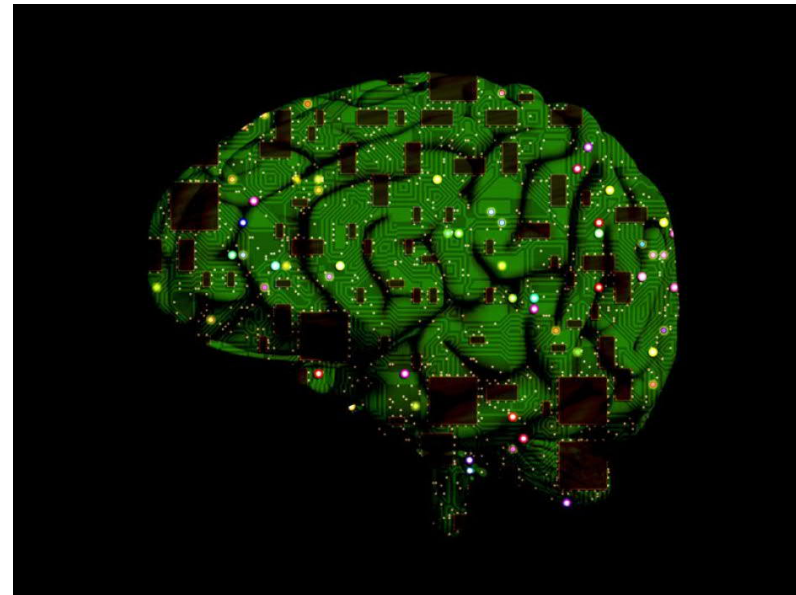
- Ziel: Möglichst große Gebietsanteile besetzen
- Anzahl verschiedener Stellungen auf dem GO-Brett: $\sim 2,08 \times 10^{170}$
- Schach: $\sim 10^{43}$ verschiedener Stellungen
- Anzahl Atome im Universum: $\sim 10^{80}$

Erfolgsgeschichte 2: KI spielt «GO»



- **Beeindruckend/Erschreckend:**
«AlphaGO» wurde NICHT programmiert, sondern ist ein *selbstlernendes* Programm.

- **Oktober 2015:**
Das KI-Programm «AlphaGO» gewinnt gegen den mehrfachen und amtierenden GO-Europameister Fan Hui 5:0.



Software Katastrophe 1: Absturz Airbus A400M (9. Mai 2015)



www.reuters.com

- **A400M:** Militärisches Transportflugzeug
- Zuladung von 37 Tonnen, Reichweite über 3'000 km

Software Katastrophe 1: Absturz Airbus A400M (9. Mai 2015)



www.spiegel.de

Absturzursache:

Ausfall der Schubsteuerung von
3 Triebwerken nach dem Start

Grund für Ausfall:

Widersprüchliche Befehle wg. unvollständige Triebwerkdaten nach
Softwareupdate.

Software Katastrophe 2: (1.08.2012)

Knight Capital 440 M\$ Verlust

<http://bilder1.n-tv.de>



Knight Capital:
Computer-Trader
im Hochfrequenzhandel

- 10'000 Trades/sec
- Haltezeit: Millisekunden

Handelsverlust am 1.8.2012 (NYSE): **440 Millionen US\$**

Software Katastrophe 2: (1.08.2012)

Knight Capital 440 M\$ Verlust

<http://www.nj.com>



Am 1.8.2012 um 9:30 generierten die Computer (ohne menschliches Zutun) Millionen von fehlerhaften Trades.

Um 9:58 hatte Knight Capital 440 Millionen US\$ verloren.

Grund: Programmierfehler im automatisierten High-Frequency Trading-Algorithmus der Firma (nach einem Software-Update)

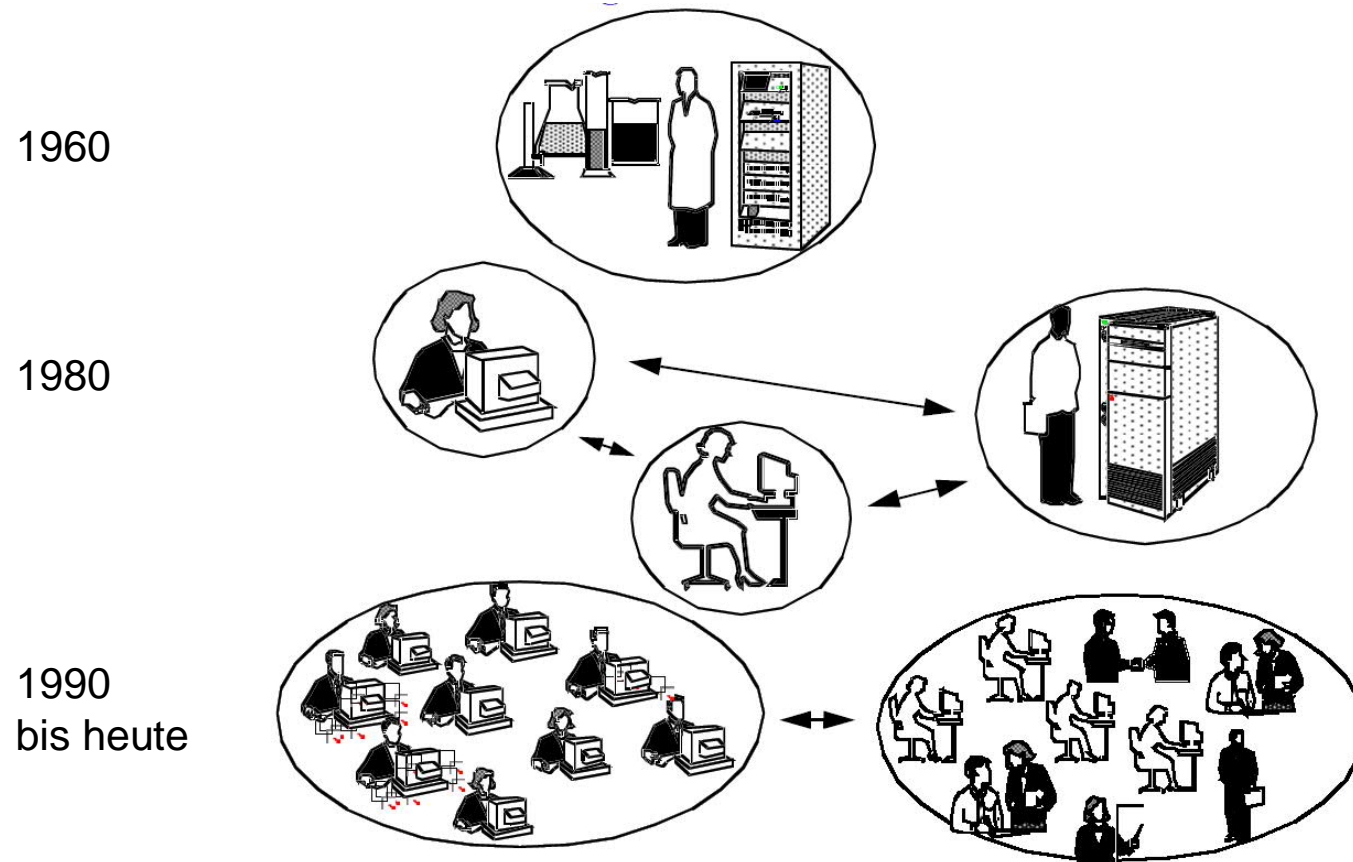


Inhalt

- Softwaretechnik = Gratwanderung?
- **Softwarekrise**
- Software Engineering
- Software Ingenieur
- Wissensgebiete des Software Engineering

Softwarekrise 1(6)

Software-Entwicklung und Software-Benutzung





Softwarekrise 2(6)

- Ende der 60er Jahre wurden Software-Entwicklungen nicht mehr mit den bekannten Mitteln beherrschbar.
- D.h. Software ...
 - ☐ tut nicht das, was der Kunde wünscht.
 - ☐ ist unzuverlässig.
 - ☐ -Entwicklung und –Wartung dauern zu lange und sind zu teuer (auch meist länger und teurer als geplant).
 - ☐ verbraucht zu viele Ressourcen.
 - ☐ ist schlecht anpassbar.



Softwarekrise 3(6)

- Ursachen für die Softwarekrise
 - Es existierten keine geeigneten Zerlegungsschemata:
 - Wie zerlegt man komplexe Aufgaben in einfache?
 - Es existierten keine Organisationsschemata:
 - Wie wird ein Projekt geplant, wie werden Aufgaben durchgeführt und überwacht?
 - Schlechte Werkzeuge:
 - Betriebssysteme, Programmiersprachen, Entwicklungsumgebungen, usw.
 - Mangelnde Spezialisierung:
 - Kaum zugeschnittene Lösungsansätze für spezifische Probleme
 - Software ist ein immaterielles Produkt

Softwarekrise 4(6)

- Software ist ein immaterielles Produkte (cont.)



[bleibgeschmeidig.com/96412]

*Diese Konstruktion
ist offensichtlich falsch!*



[pixabay.com]

*Aber ... wie erkennen wir Fehler
in der hier gespeicherten
Software-Konstruktion?*



Softwarekrise 5(6)

- Software ist ein immaterielles Produkte (cont.)
 - Software ist schwer verständlich
 - Kommunikationsprobleme:
 - Auftraggeber \Leftrightarrow Entwickler
 - Entwickler \Leftrightarrow Entwickler
 - Software ist nur beobachtbar
 - in den Wirkungen beim Ablauf auf Computern.
 - indirekt über die Dokumentation der Software.
 - Software-Fehler sind schwieriger erkennbar.
 - Software hat keinen Materialwert.
 - Software verschleißt nicht:
 - verführt zum „Erkerbau“ (immer neue Erweiterungen)
 - altert aber trotzdem



Softwarekrise 6(6)

- Software ist ein immaterielles Produkte (cont.)
 - Software ist scheinbar leicht zu ändern.
 - Fehlerbehebung und Weiterentwicklung werden unterschätzt.
 - Kleinste Änderungen in einer Software können massive Änderungen im Verhalten zur Folge haben.
 - Nachweis des wunschgemäßen Verhaltens einer Software ist schwierig.
 - Fazit:
 - *Software ist schwieriger zu handhaben und entwickeln als andere technische Produkte!*



Inhalt

- Softwaretechnik = Gratwanderung?
- Softwarekrise
- **Software Engineering**
- Software Ingenieur
- Wissensgebiete des Software Engineering



Software Engineering 1(3)

- Auslöser für die Einführung des Begriffes „Software Engineering“ (Deutsch: Software-Technik) war die beschriebene Software-Krise Ende der 1960ziger Jahre.
- Der Begriff „Software Engineering“ wurde 1967 von F.L. Bauer (ehemaliger Prof. in München) im Rahmen einer „Study Group on Computer Science“ der NATO geprägt.
- Software Engineering-Konferenzen der NATO 1968 in Garmisch und 1969 in Rom:
 - Software-Programme wurden erstmals (wie in anderen Ingenieurdisziplinen) als *Industrieprodukte* bezeichnet.
 - Es wurde gefordert, Software Engineering nicht als Kunst, sondern als *ingenieurmäßige Tätigkeit* anzuerkennen.



Software Engineering 2(3)

- Definition Software Engineering (= Softwaretechnik):

- Software Engineering ist:

- die Entwicklung
 - die Pflege und
 - der Einsatz

qualitativ hochwertiger Software unter Einsatz von

- wissenschaftlichen Methoden
 - wirtschaftlichen Prinzipien
 - geplanten Vorgehensmodellen
 - Werkzeugen und
 - quantifizierbaren Zielen.

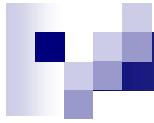
- B. Kahlbrandt: *Software-Engineering: Objektorientierte Software-Entwicklung mit der Unified Modeling Language*, Springer Verlag (1998)



Software Engineering 3(3)

Weitere Definitionen

- „*Software Engineering the systematic approach to the development, operation, maintenance, and retirement of software.*“
 - IEEE: *Standard Glossar of Software Engineering Terminology - IEEE Standard 729*, IEEE Computer Society Press (1983)
- „*Software Engineering is the science and art of specifying, designing, implementing, and evolving, with economy, timeliness and elegance, programs, documentations, and operating procedures whereby computers can be made useful to humanity.*“
 - A.W. Brown, A.N. Earl, J.A. McDermid: *Software Engineering Environments: Automated Support for Software Engineering*, McGraw-Hill (1992)
- „*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 z.B. von Kosten, Zeit, Qualität.*“
 - H. Balzert: *Lehrbuch der Software-Technik (Band 1): Software-Entwicklung*, Spektrum Akademischer Verlag (1996)



Inhalt

- Softwaretechnik = Gratwanderung?
- Softwarekrise
- Software Engineering
- **Software Ingenieur**
- Wissensgebiete des Software Engineering



Software Ingenieur

- Software Ingenieur (= Software Techniker, Software-Entwickler):
 - *“A software engineer must of course be a good programmer, be well-versed in data structures and algorithms, and be fluent in one or more programming languages. ... The software engineer must be familiar with several design approaches, be able to translate vague requirements and desires into precise specifications, and be able to converse with the user of a system in terms of application rather than ‘computers’.”*
 - Def. nach: C. Ghezzi, M. Jazayeri, D. Mandrioli: *Fundamentals of Software Engineering*, Prentice Hall (1991)
- Daher benötigt der Software Ingenieur folgende Fähigkeiten:
 - Kommunikation auf verschiedenen Abstraktionsebenen
 - Erstellung und Verwendung von Modellen/Spezifikationen
 - Kommunikation mit Personen mit unterschiedlichen Vorstellungen, Ausbildungen
 - Arbeitsplanung und -koordination



Ethische Regeln und professionelles Verhalten 1(2)

- „ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices“ - sinngemäße Übersetzung aus: I. Sommerville: *Software Engineering*, Addison-Wesley - Pearson Studium, 8. Auflage (2007)
- Präambel (verkürzt):
... Software-Entwickler sollen sich verpflichten, Analyse, Spezifikation, Entwurf, Entwicklung, Test und Wartung von Software zu einem nützlichen und geachteten Beruf zu machen. In Übereinstimmung mit ihren Verpflichtungen gegenüber Gesundheit, Sicherheit und dem Wohlergehen der Öffentlichkeit sollen Software-Entwickler sich an die folgenden acht Prinzipien halten:
 1. **Öffentlichkeit:** Software-Entwickler sollen in Übereinstimmung mit dem öffentlichen Interesse handeln.
 2. **Kunde und Arbeitgeber:** Software-Entwickler sollen auf eine Weise handeln, die im Interesse ihrer Kunden und ihres Arbeitgebers ist und sich mit dem öffentlichen Interesse deckt.
 3. **Produkt:** Software-Entwickler sollen sicherstellen, dass ihre Produkte und damit zusammenhängende Modifikationen den höchstmöglichen professionellen Standards entsprechen.



Ethische Regeln und professionelles Verhalten 2(2)

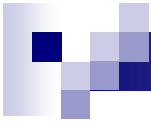
Ethische Regeln – Fortsetzung:

4. **Beurteilung:** Software-Entwickler sollen bei der Beurteilung eines Sachverhalts Integrität und Unabhängigkeit bewahren.
5. **Management:** Für das Software Engineering verantwortliche Manager und Projektleiter sollen sich bei ihrer Tätigkeit ethischen Grundsätzen verpflichtet fühlen und in diesem Sinne handeln.
6. **Beruf:** Software-Entwickler sollen die Integrität und den Ruf des Berufs in Übereinstimmung mit dem öffentlichen Interesse fördern.
7. **Kollegen:** Software-Entwickler sollen sich ihren Kollegen gegenüber fair und hilfsbereit verhalten.
8. **Selbst:** Software-Entwickler sollen sich einem lebenslangen Lernprozess in Bezug auf ihren Beruf unterwerfen und anderen eine ethische Ausübung ihres Berufes vorleben.



Inhalt


- Softwaretechnik = Gratwanderung?
- Softwarekrise
- Software Engineering
- Software Ingenieur
- **Wissensgebiete des Software Engineering**



Wissensgebiete des Software Engineering 1(2)

Der IEEE Computer Society „Guide to the Software Engineering Body of Knowledge“ (SWEBOK, <http://www.swebok.org>) zählt folgende Wissensgebiete auf:

- **Software Requirements:**
 - „Was“ soll ein Software-System leisten (und warum).
- **Software Design:**
 - „Wie“ soll die Software die Anforderungen erfüllen (Bauplan, Architektur).
- **Software Construction**
 - Das Software-System wird gemäß Bauplan realisiert.
- **Software Testing**
 - Systematische Suche und Beseitigung von Fehlern.
- **Software Maintenance:**
 - Pflege und Weiterentwicklung der Software nach der Auslieferung.
- **Software Configuration Management:**
 - Die Verwaltung von Software-Versionen und –Konfigurationen



Wissensgebiete des Software Engineering 2(2)

- **Software Engineering Management:**
 - (Projekt-)Management von Personen, Organisationen, Zeitplänen, ...
- **Software Engineering Process:**
 - Definition und Verbesserung von Software-Entwicklungsprozessen
- **Software Engineering Tools and Methods:**
 - Werkzeuge und Methoden für die Software-Entwicklung.
- **Software Quality:**
 - Messen und Verbessern der Software-Qualität.
- **Software Engineering Professional Practice**
- **Software Engineering Economics**
- **Computing Foundations**
- **Mathematical Foundations**
- **Engineering Foundations**



Lernziele von Kapitel 1

- Entwicklung der Disziplin „Softwaretechnik“
 - Begriff „Softwarekrise“
 - Gründe für die „Softwarekrise“
- Definition „Softwaretechnik“
- Berufsbild „Software Ingenieur“
 - Anforderungen
 - Ethische Regeln für den Beruf „Software Ingenieur“
- Einordnung der Disziplin „Softwaretechnik“ in die Informatik
 - Wissensgebiete der Softwaretechnik