

# ***Software-Engineering***

Vorlesung 1 vom 18.10.2004  
Sebastian Iwanowski  
FH Wedel

# Vorlesungsüberblick

## Inhaltliche Voraussetzungen:

keine

*hilfreich: Programmierkenntnisse, Praktikumserfahrung*

## Lernziele dieser Vorlesung:

Überblick über das Fachgebiet Software-Engineering

Beherrschen der Terminologie

Kenntnis der wesentlichen Methoden und Werkzeuge

Querverbindung zwischen Programmieren und Industriepraxis

## Vertiefende Vorlesungen für spezielle Themen des SWE:

Systemanalyse

Software-Design

Software-Ergonomie

# Was ist Software-Engineering?

Definition nach **Pomberger / Blaschek**:

"Software-Engineering ist die praktische Anwendung wissenschaftlicher Erkenntnisse für die wirtschaftliche Herstellung und den wirtschaftlichen Einsatz qualitativ hochwertiger Software."

Definition nach **Balzert**:

"Software-Technik ist die zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen."

Definition nach **Sommerville**:

"Software-Engineering ist die Ingenieursdisziplin, die sich mit allen Aspekten der Softwareproduktion beschäftigt, von der Systemspezifikation bis zur Wartung nach der Ingebrauchnahme."

# Unterschied zwischen Programm und Software

Ein **Programm** ist die Implementierung eines Algorithmus

**Software** ist eine Menge von miteinander verbundenen Programmen und Daten mit Anbindungsmöglichkeiten an die Außenwelt und Dokumentation

# **Bsp.: Modellbasierte Diagnose bei DaimlerChrysler**

**Anwendungsgebiet: Diagnose von Technischen Systemen**

**Aufgabe: Unterstützung durch Software (Expertensystem)**

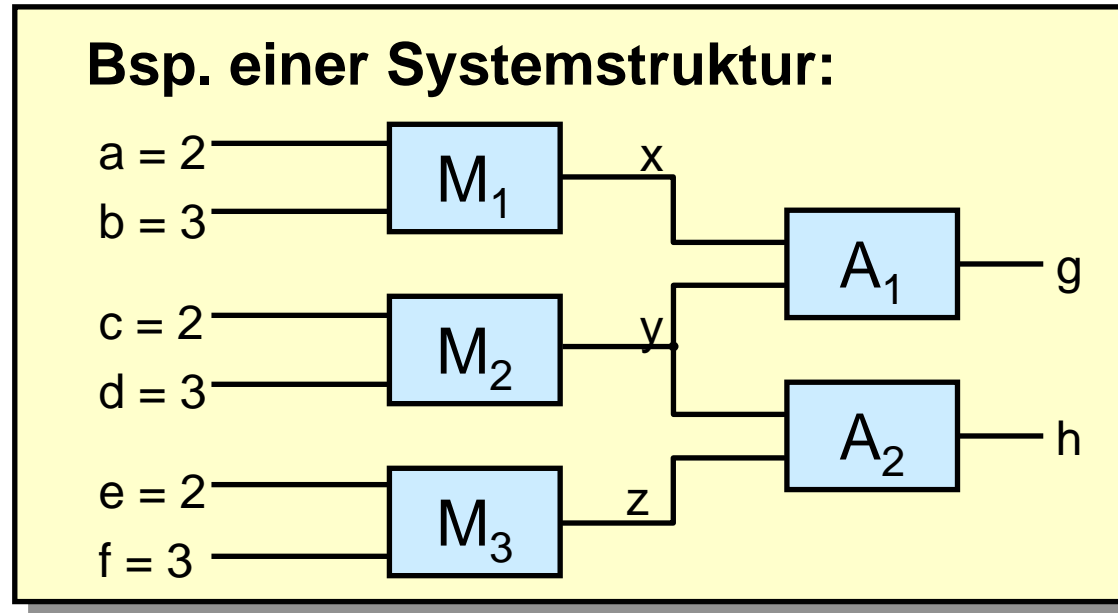
**Probleme der bisherigen Lösungen (Praxis):**

- Diagnosen ungenau bzw. unvollständig
- Intialisierung mit Expertenwissen schwierig

**Modellbasierter Lösungsansatz (Literatur):**

- **Modelliere das technische System auch in der Software gemäß seiner physikalischen Struktur:  
Setze das System aus seinen Einzelkomponenten zusammen.**
- **Erstelle eine Modellbibliothek für die Einzelkomponenten.**
- **Implementiere Algorithmus, der aus dem Verhalten der Einzelkomponenten das Verhalten des Gesamtsystems simuliert.**

# Bsp.: Modellbasierte Diagnose bei DaimlerChrysler



## Projekt 1992:

- Implementierung eines Prototypen, der einen Algorithmus für das modellbasierte Lösungsverfahren enthält
- Aufbau einer Modellbibliothek für ausgewählte technische Komponenten
- Aufbau eines Systemeditors, der es erlaubt, beliebige Systeme aus den Komponenten der Modellbibliothek zusammenzusetzen

# Bsp.: Modellbasierte Diagnose bei DaimlerChrysler

## Eigenschaften der Lösung von 1992:

- **Modellbibliothek bestand aus arithmetischen Komponenten.**
- **Systemeditor war menüorientiert.**
- **Systemnahe graphische Oberfläche war nicht implementiert.**
- **Programmierungsumgebung war Common Lisp.**
- **Implementierung lief auf PC (DOS) und war nicht ohne Weiteres auf andere Plattformen übertragbar.**

## Erweiterung 1993:

- **Modellbibliothek aus elektrischen Komponenten.**
- **Übertragung auf Sun Workstations (Unix)**

# **Bsp.: Modellbasierte Diagnose bei DaimlerChrysler**

## **Anforderungsanalyse bei der Fachabteilung 1993:**

- **Welches sind die tatsächlichen Probleme der Abteilung ?**
- **Mit welchen technischen Systemen beschäftigt sich die Abteilung ?**
- **Auf welcher Plattform soll die Lösung laufen ?**

## **Marktanalyse 1993:**

- **Vergleich kommerziell verfügbarer Werkzeuge**
- **Einsatzempfehlung: Eigene Implementierung mit Smalltalk**

## **Implementierung des MDS-Prototypen 1994:**

- **Erfüllen aller Forderungen der Anforderungsanalyse**
- **Verbesserung des Verfahrenskerns**



# **Bsp.: Modellbasierte Diagnose bei DaimlerChrysler**

## **1995: Ausbau des MDS-Prototypen**

- **Neue Anforderungen an Verfahrenskern**
- **Verbesserung der Bedienungs Oberfläche**

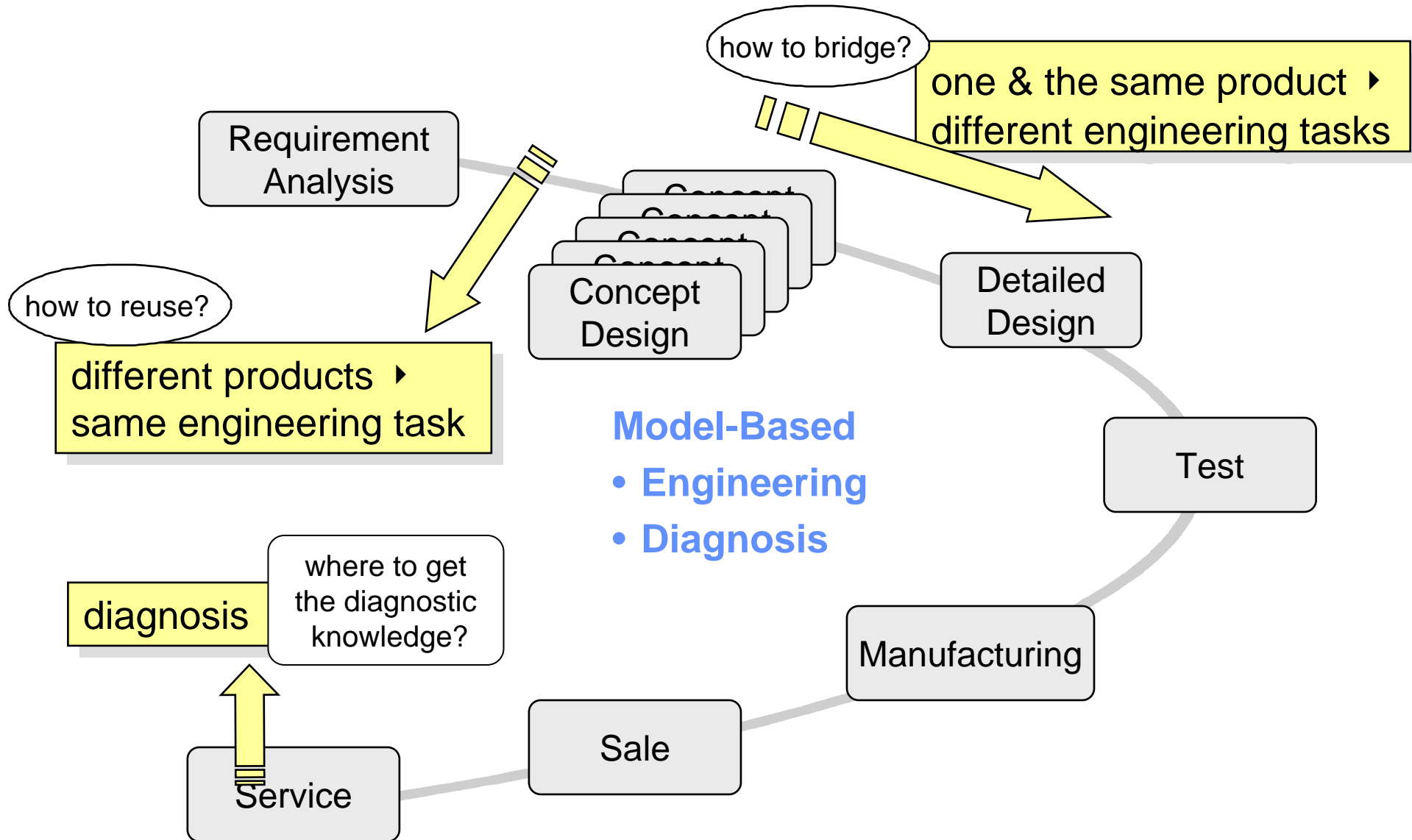
## **1996: Weitere Kunden**

- **Kontakte mit dem Bahnbereich: Neue Modellbibliotheken**
- **Kontakte mit dem Luftfahrtbereich: Neue Modellbibliotheken**
- **Mitarbeit in Kundenprojekten: Integration von MDS mit anderer Software**

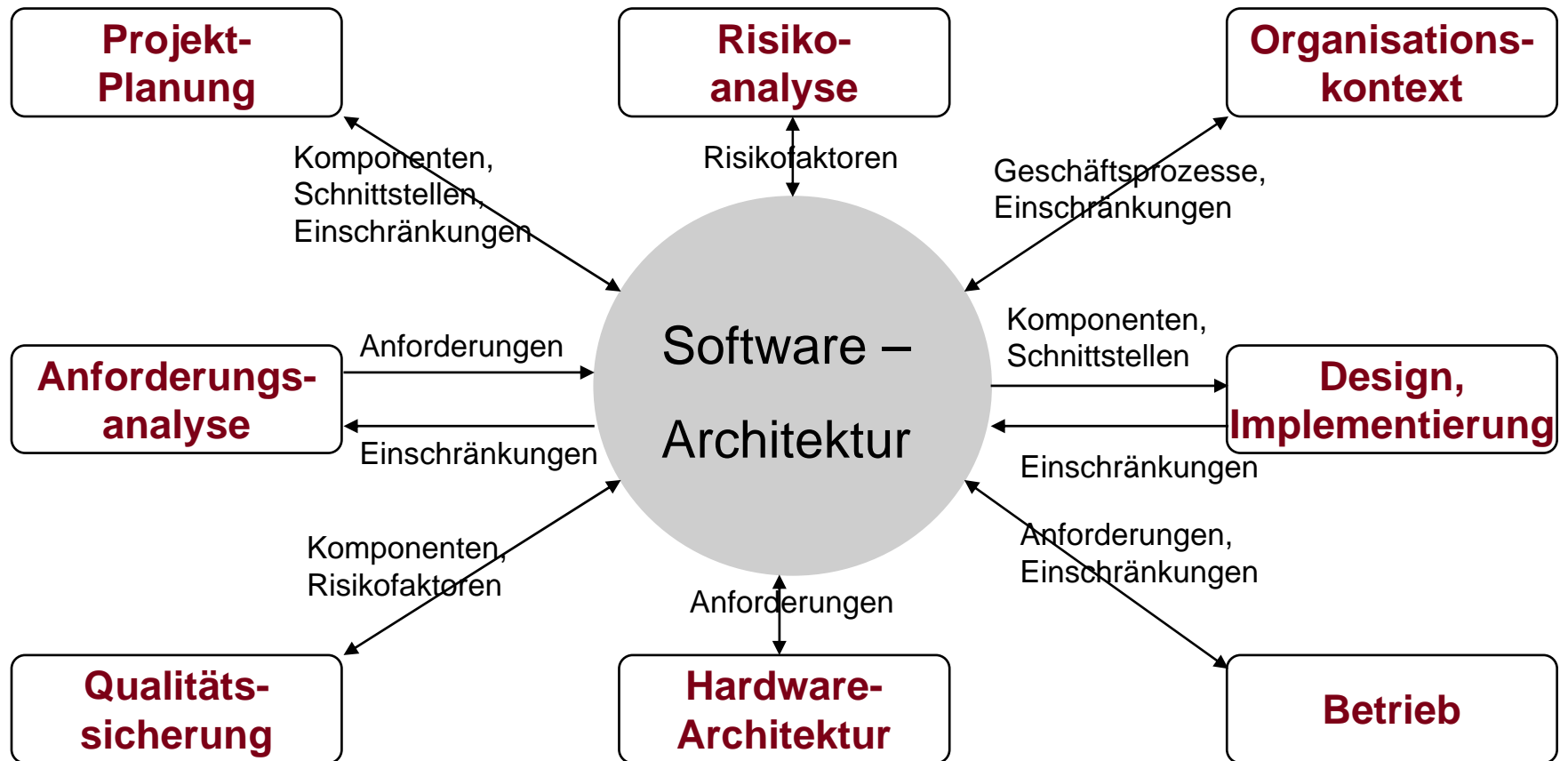
## **Ab 1997: Kommerzialisierung**

- **Weiterer Ausbau der Implementierung**
- **Transfer zu Softwarefirma**

# Ausblick: Diagnose in der Engineering Prozesskette



# Große Softwaresysteme



Nach *Effektive Software-Architekturen*, Gernot Starke. Hanser, 2002

# Warum brauchen große Softwaresysteme eine Architektur?

## Analogie aus dem Gebäudebau:

Wenige Architekten entwerfen ein Gebäude

- der Bauplan ist generisch und kann in vielen Situationen analog wiederverwendet werden

Viele Spezifikateure entwerfen die Detailpläne

- sie bringen die funktionalen Anforderungen des Kunden ein

Noch mehr "Bauarbeiter" realisieren das Gebäude

Ein solches Vorgehen ist sinnlos bei Kleinstgebäuden

# Architektonische Sichten:

Ein Architekt kennt nicht nur *eine* Planungsansicht von einem Haus, sondern es gibt z.B.

- Gesamtansichten
- Entwässerungspläne
- Stromversorgungspläne
- Statik-Pläne

Analog gibt es auch verschiedene Sichten auf ein Software-System - man nennt diese „architektonische Sichten“ (architectural views)

## Unterscheide:

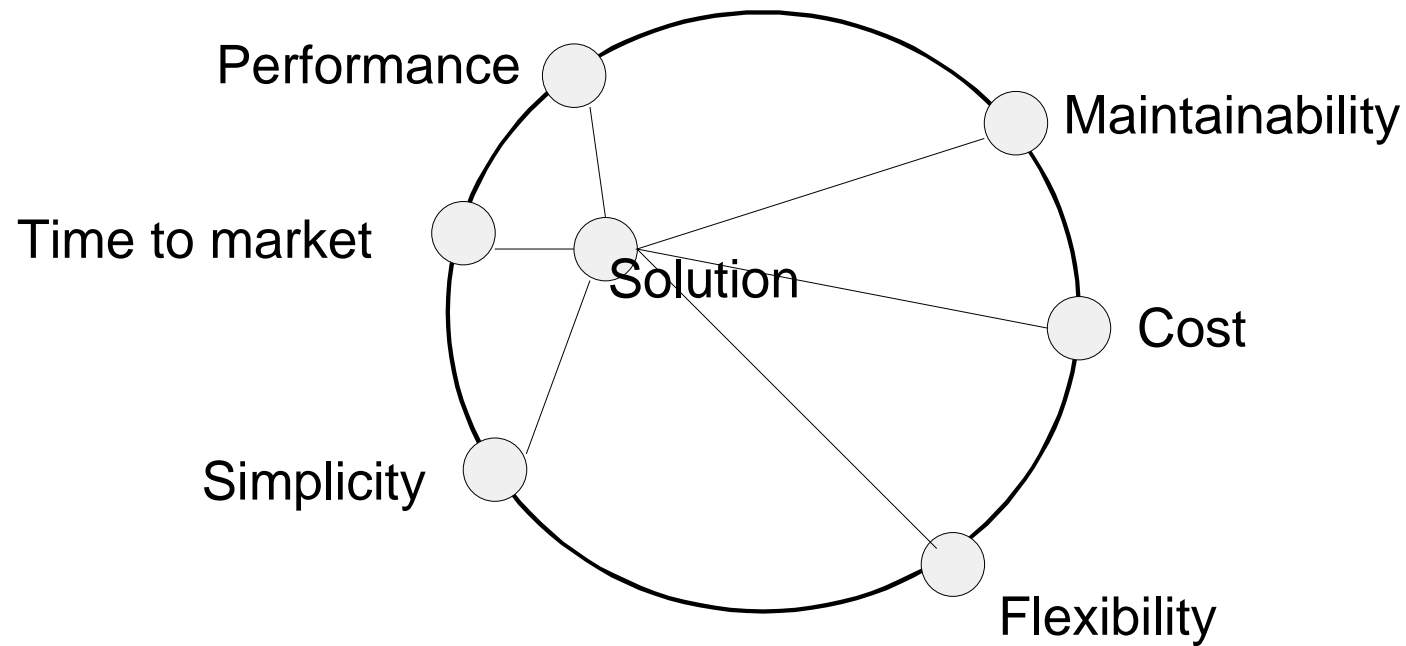
- Architektur des Anwendungsmodells, aus dem die Aufgabenstellung stammt
- Architektur der Software

# Woraus besteht Software-Architektur ?

- Funktionale Anforderungen in den verschiedenen Sichten
- Nichtfunktionale Anforderungen
- Architektonische Stile & Entwurfsmuster
- Ausbalancieren verschiedener Kräfte

# Ausbalancieren verschiedener Kräfte

## Conflicting Goals



# Besonderheiten des Produkts Software

- Software ist immateriell
- Software unterliegt keinem Verschleiß, altert aber dennoch
- Software ist "weich", daher schnell änderbar
- Die Herstellung von Software beruht weniger auf allgemein akzeptierten Prinzipien
- Software ist (häufig) komplex
- Software ist schwer zu vermessen



# Phasen der Software-Entwicklung

- **Problemanalyse und Planung**
- **Systemspezifikation**
- **Entwurf**
- **Implementierung**
- **Integration und Test**
- **Wartung**

# Vorlesungsüberblick

## **Vorlesungsthemen:**

1. Überblick über das Thema und die Vorlesung
2. Grundlegende Prinzipien
3. Softwareplanung
4. Systemanalyse
5. Aufwandsabschätzung
6. Systementwurf
7. UML
8. ARIS
9. Qualitätsmanagement
10. Projektmanagement

# Vorlesungsüberblick

**Material zu dieser Vorlesung:** <http://www.fh-wedel.de/~iw/Lehrveranstaltungen/WS2004/SWE.html>

## Empfehlenswerte Vorlesung zur Vertiefung im Internet:

Hans Hartmann, Wolfgang Keller: *Software-Engineering für große betriebliche Informationssysteme*  
Universität Leipzig SS 2004, <http://www.objectarchitects.de/leipzig2004>

## Quellen für einige Folien in dieser Vorlesung:

Ralf Möller: *Software-Engineering* FH Wedel WS 2003/2004,  
<http://www.fh-wedel.de/~mo/lectures/se-ws-03-04.html>

Hans Hartmann, Wolfgang Keller: *Software-Engineering für große betriebliche Informationssysteme*  
Universität Leipzig SS 2004, <http://www.objectarchitects.de/leipzig2004>

# Literatur allgemein



Helmut Balzert: *Lehrbuch der Software-Technik, Band1: Software-Entwicklung*  
Spektrum 2000 (2. Auflage), ISBN 3-8274-0480-0



Helmut Balzert: *Lehrbuch der Software-Technik, Band2: Software-Management,  
Software-Qualitätssicherung, Unternehmensmodellierung*  
Spektrum 1998, ISBN 3-8274-0065-1

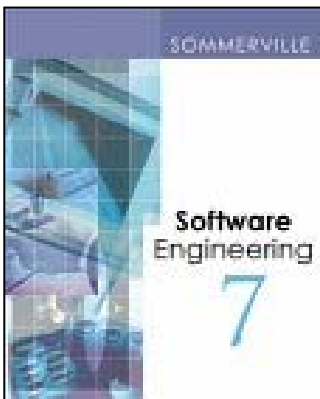
# Literatur allgemein



Gustav Pomberger / Günther Blaschek: *Software-Engineering: Prototyping und objektorientierte Softwareentwicklung*

Hanser 1996 (2. Auflage), ISBN 3-446-18690-5

in der FH-Bibliothek: 1. Auflage (1993), ISBN 3-446-16262-3



Ian Sommerville: *Software Engineering*

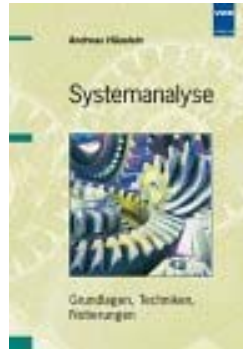
Addison-Wesley 2004 (7. Auflage), ISBN 0-321-21026-3

in der 6. Auflage (2001) auch auf Deutsch: ISBN 3-8273-7001-9

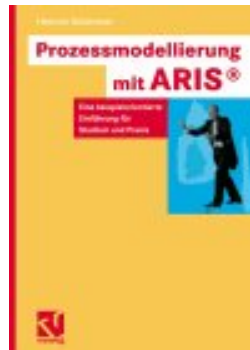
in der FH-Bibliothek: 3. Auflage Englisch (1989), ISBN 0-201-17568-1



# Literatur für spezielle Themen



Andreas Häuslein: *Systemanalyse*  
VDE 2004, ISBN 3-8007-2715-3



Heinrich Seidlmeier: *Prozessmodellierung mit ARIS*  
Vieweg 2002, ISBN 3-528-05804-8

Kent Beck: *Extreme Programming*  
Addison-Wesley 2000 (Deutsche Übersetzung), ISBN 3-8273-1709-6

***Beim nächsten Mal:  
Grundlegende Prinzipien / Softwareplanung***