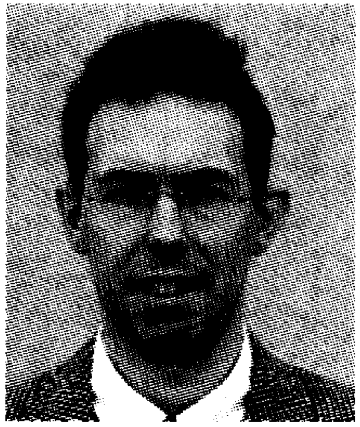
物理学硕士，数学硕士

安诺夫·丹策 博士

Dr. Arnulf Deinzer

44a, verheiratet, 1 Kind (5.5a)
1980-86 Mathematik&Physik Uni Würzburg
1998 Promotion Informatik Uni d. BW

14.5a Siemens AG
- 8a SW-Entwicklung (Betriebssystem)
- 1a Integrationstest (Boca Raton FL, USA)
- 5a Systems Engineering



Dipl.-Phys., Dipl.-Math.
Dr. Arnulf Deinzer
Systems Engineering TD-SCDMA
Director Call Processing

Siemens AG
Information and
Communication Mobile

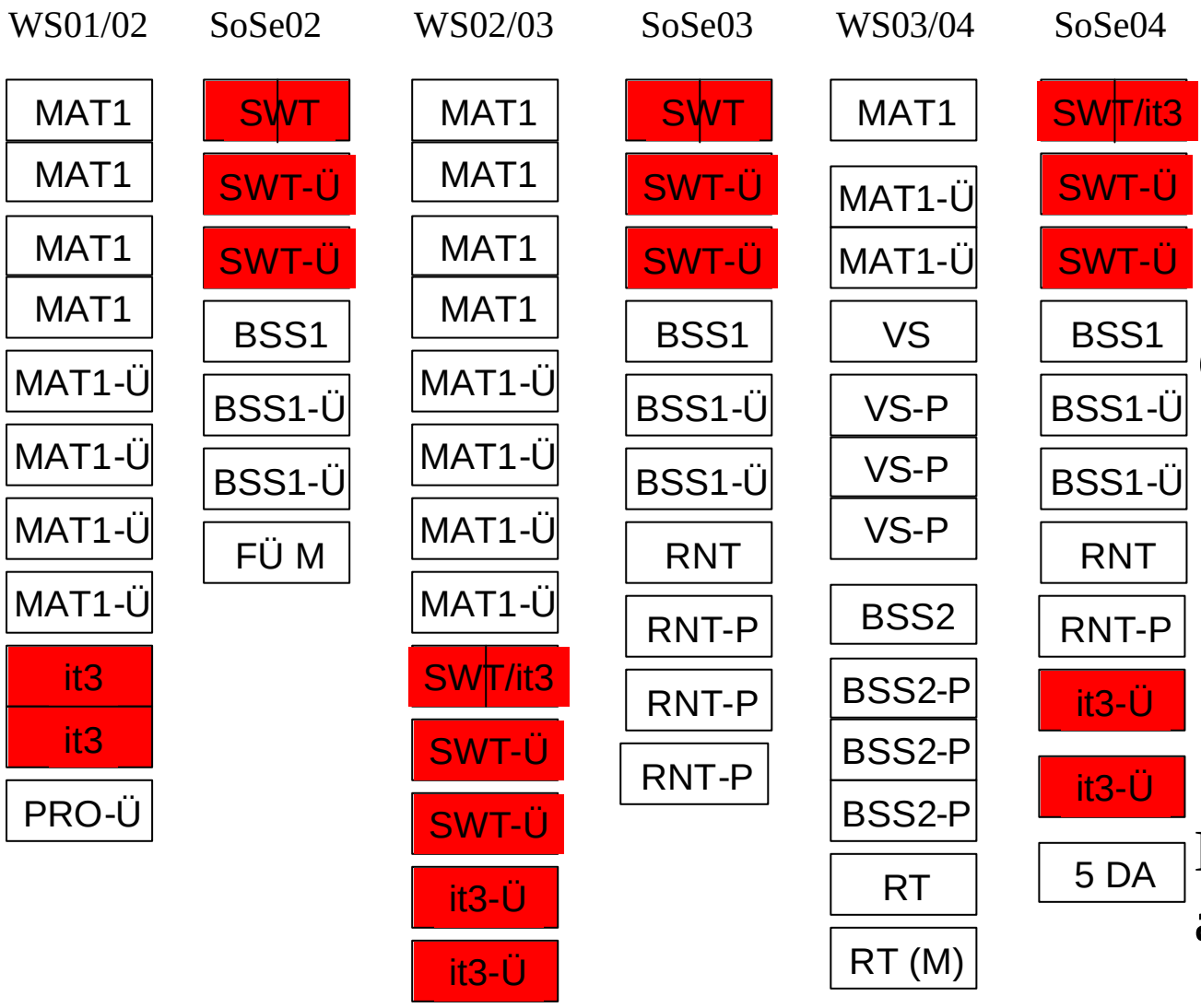
Postal Address:
Siemens AG, ICM N MR TS SE 2
D-81359 Munich

Office Address:
Hofmannstr. 51
D-81379 Munich

Tel. +49 89 722-37212
Fax +49 89 722-54889

arnulf.deinzer@icn.siemens.de

Veranstaltungen



Grobgliederung:
14h klassische SWE (AD)
14h UML (UG)

Aktive Teilnahme an Übungen/Praktikum!

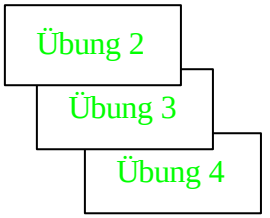
Erste Übungen:
ab 25.03.04

Organisatorisches

it3-Übungen Sommersemester 2004

Übung 1, **Do, 14:00-15:30** T315

Name, Vorname	Matrikel-Nr.



Bitte gleichmäßig verteilen!

Literatur

Six, H.-W., Pagel, B.U.	Software engineering – Die Phasen der Softwareentwicklung Addison-Wesley, Bonn 1994 (6-fach in Bibliothek, 1 Präsenzexemplar)
Sommerville, Ian	Software Engineering 6. Aufl., Pearson, München 2001 (1-fach in Bibliothek, 1 Präsenzexemplar)

Gliederung „klassische“ SWE – sicher angeboten

1 **Einführung**

1.1 **Einordnung**, Qualitätskriterien und Vorgehensmodelle
1.2 Einordnung, **Qualitätskriterien** und Vorgehensmodelle
1.3 Einordnung, Qualitätskriterien und **Vorgehensmodelle**

2 **A&D**

2.1 **Einordnung** und Ergebnisse
2.2 Einordnung und **Ergebnisse**
2.3 A&D-Methoden: **Datenflussmodellierung**
2.4 A&D-Methoden: **Datenmodellierung**
2.5 A&D-Methoden: **Zustandsmodellierung**
2.6 A&D – **OOA**
2.7 A&D – **OOA** (Fallbeispiel FHK)

3 **Entwurf**

3.1 **Einordnung** und Ergebnisse
3.2 Einordnung und **Ergebnisse**
3.3 Methoden: **Moduln**
3.4 Methoden: **Benutzungsbeziehungen zwischen M.**
3.5 **Funktionsorientierung**, (abschreckendes) Beispiel
3.6 **Qualitätskriterien**: Kohäsion und Kopplung
3.7 Entwurfsprinzip **Datenabstraktion**
3.8 **Datenstrukturorientierung**, (nachahmenswertes) B.
3.9 Entwurfsprinzip **Funktionsabstraktion**
3.10 Modularer Entwurf für **Fortgeschrittene**
3.11 **Grobentwurf**
3.12 **Feinentwurf**

Gliederung „klassische“ SWE – wahrscheinlich leider nicht

4 Implementierung

4.1 Implementierung **modularer Architekturen**
4.2 Implementierung **virtueller Komponenten**
4.3 Implementierung – **Codierung und Zuverlässigkeit**
4.4 Implementierung – **Codierung und Wartbarkeit**

5 Analytische Qualitätssicherung (Test)

5.1 Analytische Qualitätssicherung – **Übersicht**
5.2 Analytische Qualitätssicherung **vs. Testen**
5.3 Analytische Qualitätssicherung – **Kontrollflussbezogene Verfahren**
5.4 Analytische Qualitätssicherung – **Datenflussbezogene Verfahren**
5.5 Analytische Qualitätssicherung – **Testmethodik**
5.6 Analytische Qualitätssicherung – **Testen im Kleinen und im Grossen**
5.7 Analytische Qualitätssicherung **während des Entwicklungsprozesses**

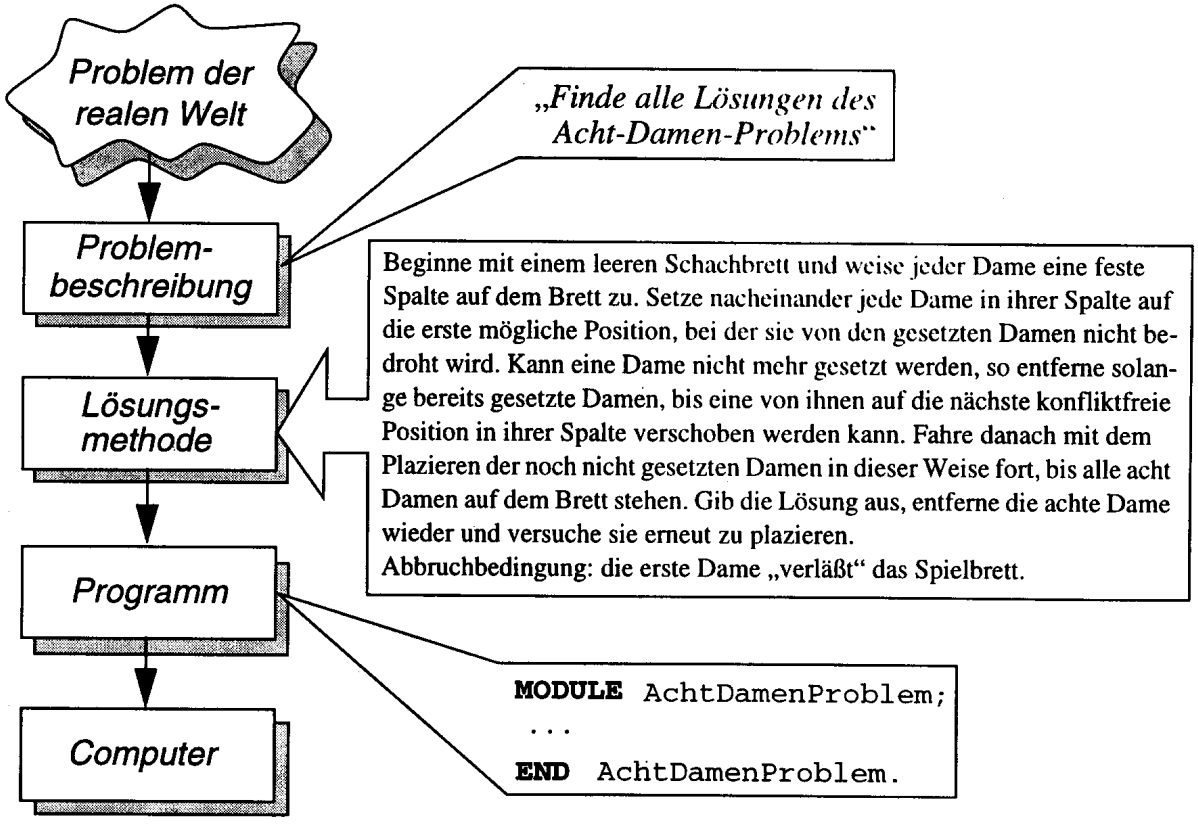
Geschichtlicher Überblick – Steinzeit bis SW-Krise

1965 'Software Crisis'
1966 GOTO obsolet (C.Böhm&G.Jacopini)
1968 'Software Engineering' (F.L.Bauer, F.McIlroy)
PiK: 'Wohlstrukturiertheit'
- ausgewählte Konstrukte
- **prozedurale Zerlegung**

PiG: Entwicklungsmodell
1971 'Stepwise Refinement' (N.Wirth)
1972 'Information Hiding' (D.L.Parnas)
1977 'Abstract Data Types' (B.Liskov&S.Zilles)

...(the so-called "software crisis" was caused by the observation that software was costing more than people thought it ought to). /Entsminger91/

Geschichtlicher Überblick - PiK



Programmieren:
Lösen von Problemen der realen Welt mit Hilfe des Computers.
PiK: (Programmieren im Kleinen)
•geringe Komplexität der Probleme
•relativ kleine Programme
(Eine/r an einem Tag)

Zerlegung
•top down
•am Kontrollfluss orientiert
•hierarchisch

Grundsätze der strukturierten P.:
•kein GOTO
•Programmierstil
•Layout und (online) Kommentierung

Wohlstrukturiertheit:
adäquate prozedurale Zerlegung + strukturierte Programmierung

Geschichtlicher Überblick - PiG

PiG (Programmieren im Grossen):

- (um Größenordnungen!) höhere Komplexität
  - von Problemstellung und
  - resultierender SW
- häufige Änderungen(*) während SW-Einsatz
- Wiederverwendung

(*) wg. langer Lebensdauer

Änderungsgründe/anlässe:

- Ansprüche bei (akzeptierter) SW wachsen bzgl. Funktionalität und Komfort
- HW ändert sich (erfolgreiche SW überlebt viele HW-Generationen)

Nach n Jahren (3..10) existiert praktisch keine Originalzeile mehr!

Geschichtlicher Überblick
SW Engineering – ein Begriff wird geboren(1)

1968 'Software Engineering' (F.L.Bauer, F.McIlroy)

PiK: 'Wohlstrukturiertheit'

- ausgewählte Konstrukte
- **prozedurale Zerlegung**

PiG: Entwicklungsmodell: Cascade

NATO-Konferenzen (1968 und 1969)

The establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines (/Bauer68/).

Geschichtlicher Überblick
SW Engineering – ein Begriff wird geboren(2)

... [is] the systematic approach to the development, operation and maintenance and retirement of software (/IEEE83/).

Softwaretechnik ist das Fachgebiet der Informatik, das sich mit der Bereitstellung und systematischen Verwendung von Methoden und Werkzeugen für die Herstellung und Anwendung von Software beschäftig (/Hesse84/).

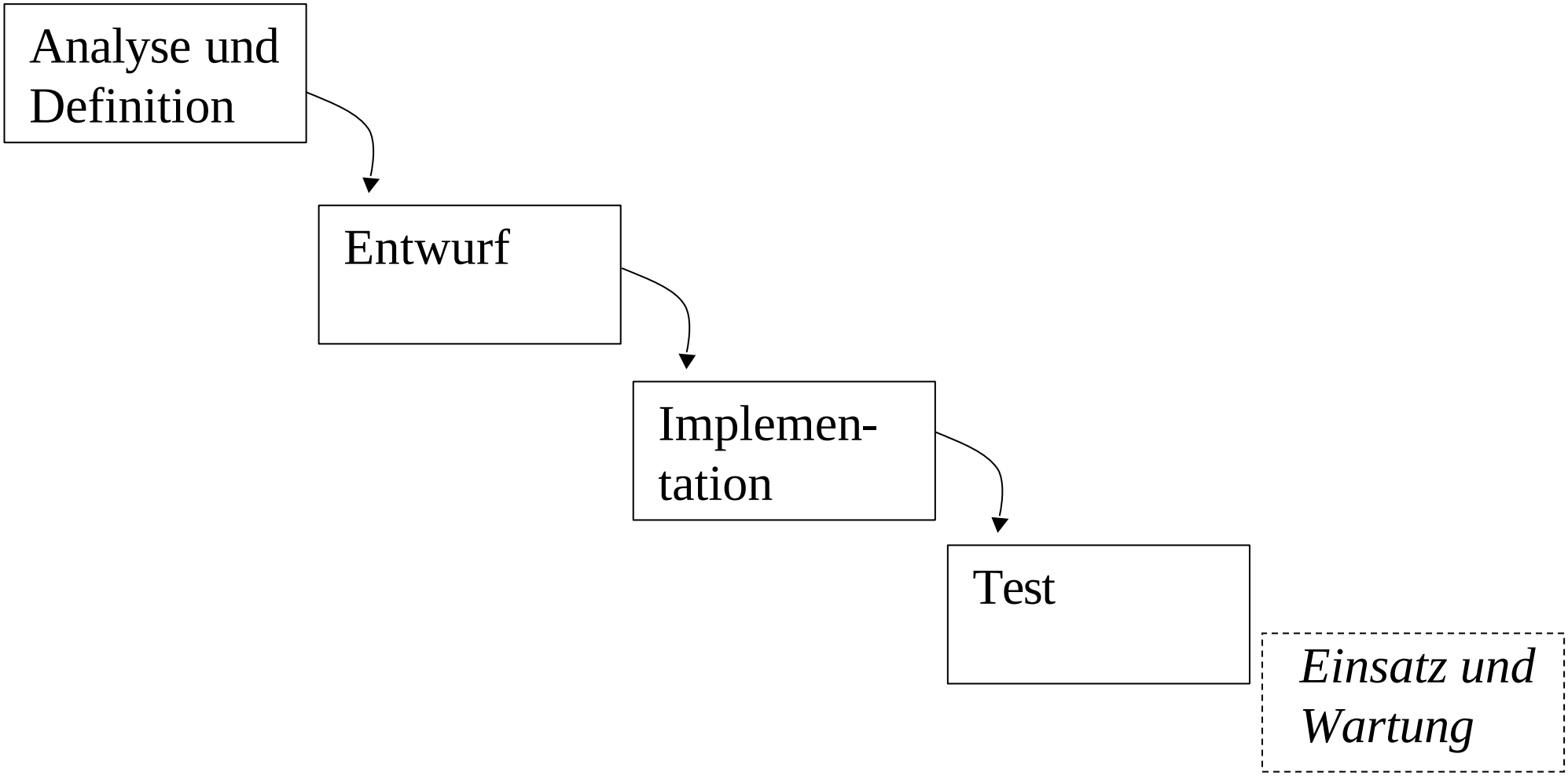
/Böhm66/ C. Böhm, G. Jacopini. *Flow diagrams, turing machines and languages with only two formation rules*. Communications of the ACM 1966

/Wirth71/ Niklaus Wirth. Program development by stepwise refinement. Communications of the ACM ´71

/Parnas72/ David Lorge Parnas. *On the Criteria to be used in Decomposing Systems into Modules*. Communications of the ACM, vol. 5, no. 12, pp. 1053-1058, Dec. 1972

/Liskov77/ B. Liskov, S. Zilles. *Programming with abstract data types*. ACM SIGPLAN Notices, 1977

Geschichtlicher Überblick
klassisches Phasenmodell

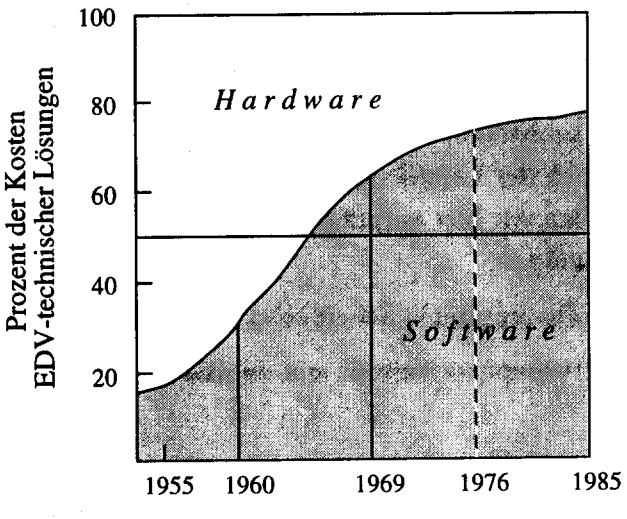


SW-Entwicklung heute
Highlights

- SW-Projekt >= 2a
- 50000 Zeilen Quellcode (1988), 2000000 (1992)
- 70-80% der SW-Projekte werden erfolgreich abgeschlossen (oder weniger!)
- Produktivität/Tag: <10 Zeilen Code
- während Entwicklung entdeckte Fehler: 50-60/1000LOC
- nach Auslieferung entdeckte Fehler: 4/1000LOC
- RCA (Root Cause Analysis): meiste Fehler aus A&D-Phase

Hauptprobleme: **geringe Produktivität** und **große Fehlerhäufigkeit**

SW-Entwicklung heute
HW- und SW-Kosten

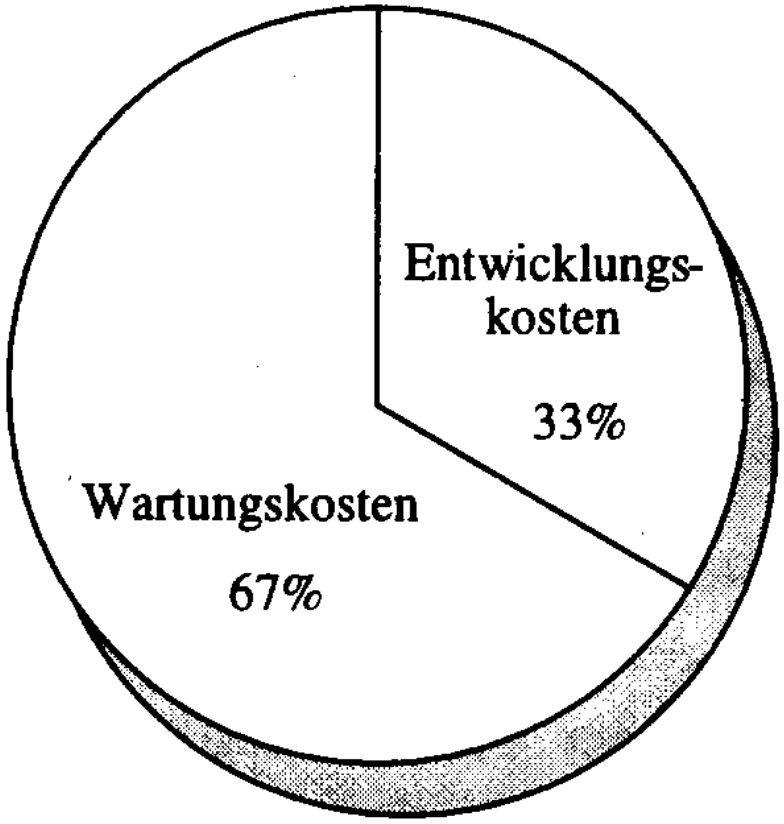


SW-Kosten dominieren bei weitem HW-Kosten (SW-Wertschöpfung dominiert bei weitem HW-Wertschöpfung – Faktor inzwischen >10!)

Auch bei momentaner Krise in IT-Bereich, auch bei Green Cards:

Programmer productivity is the critical cost factor in software engineering. An understanding of human factors can help identify possible ways of increasing productivity at relatively low cost. (/Somerville92/).

SW-Entwicklung heute
Implementierung kleinste/leichteste Aufgabe(1)

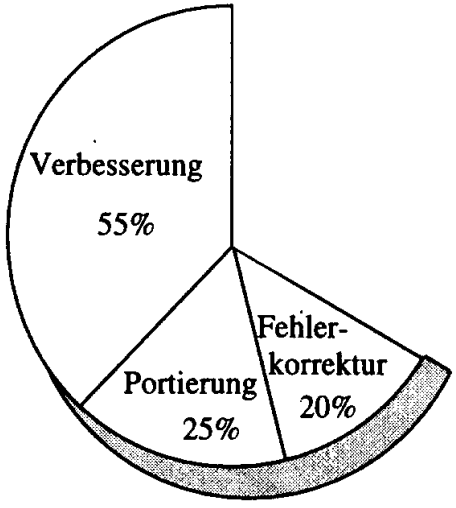


Hängt von Art der SW ab,
Verhältnisse 40:60 bis 10:90

SW-Entwicklung heute
Implementierung kleinste/leichteste Aufgabe(2)

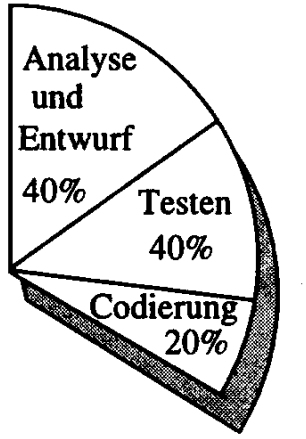
Portierung+Verbesserung: 80% der Wartung

Also: mit Installation der SW ist Ende der Entwicklung (im weiteren Sinne) noch lange nicht erreicht!

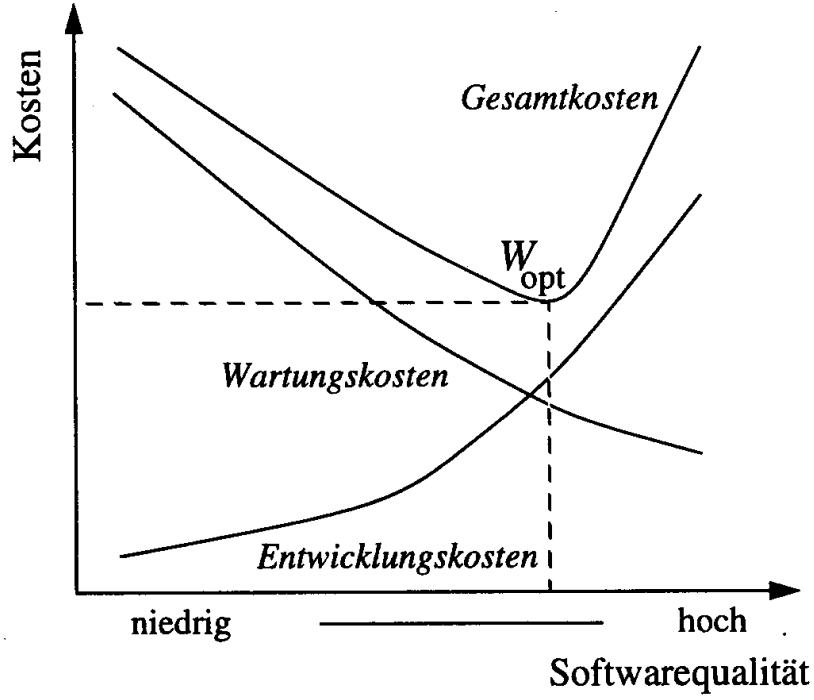


SW-Entwicklung heute
Implementierung kleinste/leichteste Aufgabe(3)

Codierung: 20% von 33%!



SW-Entwicklung heute
Qualität ja – aber nicht um jeden Preis



Graphische Ermittlung der wirtschaftlich optimalen SW-Qualität

SW-Entwicklung heute
Qualitätssicherung und ihre Kosten

Korrekturkosten korrelieren stark mit dem Zeitpunkt, zu dem der Fehler entdeckt wurde:

- Faktor 100 (bis >1000) für Kosten eines Fehlers der A&D-Phase, der im Test entdeckt wurde, im Vergleich zu Fehler aus Codierung.
- Faktor 100 für Kosten einer Fehlerkorrektur nach Produktauslieferung im Vergleich zu einer Korrektur eines Fehler in der Analysephase.

Korrekte SW entspricht nicht Wünschen des Auftraggebers: Korrektur (bzw. besser Änderung) von SW, die nicht Intentionen des Kunden wiederspiegelt, gehört zu kostenintensivsten Prozessen während der SW-Entwicklung.