1. Einführung, Grundbegriffe

Ziele:
Motivation für den Softwaretechnik-Ansatz
Vorstellung der Realität und des Stands der Technik
Einordnung der Software-Thematik
Vision: Software-Entwicklung in Zukunft

Stand: 15.09.2008

Motivation

Die Softwarekrise und Softwaretechnik

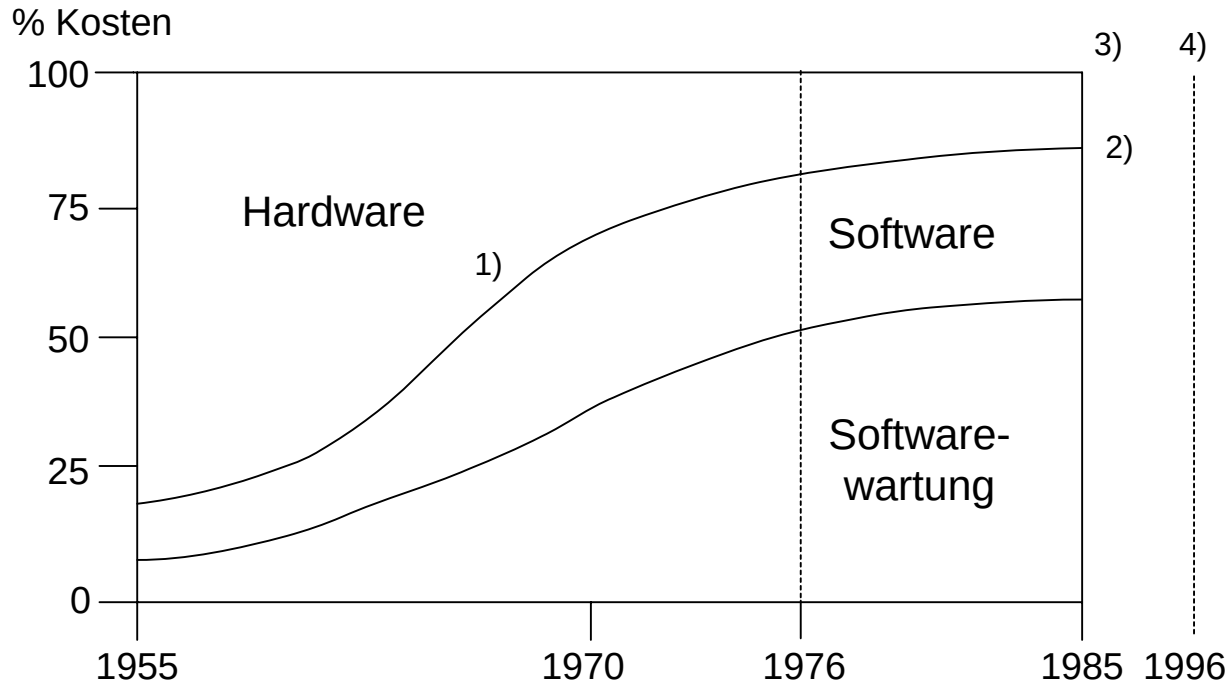
- Softwarekrise ($\geq$ 1965)

  gravierende Programmsystemfehler, kein Ansatz zur Lösung

- Gründe insbesondere:
  - Ausweitung des Aufgabenbereichs (von der Lösung einer DGL zur Steuerung der Mondfahrt)
  - kein Zergliederungsschema für diese komplexe Aufgabe (komplexe technische Aufgabe ohne Planung?)
  - kein hinreichend ausgebildetes Personal (ist jeder ein Anwendungssystem-Programmierer?)
  - kein Projektmanagement, keine Projektorganisation (ist eine Ansammlung von Einzelpersonen eine Mannschaft, geht es ohne Zielvorgabe und Überwachung?)
  - keine Qualitätssicherung (komplexes technisches Produkt ohne Zwischen- und Abnahmeprüfung?)
  - keine passenden Programmiersprachen (unübersichtlich, dem Niveau der Aufgabe nicht angepasst)
  - falsche Auffassung von Effizienz (Hardwarebeschränkung, falsches Bewusstsein, Spaghettiprogramme)

- Größenordnung des Problems der Softwareerstellung
  insbesondere Wartungsproblem

Hardware/Software-Kostenanteile nach /Boe 76/



1) Punkt der Umkehr und größten Steigerung: SW-Kosten überrunden HW-Kosten
2) Kostenanteil ’85 bei etwa 80%, heute darüber?
3) Schätzungen gehen davon aus /Boe 87/, dass ’85 die weltweiten Aufwendungen für SW bei 140 Mrd. US $ lagen, geschätzte jährliche Steigerungsrate 12%
4) Hat dieser Trend angehalten, so besitzt der SW-Markt ’96 ein Volumen von 485 Mrd. $ und übersteigt den Bundeshaushalt damit um mehr als 50%

Ist Graphik noch richtig? Ja!
trotz gegenläufiger Trends

- Antwort Softwaretechnik (1968/69 Konferenzen der Nato in Garmisch, Rom /NR 68, BR 69/):
  - ingenieurmäßiger Ansatz
  - Gegensatz zur Kunst

- Definitionen
  - F.L. Bauer in /NR 68/: "... ökonomisch Software zu erhalten, die effizient und zuverlässig auf realen Maschinen läuft."
  - IEEE /IEE 83/: "... is the systematic approach to the development, operation, maintenance, and retirement of software."
  - Hesse et al. /He 84/: "Softwaretechnik ist Fachgebiet der Informatik, das sich mit der Bereitstellung und systematischer Verwendung von Methoden und Werkzeugen für die Herstellung und Anwendung von Software beschäftigt".
  - viele andere Definitionen verbinden Softwaretechnik mit Methoden und Werkzeugen

- Thesen zur Softwaretechnik
  - Herstellung großer Software unterscheidet sich qualitativ von der kleiner Software
  - Probleme erst dann, wenn
    - viele Personen für Entwicklung/Weiterentwicklung
    - mehr als eine Version des Programmsystems
  - Geregelte Zusammenarbeit in Programmierermannschaft
    - Planung/Überwachung/Verteilung der Arbeit
    - Lösung der Arbeitsschritte und ihres Zusammenspiels
    - Sicherung der Qualität
    - Dokumentation der Ergebnisse
    - Information über Schritte und Ergebnisse
  - Auf allen Ebenen: Bewältigung der Komplexität;
    - Modellierungsproblematik

Lebenszyklusmodelle (s.u.)

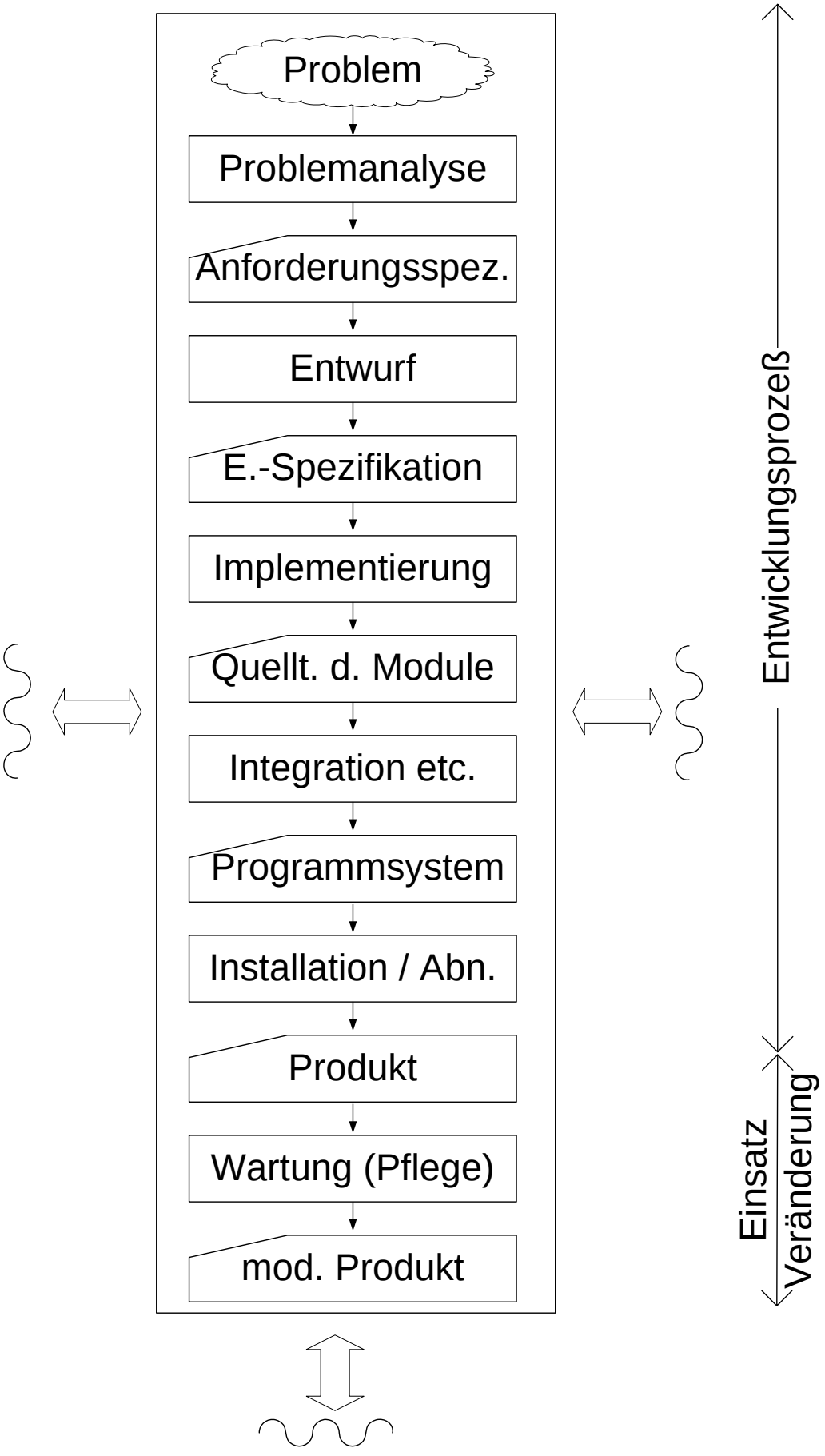
- Definition Lebenszyklus
  Strukturierung des Zeitraums
  ”Zyklus” wegen: Rückgriffen, neue Software ersetzt abgestorbene

- Lebenszyklusmodell
  spezielle Form: Phasenmodelle
  Einzelaktivitäten: Phasen
  Ergebnisse: Softwaredokumente
  Grund für Aufteilung:
  kleinere Komplexität
  Überprüfbarkeit / Abbruchpunkt

- Definition Softwaredokument

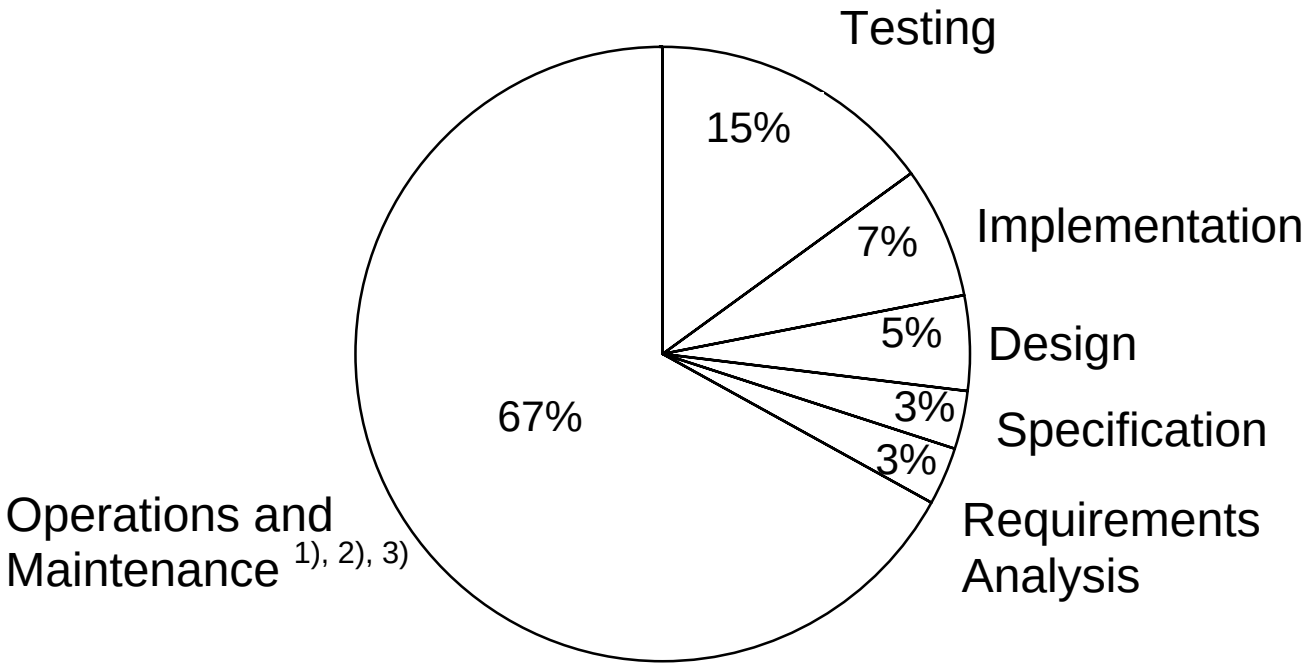
- Definition Anwendungssystem

Ein Phasenmodell



Die Realität

Verteilung des Aufwands auf Phasen



aus /Ze 79/

1) Je nach Art der Software und ihrer Lebensdauer schwanken Entwicklungskosten und Wartungskosten zwischen 40:60 und 10:90.
2) Nach /LS 80/ ist die Wartung etwa doppelt so teuer wie die Entwicklung.
3) Die Wartungskosten verteilen sich auf verschiedene Anteile, s.u.

Korrektheit

- Wahrscheinlichkeit der Korrektheit eines Programmsystems nach Dijkstra

  p Wahrscheinlichkeit, dass einzelne Komponente korrekt

  $P=p^N$ in etwa für System aus N Komponenten

  (Die Lücken werden in der Vorlesung gefüllt)

$\Rightarrow$ kein großes Programmsystem ist korrekt!

- Was heißt korrekt?

Schlechte Beispiele

- Ariane V - Rakete

  (private Mitteilung von Prof. Goos, Karlsruhe)

Die erste Ariane 5 explodierte 37 s nach dem Start am 4.6.96. Zitate nach dem offiziellen Untersuchungsbericht der ESA vom 19.7.96. (Man beachte die kurze Zeitspanne von nur ca. 4 Wochen zwischen Einsetzen der Untersuchungskommission und Bericht!)

Hauptgrund: Das (doppelt vorhandene) Trägheitsreferenzsystem (inertial reference system) SRI 2 sandte ein Testmuster und eine Fehlermeldung an den Bordrechner OBC, die von diesem als wichtig eingestuft wurde, obwohl sie irrelevant war. Die Software hatte auf Ariane 4 klaglos mehr als 500 mal funktioniert. Allerdings waren die SRIs für die Ariane 5 immer nur als schwarze Kästen und nicht in Zusammenarbeit mit dem OBC getestet worden, weil die Simulation der Geschwindigkeiten und Beschleunigungen für das Gesamtsystem zu aufwendig erschien. Da das SRI 1 die identische Software benutzte, produzierte es denselben Fehler. Ingenieure können sich immer noch nicht vorstellen, daß Systeme, die Software enthalten, nicht nur zufällige Fehler wegen Abnutzung, Materialermüdung usw. produzieren, sondern systematische Fehler. Die letztere Kritik steht wörtlich im Untersuchungsbericht!

Der Fehler im einzelnen:

1. Bei der Konversion einer 64-Bit Gleitpunktzahl in eine ganze Zahl von 16 Bit ergab sich ein Überlauf. Die Konversionsroutine setzte daraufhin ein beliebiges Muster ein. Der gesamt Code ist in Ada geschrieben und produzierte eine Ausnahme.

2. Der Fehler betraf eine Variable, die die Horizontalgeschwindigkeit maß. Diese ist beim Start einer Ariane 5 größer als bei einer Ariane 4; daher war der Fehler vorher nie aufgetreten.

3. Die Konversion war nicht durch lokale Ausnahmebehandlungen abgesichert wie vergleichbare Konversionen im gleichen Codestück. Grund: Um das Echtzeitsystem nicht zu überlasten, sondern mit max. 80% Last zu fahren, hatte man im relevanten Codestück bei 3 von 7 Variablen auf die Ausnahmebehandlung verzichtet. Ferner hatte eine Analyse (laut Dokumentation) gezeigt, daß bei den 3 Variablen entweder wegen physikalischer Grenzen oder im Hinblick auf den sehr großen Sicherheitsabstand Ausnahmen nicht auftreten konnten; offensichtlich eine fehlerhafte Aussage.

4. Der Fehler trat in einem Codestück auf, das nur der Grundeinstellung des noch am Boden befindlichen Trägheitssystems dient. Sobald die Rakete abgehoben hat, sind diese Daten bedeutungslos.

5. Die Daten werden trotzdem noch für 50 s nach dem Start des Flugmodus, d.h. 40 s nach dem Feuerbefehl berechnet, um bei einem eventuellen Startabbruch schnell zurücksetzen zu können. Das war eine Anforderung bei Ariane 4, die aber für Ariane 5 unzutreffend ist, da hier ein Startabbruch nur bis spätestens 3 s vor dem Feuerbefehl möglich ist. Der Fehler trat 36 s nach dem Feuerbefehl auf.

6. Da die Ausnahme ist nicht lokal abgefangen wurde, führte sie zum Absturz des Trägheitssystems und wurde dem OBC als schwerwiegender Fehler gemeldet. Der OBC konnte also den an sich irrelevanten Fehler nicht ignorieren, sondern versuchte auf das andere Trägheitssystem SRI 1 umzuschalten. Dieses war aber wegen identischer Software ebenfalls abgestürzt. Danach wurde auf den zweiten Bordrechner umgeschaltet. Für diesen war aber ebenfalls kein korrekt arbeitendes Trägheitssystem verfügbar, die die Spezifikation verlangt, daß ein fehlerhaft arbeitendes Trägheitssystem abgeschaltet werden soll. Somit war die Rakete schließlich ohne Trägheitssystem.

7. Die Rakete kam um mehr als 20 Grad vom Kurs ab, Teile wurden abgerissen und der Selbstzerstörungsmechanismus reagierte.

Resümee:

Einordnung

Software: Für wen?

- spezieller Auftrag von *externem Auftraggeber* für ein Softwarehaus (z.B. Reisebüro-Buchungssystem START)
- *für Systemhersteller* (z.B. technisches System Auto, Flugzeug) oder eingebettetes, produktionstechnisches System (z.B. Walzstraßensteuerung)
- Gruppe in Firma entwickelt spezielle Software für *andere Gruppen* derselben Firma (z.B. Datenhaltungssystem)
- Softwarehaus entwickelt allgemeine Anwendungssoftware *für* den *Markt* (z.B. Buchhaltungssoftware)
- Softwareentwicklung für *persönlichen Gebrauch*

Software welchen Charakters?

- komplettes Anwendungssystem: für Endnutzer
  - Einzellösung (START)
  - Massenlösung (Buchhaltungssystem)
- Softwaresystem als Teil einer umfassenden Lösung eines technischen Systems
- Allgemeine Komponente eines Softwaresystems für Anwendungssystemersteller
  - Komponente für ein Anwendungssystem
  - Komponenten für viele Anwendungssysteme
- Werkzeuge für Softwareerstellung, für Softwareingenieur
  - allgemeine Hilfsmittel (Compiler, Software- Entwicklungsumgebungen)
  - spezielle Hilfsmittel für bestimmte Anwendungsgebiete (z.B. Jackson-Tools) für Manager, Dokumentator
- Basismaschinerie
  - Betriebssystem, Dateiverwaltungssystem
  - Netzschicht für verteilte Anwendungen

Zum Stand der Softwaretechnik (aus /Nag 90/)

Stand der Softwareerstellung: auch heute kaum wissenschaftlich durchdrungen, im folgenden einige Gründe, Vergleich mit anderen Ingenieurwissenschaften

- Hektik der Entwicklung
- Breite der Anwendungen
- Probleme der Software werden nicht verstanden
- Immaterialität von Software:
  - Erfahrungen mit Software können nur durch Beschreibungen derselben oder durch das Ausprobieren eines Produkts gewonnen werden. Software kann man *nicht* durch *Sinneswahrnehmung* erfahren.
  - Software hat eine unglaubliche *"Wandelbarkeit"*. Dies ist ein Vorteil, aber auch ein gravierender Nachteil. So manches Programmsystem ist in der Wartungsphase von einem Küstenmotorschiff zu einem Mammuttanker geworden.
  - Unterschiede der Realisierung kann man bei Software kaum feststellen, man muß hierzu schon ein Fachmann des Anwendungsgebietes sein. *Qualitätsmerkmale* sind also *schwer überprüfbar*. Der Software ist von außen nicht anzusehen, ob sie solide entwickelt oder zusammengeschustert worden ist.
  - Mit der Schwierigkeit, eine Realisierung einzuschätzen, ist die Schwierigkeit verbunden, den *Wert* von Software zu *bestimmen*.

  - *Software altert nicht* und unterliegt keinem Verschleiß. Dadurch ist man nicht gezwungen, Software, die dem Stand der Technik nicht mehr entspricht, wegzuwerfen. Andererseits verhindert langandauernde Wartung, dass ein Softwaresystem funktionsfähig bleibt.
  - Die *Zerstörung* von Software *durch Wartung* ist von außen *schwer feststellbar*. Dadurch wird auch wenig Nachdruck auf Wartung gelegt, die die Struktur des Produkts erhält. Dies ist einer der wesentlichen Gründe dafür, dass ein großer Teil der Softwareentwickler mittlerweile mit Wartung beschäftigt ist.
  - Softwaretechniker gehen ausschließlich mit geistigen Produkten um, d.h. der Kernpunkt der Softwareentwicklung liegt in der Modellierungsproblematik. Wegen der Breite der Anwendungen, für die Software entwickelt wird, ist dies die *Modellierungs-* oder *Strukturierungsproblematik schlechthin*. Für diese wird man in keiner Wissenschaft tiefschürfende Ergebnisse finden.
- Mangelndes Übereinkommen
- Schwierige Randbedingungen

Resümee:

- Hektik verhindert tiefes Durchdringen
- Breite macht allgemeine Aussagen schwer
- Software ist *spezielle Materie*
- Softwareprobleme auch heute noch unverstanden
- Rad täglich neu erfunden
- Randbedingungen verstellen Blick auf das Wesentliche

Allgemeine und aktuelle Probleme der Softwaretechnik

- Akzeptanz
  (Viele SW-Systeme kommen nicht zum Einsatz)
- Qualität (insbesondere Korrektheit, s.u.) von SW-Systemen
  (s. Beispiele)
- Wartungsprobleme bei SW-Systemen
  (zu wenig Kapazität für Neuentwicklungen)
- Integration von SW-Systemen
  (bisher viele Insellösungen, die in Zukunft zu integrieren sind)
- Verteilung von SW-Systemen
  (die Welt verändert sich dramatisch, s. Internet)
- Wiederverwendung für SW-Systeme
  (bisher kaum praktiziert)
- "Methoden"-Unterstützung bei der SW-Entwicklung
  (mangelnde Präzision von Sprachen, Klärung der Zusammenhänge, Prozessverständnis)
- Werkzeugunterstützung bei der Entwicklung
  (nur rudimentär vorhanden)
- Schätzverfahren in SW-Projekten
  (viele Projekte und Firmen scheitern)
- Klare Terminologie
  (jedes Buch anders, stets neue Begriffe für alte)
- Mangelnde Spezialisierung
  (wir sind allgemeine Problemlöser)
- Mangelnde Produktivität
  (Wer leistet sich in Zukunft die teure SW-Entwicklung?)

Hot Topics

- Softwareentwicklungs-Prozesse
- OO-Techniken, insb. Vereinheitlichung
- OO-Entwurf, Design Patterns
- Uniforme Ansätze OO-Analyse/Design/Programmierung
- Verteilte Softwaresysteme, Plattformen, Komponenten, selbst-organisierende Agenten

Was ist Softwaretechnik?

- Ingenieurwissenschaft
- Kunst
- Zufall
- Strukturwissenschaft

Vision

Spezialisierung in der Softwaretechnik

- Anwendungsbereiche:
  - betriebswirtschaftliche Software (Buchungssystem, Lohnprogramm)
  - Steuerung technischer Systeme (Steuerung einer chemischen Fabrik, Fertigungsautomatisierung)
  - eingebettete Software, Hard-Software-Systeme (Motorsteuerung, Steuerung des Fahrverhaltens eines Automobils)
  - mathematisch-naturwissenschaftliche, technische Berechnungen (Experimentauswertung, FEM)
  - Kommunikations-Software (Dokumentaustausch, Nebenstellenanlagensteuerung)
  - Systemsoftware (SEU, Datenbanksystem)
  - Dokumentbearbeitung (Zeitungsbearbeitung, Multimediaanwendung)
  - Simulation/Vorhersage (Wettervorhersage, Simulation des Verhaltens eines technischen Systems)
  - ...

$\Rightarrow$ Anwendungstechnik

- Strukturklassen von Systemen:
  - Transformationssysteme, interaktive Systeme, reaktive Systeme
  - sequentielle Systeme, nebenläufige Systeme, parallele Systeme
  - festverdrahtete, tabellengesteuerte, generierte, regelbasierte Systeme
  - gebundene, dynamisch geladene, verteilte Systeme
  - vollständige Systeme, Systeme in Hard- oder Softwaresystemen
  - Basissoftware, Anwendungssoftware, Entwicklungssoftware
  - Plattform: Mikrorechner, PC, homogene/heterogene Netze, Parallelrechner, technisches System

$\Rightarrow$ Strukturklassentechnik

- Projektarten
  - Neuentwicklung
  - Erweiterung, Modifikation
  - Reengineering, Reverse Engineering
  - kleines Projekt vor Ort, ..., verteiltes Großprojekt
  - Projektfamilie (Wiederverwendung:) Produkt, ..., Prozess
    Rahmenwerksprojekt, Generatorprojekt

  $\Rightarrow$ Projekttechnik

- Zugrundeliegende Gedankenwelt/Realisierungssicht
  z.B. Programmiersprachen:
  - Assembler, C
  - FORTRAN, COBOL, Ada
  - reine OO-Sprachen, regelbasierte Sprachen
  - Interpreter-orientierte Sprachen (KI):
    Lisp, Prolog
  - Funktional-/mathematische Programmiersprachen:
    Miranda/Haskell

  analog Konzepte/Sprachen für andere Arbeitsbereiche,
  analog Basisschicht, Plattform

  $\Rightarrow$ Modellierungs-/Realisierungstechnik (allgemeine Technik, teilweise im Grundstudium kennengelernt)
- Weitere Dimensionen:
  - SW: für wen? (s.o.)
  - Charakter der SW (s.o.)

- Resümee:
  - Spezialisierung nötig in obigen Dimensionen
  - zur Zeit Informatiker allgemeiner Ingenieur oder Modellierer/Strukturierer
  - handwerkliche Fertigung nur durch Spezialisierung zu überwinden (Beispiel Compilerbau, SEU)
  - So etwas wie ”Stromgeneratoren I, ...” als Teilgebiet der E-Technik in der Informatik/Softwaretechnik
- dabei Technik: Konzepte, Sprachen, Methoden, Hilfskomponenten, Werkzeuge, Standards, Erfahrungen etc., die für Aufgabe nützlich sind und dabei spezialisiert nach obigen Dimensionen

Zukunft der Softwareentwicklung?

- Rahmenwerk, nur noch spezielle Bausteine
- Plattformen: verteilte Betriebs-, Datenbank-, Kommunikationssysteme etc.
- mechanische Erstellung, ..., automatische Generierung
- Systemerstellung durch Konfigurieren von vorgefertigten Komponenten
- Hilfsmittel der Softwareerstellung umfassender und semantischer
- Programmier- oder Realisierungsniveau höher als heute
- Spezialisierte Anwendungsumgebungen

$\Rightarrow$ Softwareentwicklung ist ein bekannter, verstandener, formalisierbarer, statisch festlegbarer Entwicklungsprozess

Fragen und Feststellung:

Fragen:

- Welche Software (→ Requirements Engineering): Validierung keineswegs trivial
- Produkteigenschaften: Adaptabilität, Portabilität, etc.
- Welche Projekte: Neuerstellung, Erweiterung, Reengineering, Teilsystementwicklung, Generalunternehmerschaft, wie erhalten wir Projekttechnik
- Auf welchen Ebenen Produkt, was ist Produkt
- Beherrschen der Prozesse, auf welchen Ebenen Prozesse, Wechselwirkung mit Produkt
- Wiederverwendung und Abkehr von Einzelfallerstellung (Beispiele Compiler, Softwareentwicklungs-Umgebungen)
- Was ist Qualität, auf welchen Ebenen tritt sie auf
- Was ist Software, worin unterscheidet sich Software von anderen Produkten
- Welche Anwendungsbereiche, wie erwerben wir Anwendungstechnik
- Welche Strukturklassen von Softwaresystemen, wie kommt man zu Strukturtechnik
- Was ist ein umfassendes Modell für Software, Softwareerstellung/-wartung und alle wichtigen Aspekte: Softwareprojekt oder Projektfamilien
- Welche Unterstützung durch Werkzeuge ist denkbar, welche gibt es (→ SEU)
- Was sind geeignete Konzepte, Sprachen, Methoden, Standards für Teilbereiche der Softwaretechnik

Feststellung:

- Unter Softwaretechnik verstehen verschiedene Menschen unterschiedliches
- Saubere Strukturierung als Wissenschaftsdisziplin nicht verstanden: Teilbereiche und ihr Zusammenhang nicht klar
- Bevor wir Klarheit erzielen, brechen wir zu neuen Ufern auf
- Über viele praktischen Probleme, z.B. Re-Engineering, kann wenig gesagt werden oder wird wenig gesagt
- ST-Bücher sagen über die in der Praxis am häufigsten vorkommenden Prozesse (Wartung, Re-Engineering, Reverse Engineering) nichts oder wenig aus
- Keine standardisierte Gedanken- und Sprachwelt vorhanden: Vielerlei Sprachen/Konzepte/Ansätze für RE, Entwurf, Kodierung, Projektorganisation, Qualitätssicherung, Dokumentation

Aufgaben zu Kap. 1:

1. Zur Graphik HW/SW-Kosten:
   a) Welche gegenläufigen Entwicklungen, die zur Entschärfung der SW- bzw. speziell der SW-Wartungskosten hat es seit ’76 gegeben?
   b) Welche neuen Herausforderungen sind seitdem jedoch eingetreten?
   c) Wie stehen Sie zu der in der Vorlesung gemachten Aussage, dass die HW/SW-Relation nach wie vor so ist, wie ’76 geschätzt?
2. Die Korrektheitsformel von Dijkstra ist vereinfachend und falsch. Welche Simplifikationen sind in ihr enthalten? Wie steht der dort angesprochene Korrektheitsbegriff zu einem praxisnahen? Wann treten neue Fehler eines längere Zeit im Einsatz befindlichen SW-Systems gehäuft auf?
3. In allen Ingenieurwissenschaften ist der Trend zu Software seit langem ungebrochen (Mechanik durch Software ersetzt, Regelung durch Software etc.). Dadurch erben die Ingenieurwissenschaften die Probleme der Softwaretechnik. Darüber hinaus sind Ingenieure in Bezug auf SW-Erstellung schlechter ausgebildet als Informatiker. Trotzdem sind in technischen Anwendungsbereichen nur verhältnismäßig wenig Informatiker zu finden. Welche Probleme ergeben sich daraus? Welche Schlussfolgerungen für eine Verbesserung müssten gezogen werden?
4. Stellen Sie sich bitte vor, Sie seien ein Ingenieur einer klassischen Disziplin, wie etwa des Automobilbaus. Versuchen Sie, die Probleme der Softwaretechnik (s. entsprechenden Abschnitt) in der Vorstellungs- und Begriffswelt dieser Disziplin auszudrücken.

5. Stellen Sie in Form einer Matrix Beispiele für Rückgriffe im Softwarelebenszyklus zusammen, mit den Phasen der Softwareerstellung in den Zeilen und Spalten. Jeder Eintrag der Matrix sollte mindestens ein Beispiel enthalten. Jede Zeile steht für die Phase, von der der Rückgriff ausgeht und jede Spalte für die Phase, auf die zurückgegriffen wird.
6. Versuchen Sie, ein Beispiel einer Aufgabe der Softwareerstellung zu finden, bei der alle Arbeitsbereiche angesprochen sind, und versuchen Sie, anhand dieses Beispiels, die Vernetzung der Arbeitsbereiche aufzuzeigen. Welche Unterschiede sehen Sie zwischen einer klassischen Ingenieurwissenschaft und der Informatik/Softwaretechnik?
7. In der Vorlesung wurde die Vision einer zukünftigen Softwaretechnik durch entsprechende Spezialisierung und Ansammlung spezifischen Wissens ausgebreitet. Stellen Sie die Situation eines üblichen Softwareprozesses (z.B. Wartungsprozess mit teilweiser Restrukturierung und Erweiterung) dem eines technikgetriebenen Wiederverwendungsprozesses gegenüber (z.B. Compiler- oder SW-Entwicklungsprozess). Charakterisieren Sie dabei das entsprechende Produkt, den Prozess und das Projekt.

Literatur zu Kap. 1:

/Ari 96/ Ariana 5 Flight 501 Failure, Report by the Inquiry Board

/Boe 76/ B.W. Boehm: Software Engineering, IEEE Trans. on Computers C-25, 1226-1241, 1076

/Boe 87/ B.W. Boehm: Improving Software Productivity, IEEE Computer 20, 9, 43-58, 1987

/BR 69 J.N. Buxton, B. Randell (Eds.): Software Engineering Techniques, Report on a Conference, Rome 1969, Brussels, Nato Scientific Affairs Division, 1969

/He 84/ W. Hesse et al.: Ein Begriffssystem für die Softwaretechnik - Vorschläge für Terminologie, Informatik-Spektrum 7, 4, 200-213, 1984

/IEE 83/ IEEE Standard Glossar of Software Engineering Terminology, IEEE Standard 729, IEEE Soc. Press, 1983

/LS 80/ B. Lientz, E. Swanson: Software Maintenance Management, Addison-Wesley, 1980

/Nag 90/ M. Nagl: Softwaretechnik: Methodisches Programmieren im Großen, Springer, 1990

/Nag 96/ M. Nagl (Ed.): Building Tighly-Integrated Software Development Environments: The IPSEN Approach, LNCS 1170, 1996, Kap. 1

/NR 68/ P. Naur, B. Randell (Eds.): Software Engineering: Report on a Conference, Garmisch 1968, Brussels, Nato Scientific Affairs Division, 1968

/Zel 79/ M. Zelkowitz et al.: Principles of Software Engineering and Design, Prentice Hall, 1979