

## Software Engineering

**Autor:** **Dipl.-Math.  
E. Engelhardt**

**Stand:** **15. Oktober 2001**

### Was ist Informatik?



- ◆ Einfache Definition:
  - Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen, insbesondere der automatischen Verarbeitung mit Hilfe von Digitalrechnern.
- ◆ Informatik ist weder Natur- noch Technikwissenschaft, sondern Strukturwissenschaft (wie Mathematik), d.h.:
  - Informatik behandelt und untersucht vom Menschen geschaffene formale Strukturen (Datenstrukturen, Sprachstrukturen, Systemstrukturen)
  - Informatik ist eine Disziplin der Informationswissenschaften
- ◆ Zentrales Thema der Informatik ist die Formulierung und Realisierung von Algorithmen

- ◆ Informationstheorie
  - Verarbeitung und Übertragung von Informationen; Codierung, Störungen, Korrekturverfahren
- ◆ Kybernetik
  - Wissenschaft von der Steuerung, Regelung und Informationsübertragung in technischen und biologischen Systemen (Rückkopplung, Lernen, adaptive Verfahren)
- ◆ Systemtheorie
  - untersucht die Rolle der Information unter Berücksichtigung gesellschaftlicher, soziologischer und politischer Aspekte
- ◆ Kognitionswissenschaften
  - Wissenschaft vom „Wissen“, Berücksichtigung psychologischer Aspekte
- ◆ Linguistik
  - Sprachwissenschaften

- ◆ Theoretische Informatik
  - Formale Sprachen, Automatentheorie, Theorie der Berechenbarkeit, Komplexitätstheorie, Algorithmenanalyse, Theorie der Programmierung
- ◆ Technische Informatik
  - Hardwarekomponenten, Schaltnetzwerke, Schnittstellentechnik, Computerkonstruktion, Steuerungen, Mikroprogrammierung
- ◆ Praktische Informatik
  - Algorithmen und Datenstrukturen, Softwaretechnik, Programmiersprachen und Compiler, Programmiermethoden, Betriebssysteme
- ◆ Angewandte Informatik
  - Informationssysteme, Prozess-Steuerung, Künstliche Intelligenz, Computergrafik, Simulation und Modellierung, spezifische Gebiete in Natur- und Ingenieurwissenschaften

- ◆ Software:
  - Gesamtheit von Computerprogrammen und Dokumentation
  - System-Dokumentation, Benutzer-Dokumentation
- ◆ Systementwicklung:
  - alle Aspekte computerbasierter Systementwicklung (Hardware-, Software- und Verfahrensentwicklung)
- ◆ Software Engineering:
  - Teil der Systementwicklung
  - praktische Probleme der Software-Herstellung
- ◆ Softwareprozess:
  - Tätigkeiten zur Entwicklung oder Weiterentwicklung von Software

- ◆ Vorgehensmodell:
  - Vereinfachte Darstellung eines Softwareprozesses aus einer bestimmten Perspektive gesehen
- ◆ Methoden des Software Engineering:
  - Systemmodelle
  - Notationen (UML, Struktogramme, ...)
  - Regeln
  - Vorgehensweisen

- ◆ Begriff entstand auf einer Konferenz (1968) zum Thema „Softwarekrise“ (Bauer)
  - 3. Computer-Generation (1960 – 1970)
  - Einführung von integrierten Schaltkreisen (SSI = small scale integration) mit bis 1000 Transistorfunktionen
  - ermöglichte Serienfertigung von Großrechnern für Wissenschaft und Wirtschaft
  - Einführung von Multi-Tasking und Magnetplatten
  - machte große komplexe Softwaresysteme möglich
  - Software war um Vielfaches größer und komplexer als bisher
  - Große Projekte waren oft um Jahre im Rückstand
  - Kosten „explodierten“
  - Kosten für Hardware sanken

- ◆ Definition des Software Engineering nach IEEE (Institute of Electrical and Electronics Engineers)
  - „(1) The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
  - (2) The study of approaches as in (1).“
- ◆ Software Engineering befasst sich mit allen Aspekten der Softwareherstellung

- ◆ Organisierte Softwareproduktion mit ingenieurtechnischen Mitteln (wie z.B. im Maschinenbau)
  - Theorie
  - Modelle (Wasserfall-Modell, V-Modell, Spiral-Modell, ...)
  - Werkzeuge (CASE-Tools)
  - Qualitätssicherung (Testen)
  - Nachnutzung von Software (Modularität, Allgemeinheit)
  - Konfigurationsmanagement
  - Projektmanagement
  
- ◆ Herstellung großer Programme
  - meist mehrere (viele) Programmierer
  - meist für viele Benutzer (Kunden)
  - Entwicklung über längeren Zeitraum

## Ziel des Software Engineering

- ◆ Versuch der Lösung folgender Probleme:
  - Zeitrahmen für die Entwicklung wird überschritten
  - Budget wird überschritten
  - Produkt trifft nicht die Anforderungen des Auftraggebers
  - Produkt ist fehleranfällig
  - Wartungskosten für die Software sind zu hoch
  - Wartung der Software ist zu komplex und schwierig
  - 70% der Entwickler sind mit Systemwartung beschäftigt
  - ...

- ◆ **Prinzipien:**
  - Allgemeine und abstrakte Aussagen über wünschbare Eigenschaften von Software und ihren Entwicklungsprozessen
- ◆ **Methoden:**
  - Generelle Richtlinien zur Prozessorganisation
- ◆ **Techniken:**
  - Spezielle Verfahren zur Lösung von Teilproblemen im Entwicklungsprozess
- ◆ **Werkzeuge:**
  - Implementierungen bestimmter Techniken oder Methoden

Prinzipien und Werkzeuge sollen das Erreichen von Qualität unterstützen

## Vier Grundprinzipien des SE (1/2)

- ◆ **1. Strenge und Formalität**
  - Lehrbücher der Mathematik sind streng, aber nicht formal!
  - Strenge Verfahren in der Softwareentwicklung: Entity-Relationship-Diagramme. Schrittweise Verfeinerung
  - Formale Verfahren in der Softwareentwicklung: Algebraische Spezifikation.
- ◆ **2. Separation der Interessen => Modularität („Teile und Herrsche“)**
  - Verschiedene Aspekte wie Funktionalität, Qualitätsmerkmale, Hard- und Softwareumgebung, Teamorganisation, Kostenkontrolle . . . werden (soweit möglich) isoliert betrachtet. Dies ist die einzige Möglichkeit, die Komplexität der Entwicklung in den Griff zu bekommen!
  - Trennung verschiedener Sichten: Datenfluss -- Kontrollfluss ...

- ◆ **3. Abstraktion => Allgemeinheit**
  - Ignorieren von irrelevanten Details
  - höhere Programmiersprachen
  - Definition geeigneter Funktionsbausteine
  - Das Finden geeigneter Abstraktionen ist der Kern der Informatik
  
- ◆ **4. Evolutionsfähigkeit => Inkrementalität**
  - In Software ist nichts beständiger als der Wandel Bereits bei der Erstellung muss der zukünftigen Software-Evolution Rechnung getragen werden.
  - Inkrementalität = Vorgehen in kleinen Schritten
  - Teilsysteme mit eingeschränkter Funktionalität können frühzeitig ausgeliefert werden
  - Hohle Schnittstellen zu bauen ist gängige Praxis!

- ◆ **Software ist immaterielles Produkt**
  - Dokumentation stimmt oft nicht mit dem Programm überein (bei technischen Produkten aber oft auch nicht)
  - Kopien sind identisch
  
- ◆ **Kein Verschleiß**
  - außer moralischem Verschleiß (Veralterung)
  - kann beliebig oft ablaufen
  
- ◆ **Software wird nicht durch physikalische Gesetze begrenzt**
  - künstliches Produkt des menschlichen Geistes
  - großer Gestaltungsspielraum

- ◆ Software ist leicht und schnell änderbar
  - keine neuen Werkzeuge nötig
  - wird durch gute Struktur und Modularität unterstützt
  
- ◆ Für Software gibt es keine Ersatzteile
  - evtl. doch: z.B. anderes Sortierverfahren ...
  
- ◆ Software ist schwer zu vermessen
  - Qualität ist schwer definierbar und quantifizierbar
  
- ◆ Software ist unstetig
  - ein falsches Bit kann Software unbrauchbar machen

- ◆ Korrektheit
- ◆ Benutzerfreundlichkeit
  
- ◆ Effizienz
- ◆ Wartbarkeit
  
- ◆ Wiederverwendbarkeit
- ◆ Kompatibilität
  
- ◆ Zuverlässigkeit
  
- ◆ Robustheit
  
- ◆ Portierbarkeit

- ◆ Anforderungen des Marktes
    - Funktionstreue
    - Qualitätstreue
    - Termintreue
    - Kostentreue
  
  - ◆ Erschwernisse während der Software-Erstellung
    - sich ändernde Produkthanforderungen
    - andere Hardware
    - veränderte Systemkomponenten
    - sich ändernde Werkzeuge (CASE-Tools)
    - hohe Portabilitätsforderungen (Software-Produkte haben längere Lebensdauer als Hardware)
- ==> Software-Krise

- ◆ Grundlegenden Prozessaktivitäten (nach Ian Sommerville)
  - Spezifikation
  - Softwareentwicklung
  - Validation
  - Evolution
  
- ◆ Bezogen auf Software ist der Entwicklungsprozess der Software-Lebenszyklus
  - Wasserfall-Modell wurde von ingenieurtechnischen Entwicklungen abgeleitet
  - Für evolutionäre Softwareentwicklung ist das Spiral-Modell (nach Böhm) geeignet, dabei gibt es:
    - a) Explorative Softwareentwicklung und
    - b) Prototyping

Quelle unbekannt



Produktplan

Produktdefinition

Produktentwurf



Produktimplementierung

Produktabnahme

Wunsch des Benutzers

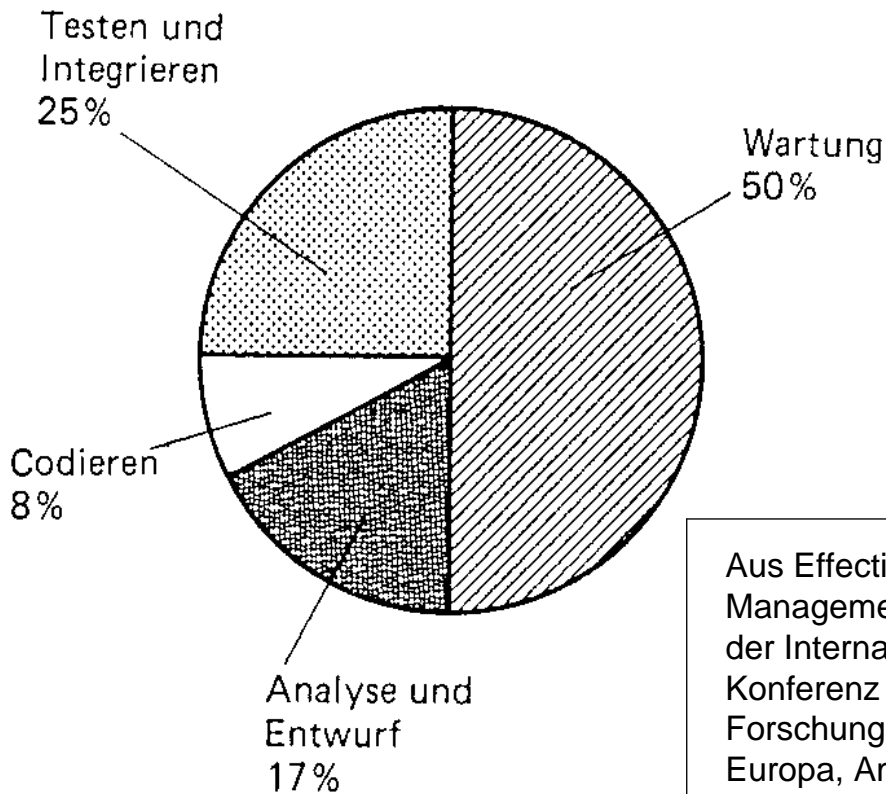
## Lebenszyklus

### ◆ Phasen

- Analyse und Design
- Implementierung
- Test (Komponententest, Systemtest)
- Anpassung, Fehlerbeseitigung
- Nutzung (Wartung, Pflege)

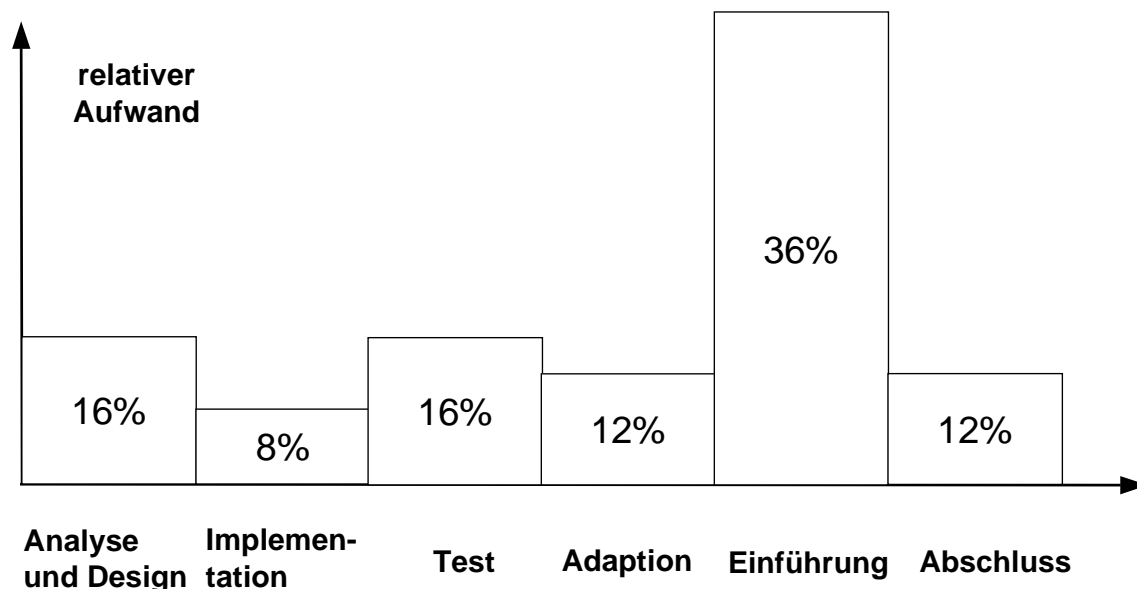
### ◆ verschiedene Phasenmodelle

- Wasserfall-Modell mit und ohne Iteration
- Spiralmodell (evolutionäre Entwicklung, Prototyping)
- V-Modell

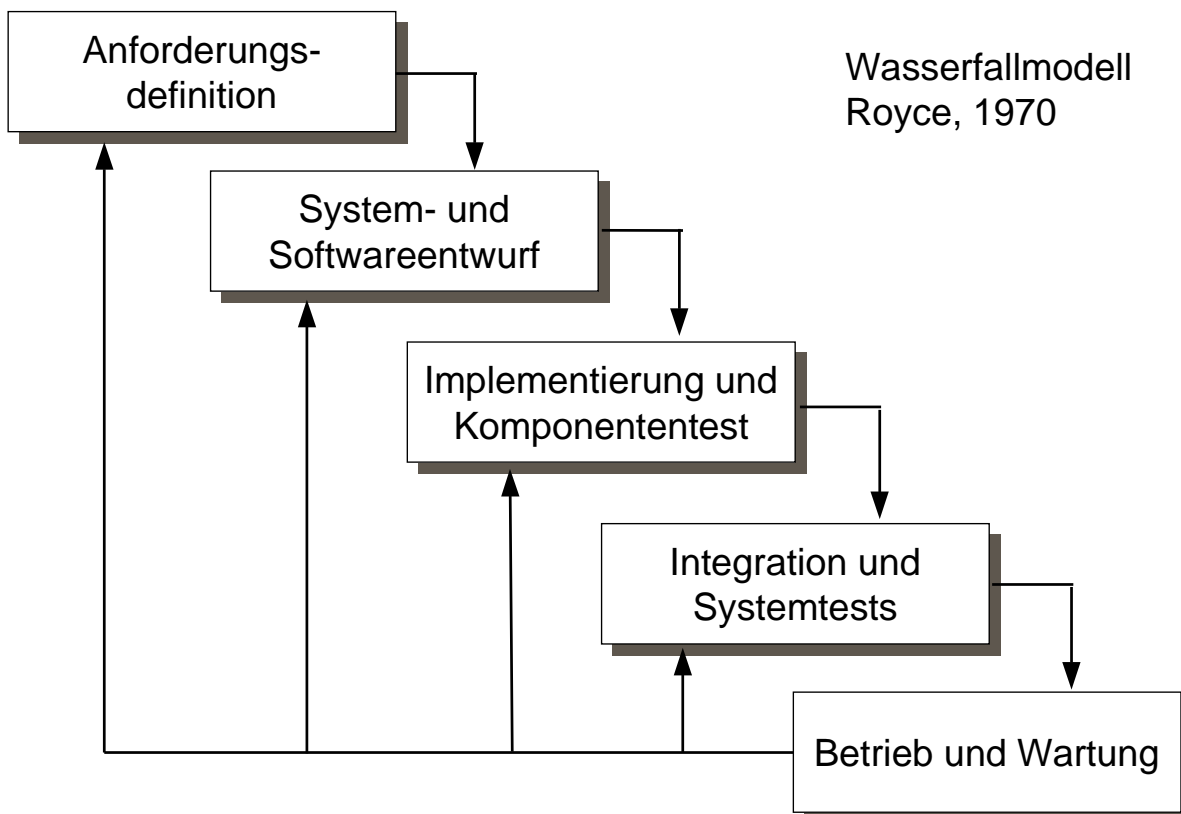
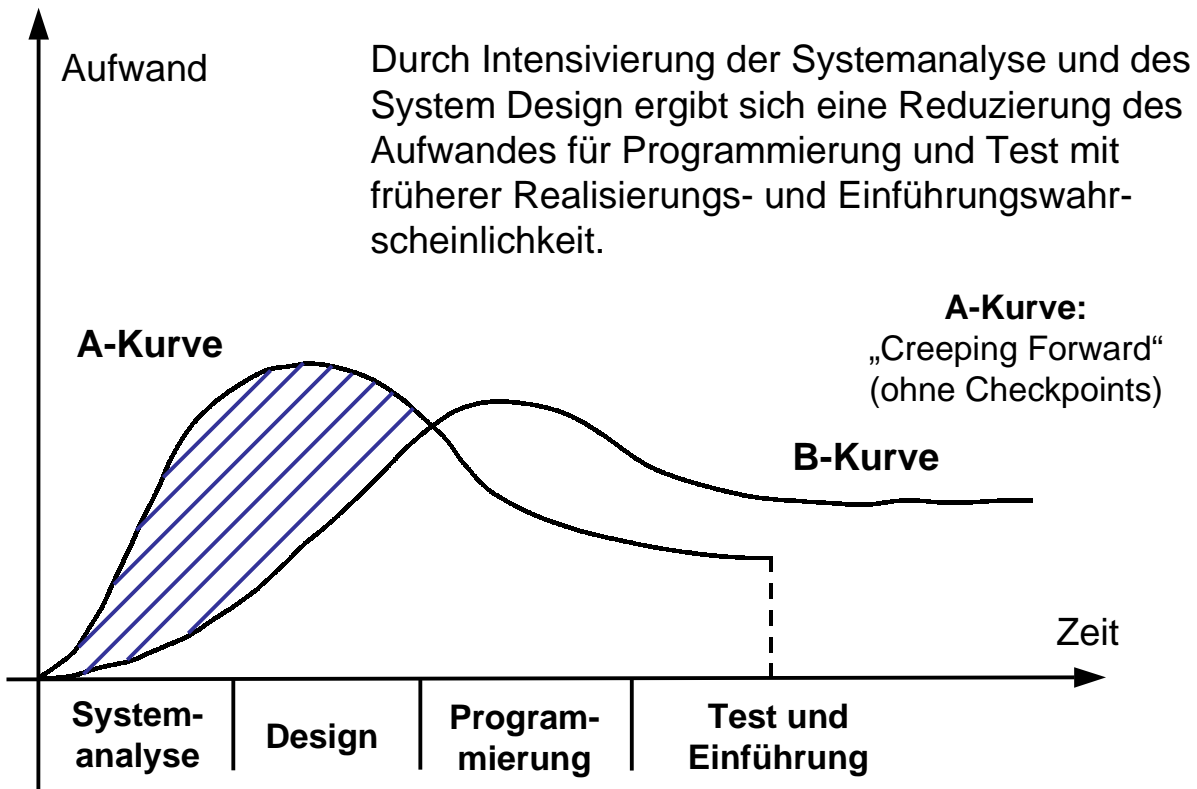


Aus Effective Software Management Proceedings der Internationalen Konferenz des Diebold Forschungsprogramms Europa, Amsterdam, 1989

Eine ähnliche Verteilung des Aufwands über verschiedene Phasen eines Projektes hinweg ist auch nach einer Untersuchung von Fairley erkennbar.



Distribution of effort in the software life cycle (Boehm, Datamation 1973)

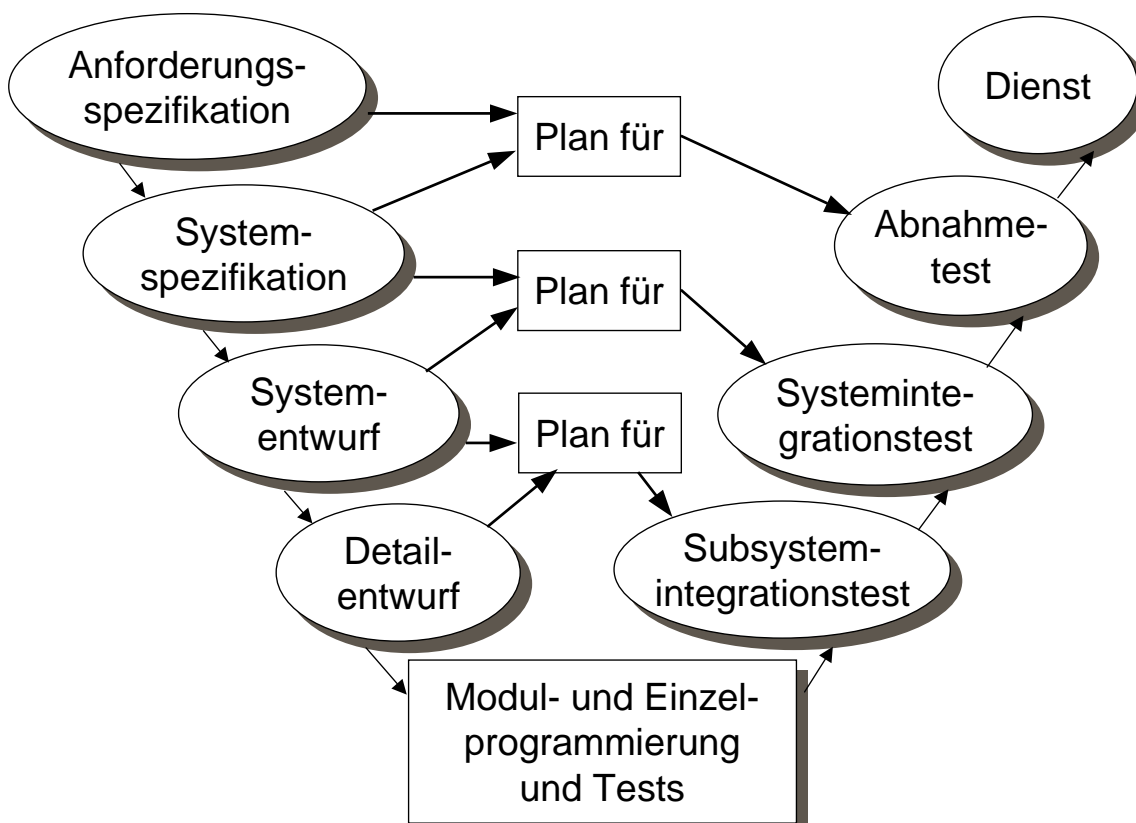


## ◆ Vorteile

- einfaches Modell entstanden aus anderen ingenieurtechnischen Entwicklungen

## ◆ Nachteile

- kann Vorgänge zwischen den Phasen unzureichend darstellen
- vernachlässigt Projekt- und Qualitätsmanagement
- keine Konfigurationskontrolle



## ◆ Beschreibung

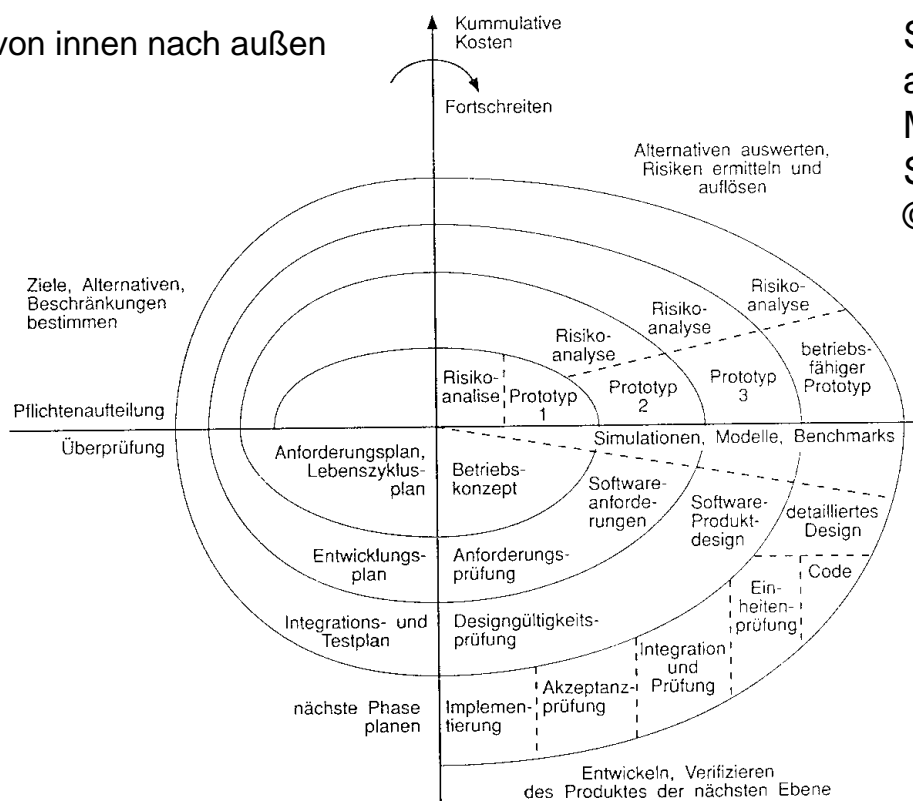
- im linken Abschnitt werden Details vertieft
- im rechten Abschnitt wird die Software zunehmend aus Modulen zusammen gesetzt
- phasenübergreifende Aktivitäten sind eingeschlossen

## ◆ Nachteile

- Iterationen werden vernachlässigt

# Spiralmodell nach Barry Boehm

von innen nach außen



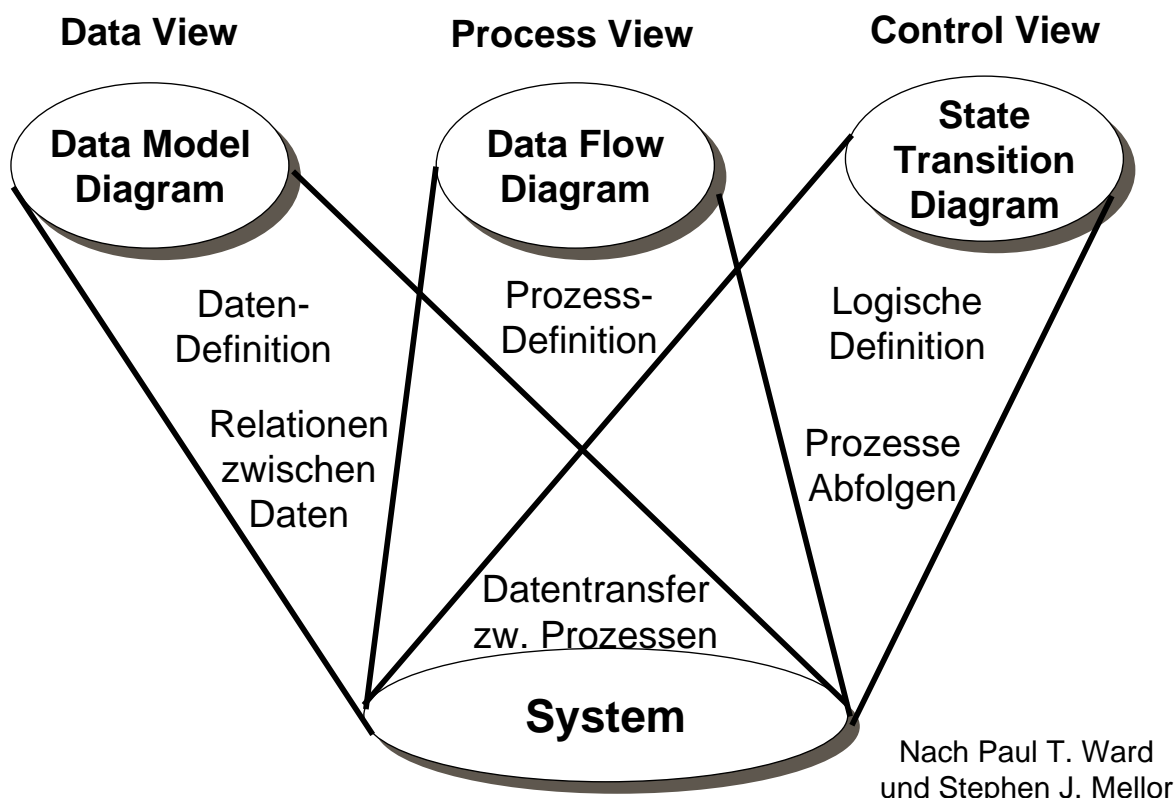
Spiralmodell  
aus Computer,  
Mai 1988,  
Seite 62,  
© 1988 IEEE

- ◆ Aufgaben der Qualitätssicherung in DIN ISO 9001
- ◆ Einteilung der Qualitätssicherungsmaßnahmen (QSM) nach E. Wallmüller
  - planerisch administrative
  - konstruktive (vorbeugende Maßnahmen)
  - psychologisch orientierte (Gestaltung der Arbeitsatmosphäre)
- ◆ Analytische QSM
  - statische analytische QSM
  - verifizierende Verfahren (symbolischer Test, Verifikation)
  - dynamische QSM
- ◆ Validation: das richtige Produkt wird entwickelt
- ◆ Verifikation: Produkt leistet das Gewünschte

- ◆ Symbolischer Test
  - Interpretation in künstlicher Umgebung
  - Stichprobencharakter
  - Fehler können nachgewiesen werden, aber nicht Abwesenheit von Fehlern (E. W. Dijkstra)
- ◆ Fehler
  - Korrektheit eines Programms: Abweichung der Implementation von der Spezifikation
  - Strukturelle Softwaretests: strukturelles Merkmal des Programmtextes
- ◆ Klassifizierung der Fehler
  - Datenreferenzfehler, Datendeklarationsfehler
  - Berechnungsfehler, Vergleichsfehler
  - Kontrollflussfehler, Schnittstellenfehler, E/A-Fehler

- ◆ **Klassifizierung der Testverfahren**
  - Musterarchitektur (GUI-Test, Client-Server-Test)
  - Komponentenstruktur (Unit-Test, Integrationstest)
  - Verfügbarkeit der Quellprogramme (Whitebox, Blackbox)
  - Testreferenz (funktionaler Test, zustandsbasierter Test)
  - Einbeziehung der Versionierung (Mutationen-Test)
  - Struktur der Eingabedaten (Bereichstests)
  - Fehlererwartung (Fehlerorientierte Testverfahren)
  - kein Merkmal (Zufallstest)
- ◆ **Teststrategien**
  - Top-down-Testung
  - Bottom-up-Testung
  - Thread-Testung
  - Stress-Testung (Belastungs-Tests)
  - Back-to-Back-Testung (verschiedene Versionen)

## Drei Sichten auf das System



## ◆ Hauptrichtungen

- Daten (Betrachtung der Ein- und Ausgabeflüsse)
- Prozesse und Verarbeitungsfunktionen
- Information Engineering

„Der Schwerpunkt liegt auf den Informationsanforderungen einer Organisation, der Modellierung dieser Informationen und dem Aufbau eines Informationssystems, das auf diesem Modell basiert.“ (James Martin)

## ◆ Strukturierte Analyse / Strukturiertes Design

- Datensicht: Data Model Diagram (Bachman Notation)
- Prozess-Sicht: Data Flow Diagram (Gane/Sarson Notation)
- Steuerungssicht: Finite State Machine

# Information Modeling

## ◆ Entity-Relationship-Attribute Model (E-R-A) nach Chen

- Entitäten: unterscheidbare Dinge der realen Welt
- Relation: Beziehung zwischen zwei oder mehreren Entitäten
- Attribut: kennzeichnen Entitäten (Eigenschaften)

## ◆ Merkmale

- Standard für Datenmodellierung
- keine Prozess-Sicht
- leicht verständlich
- gute Basis für objektorientierten Ansatz

- ◆ Jackson - Structured Design
  - Drei Elemente: Sequenz, Selection, Iteration  
Folge, Verzweigung, Schleife
  - Vorgehen in vier Schritten:
    1. Darstellung des Datenflusses
    2. Zusammenfassung der Datenstrukturen in eine einzelne Programmstruktur
    3. Aufbau einer Liste der Prozesse
    4. Erstellen von Pseudocode
  
- ◆ Hierarchical Input, Processing, Output (HIPO von IBM)
  - HIPO Diagramme bestehen aus drei Formen:
    - Hierarchiebaum der funktionalen Zerlegung (Table of Contents)
    - Übersichtsdiagramm (Eingabe - Verarbeitung - Ausgabe)
    - detaillierte HIPO-Diagramme

- ◆ Methoden basieren auf Objekten und Klassen und unterstützen Vererbung
  
- ◆ Definition von Objekt-Orientierung nach B. Meyer
  1. Object-based modular structure
  2. Data abstraction
  3. Automatic memory management
  4. Classes
  5. Inheritance
  6. Polymorphism and dynamic binding
  7. Multiple and repeated inheritance

- ◆ Begriffe mit Lebenszyklus-Konzept
  - ANSI / IEEE Std. 729 - 1983 „Glossary of Standard Software Engineering Terminology“
- ◆ Allgemeine Qualitätssicherung (zwei Normen)
  - ANSI / IEEE Std. 730 – 1984 „Software Quality Assurance Plans“  
Maßnahmen zur Sicherung der Qualität im gesamten Lebenszyklus
  - ANSI / IEEE Std. 983 – 1986 „Software Quality Assurance Planning“  
Planung der Qualitätssicherungsmaßnahmen
- ◆ Spezifikation der Anforderungen mit Fallbeispiel
  - ANSI / IEEE Std. 830 – 1984 „Software Requirements Specifications“

- ◆ Entwurf der Software-Systeme
  - ANSI / IEEE Std. 1016 – 1987 „Software Design Description“  
mit genormten Diagrammtechniken zur Entwurfsdokumentation
- ◆ Programmierung
  - ANSI / IEEE Std. 999 – 1987 „ADA as a Program Design Language“  
Empfehlung, ADA als Programmierungssprache zu nutzen
- ◆ Programmtest
  - ANSI / IEEE Std. 1008 – 1987 „Software Unit Testing“  
mit ausführlichem Modultestverfahren

- ◆ Systemtest
  - ANSI / IEEE Std. 829 – 1983 „Software Test Documentation“  
Beschreibung von Testplänen, Testfallentwürfen,  
Testprozeduren und Testberichten
- ◆ Systemabnahme
  - ANSI / IEEE Std. 1012 – 1986 „Software Verification and Validation Plans“  
Liste der wichtigsten Abnahmekriterien
- ◆ Systemverwaltung
  - ANSI / IEEE Std. 828 – 1983 „Software Configuration Management Plans“  
Richtlinien für Versionsführung, Änderungsdienst und Zustandsverfolgung

## Literaturangaben

- [1] Ian Sommerville: „*Software Engineering*“,  
ADDISON-WESLEY Verlag,  
ISBN 3-8273-7001-9
- [2] Siegfried Stein:  
„*Software Engineering Methoden*“,  
IBM Bildungszentrum NordWest,  
DOC SWE-ME SCRIPT 1992
- [3] *Extremes Programmieren*,  
Informatik Spektrum, Springer-Verlag  
Band 23, Heft 2, April 2000