Beobachtungen beim "Programmieren im Großen"

- Der Mensch kann kaum 2 Pfade (parallel) übersehen, maximal etwa 7 Objekte
- Die Zahl der inneren Verbindungen steigt mind. quadratisch mit der Größe an
- Module in großen Systemen müssen extrem fehlerfrei sein:
  Jedes Modul sei korrekt mit Zuverlässigkeit p_1 (<1). Dann ist die Wahrscheinlichkeit für gleichzeitiges Funktionieren von N Teilen P =
  Dieser Wert ist auch für
- Ergebnis: Eine lange Liste von Softwareprojekt-Fehlschlägen bzw. software-erzeugten Problemen:
  - nur 9% der Projekte werden fristgerecht und in etwa zur Kundenzufriedenheit ausgeliefert!
  - nahezu 1/3 scheitern vollständig
  - von den Fehlern beruhen etwa 1/3 auf Codierungsfehlern, der Rest auf "Verständnisproblemen"
  - Beispiele: Ariane-Absturz, Flughafensteuerung Denver, Airbus-Steuerung, ...

Vergleich Fortschritt: Hardware vs. Software

- Hardware wird seit 1950 in den Kerngrößen um 20-30 % pro Jahr verbessert, d.h. 1000 x (Anzahl der Transistoren, Energieverbrauch/MIPS, Kosten/MIPS, Kosten/MByte)
- Software dagegen nur 1-10 fach, d.h. eine LOC (Line of Code) kostet ungefähr heute soviel wie vor 30 Jahren - eventuell ist die LOC heute mächtiger durch moderne Sprachen
- "Software-Krise" durch geringe Produktivität und mangelhafte Softwareprojekte
- Immer weniger Hardware/Betriebssystem-Entwickler, immer mehr Anwendungsentwickler notwendig

Probleme der Software-Entwicklung

- Dauer von Softwareprojekten (1 Jahr bis zu mehreren Jahren)
- Umfang von Softwarepaketen (z.B. 50.000 Zeilen Quellcode)
- Größe der Entwicklergruppen (z.B. 5-200 Personen)
- Durchschnitt über die gesamte Entwicklungszeit:
  10 Zeilen Quellcode pro Tag
- Fehlerrate im Durchschnitt: 5-6 % (50-60 F. in 1000 Zeilen Code)
  - nach Auslieferung immer noch 0,4%
  - meistens Fehler in der Analyse- und Design-Phase
- Präsentation und Oberflächen erst am Ende entwickelt
- Dokumentation nach Entwickleranforderungen
  (nicht Kundenanforderungen)
- Auslieferung unter Zeitdruck