

## Eigenschaften von Software

- Immateriell, keine physikalischen Grenzen  
(außer Speicher & evtl. Laufzeit)
- kann „leicht“ geändert & erweitert werden
- nutzt sich nicht ab
- immer neue Anforderungen
- immer aufwändigeres Programme
- Aufwand steigt überproportional mit dem Umfang  
(Kommunikations-, Verwaltungsaufwand)

## Produktqualität

- Korrektheit
- Zuverlässigkeit
- Benutzerfreundlichkeit
- Wartungsfreundlichkeit
- Effizient
- Portabilität

## Prozessqualität

- kurze Entwicklungszeit
- geringe Kosten
- geringer Personaleinsatz
- Planbarkeit
- Projektverfolgung
- Hochwertigkeit (Überprüfung)

## Zwischenergebnisse

## Entwicklerproduktivität

- Frage: Wie viele Quellzeilen pro Monat schafft ein „durchschnittlicher“ Entwickler?
- Antwort: ca. 350 Quellzeilen / Monat
  - das sind 17,5 Zeilen pro Tag bei 20 Arbeitstagen / Monat
  - bei einem 8-Stunden-Tag: 27,6 Minuten pro Zeile
- nach einer Studie von Hewlett-Packard, 1992
- ohne Kommentare & Leerzeilen, aber mit
  - Spezifikation
  - Implementierung
  - Dokumentation
  - Test

## Abschätzung des Entwicklungsaufwands

- Folgerung für ein Projekt mit geschätzten 200.000 Quellzeilen:  
577 „Mannjahr“, also 47,6 „Mannjahre“
- „Mehrjäckchenrechnung“

$$\text{Entwicklungszeit} = \frac{\text{Mannjahr}}{\text{Personen}}$$

Personen	3	5	10	20
Jahre	75,8	9,5	4,8	2,4

- Mit 10000 Entwicklern wäre das Projekt in knapp 2 Tagen fertig
- jeder schreibt 20 Zeilen

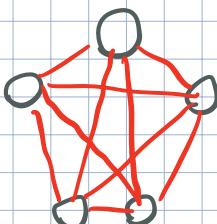
**NEIN!**

## Folgen der Arbeitsteilung

- zur Entwicklungsaufwand kommen noch:
  - teamgerechter Entwurf (Schnittstellen)
  - Kommunikation zwischen Entwicklern
  - Werkzeiten (bei ungleicher Leistungsteilung)
  - Verwaltung (Projekt-Management)
  - Integrationsprobleme beim Zusammenfügen der Einzelteile

## Schnittstellen bei n Teilen

Teile	Schnittstellen
2	1
3	3
4	6
5	10
10	45
20	190
n	$\frac{n \cdot (n-1)}{2} \approx \frac{n^2}{2}$



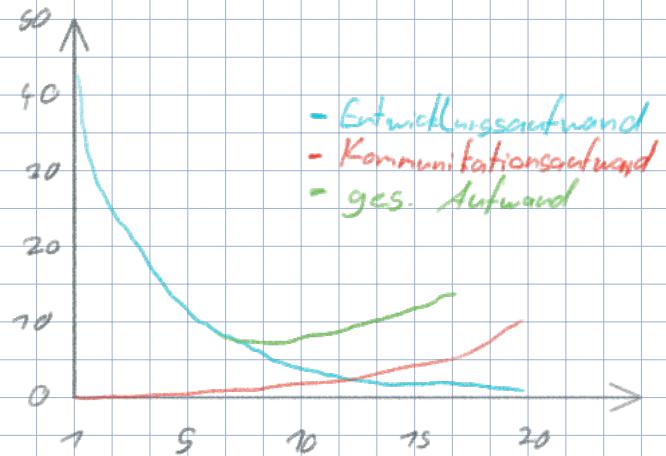
$$T(n) = \frac{k_1}{n} + k_2 \frac{n^2}{2}$$

Zeitbedarf

↓      ↓      ↑

Entwickler      Entwicklungsaufwand      Kommunikation

- Annahmen
- Gesamtaufwand
- 47,6 Mannjahre
- Kommunikationsaufwand / Monat pro Schnittstelle



## Abhilfe: Beschränkung der Kommunikationskanäle



Kommunikation soll nur dann erforderlich sein, wenn die bearbeiteten Software-Artefakte untereinander zu tun haben

- ⇒ Möglichst wenige Verbindungen zwischen Software-Teilen
- ⇒ Software-architekturelle & organisatorische Struktur des Entwicklungspersonals hängen eng zusammen.

## Was ist Architektur?

- Regelsystem
- Architektur eines Software-Systems = die **Struktur** des Systems:
- „Elemente“ aus denen das System besteht
- äußerlich sichtbare Eigenschaften der Elemente
- Beziehungen zwischen den Elementen

- Genauer: lie Strukturen oder Strukturen
- je nach Perspektive verschiedene Strukturen möglich:
  - Modulkonstruktion
  - Vererbungshierarchie
  - Prozess - Sicht
  - Datenzugriffe
  - Ausführungsreihenfolgen
  - ...

### Elemente:

- |                       |                           |
|-----------------------|---------------------------|
| • Modul               | • Datei, Datenbanktabelle |
| • Klasse              | • Prozesse                |
| • Übersetzungseinheit | • externes Gerät          |
| • Package             | • ...                     |

### Außerlich sichtbare Eigenschaften

- Architektonisch behandelt Abstraktionen der Elemente
- Interne Details werden absichtlich weggelassen
- Beschreiben werden jene Eigenschaften eines Elements, die für andere Elemente von Bedeutung sind
- Public / Private:
  - Offenbare Schnittstellen gehören zur Architektur
  - Implementierungsdetails sind aus architektonischer Sicht irrelevant

## Beziehungen zwischen Elementen

- Je nach Art der Elemente
  - Aufrufbeziehung (Funktionen, Methoden)
  - Vererbungsbeziehungen (Klassen)
  - Enthält - Beziehung (Modul / Funktion, Package / Klasse)
  - Datenzugriff (Funktionen, Module, Dateien)
  - Synchronisation (Prozesse)
  - ....

## Architektur = "Einschränkungen" & Verbote

- Architektur schränkt die Freiheiten der Entwickler & Implementierer ein
  - Funktionen in Ebene 1 dürfen nur auf Funktionen in Ebene 2 zugreifen, nicht auf Ebene 3 & 4
  - Geräte dürfen nur über die Kompatibilitätsmodelle benutzt werden; direkte Zugriffe auf Geräte sind verboten
  - Alle Datenlaufragriffe haben über die Netzwerkschnittstelle zum Server zu erfolgen.
  - Bildschirmanzüge werden ausschließlich vom GUI-Prozess veranlasst

## weitreichende Wirkung von Architekturentscheidungen

- Architekturentscheidungen sind grundlegend und können später kaum mehr geändert werden, z.B:
  - Aufteilung auf Client und Server
  - Regelung von Zugriffen auf Hardware
- Fehler beim Architekturentwurf sind im weiteren Projektverlauf nur mit sehr hohen Kosten zu beheben
  - Architektur muss besonders sorgfältig geplant und entworfen werden
  - Erfordert Kenntnis der wichtigsten funktionalen und Qualitätsanforderungen
  - Architektur sollte vor den weiteren Schritten validiert werden

## Qualität

### Qualitätsmerkmale aus dem Software Engineering

Korrektheit

Zuverlässigkeit

Effizienz

Portabilität

Benutzerfreundlichkeit

Wartungsfreundlichkeit

Adäquatheit

Lesbarkeit

Erlernbarkeit

Eweiterbarkeit

Robustheit

Testbarkeit

#### **Korrektheit:**

- Übereinstimmung zwischen Spezifikation und Endprodukt
- Problem: Wie genau ist die Spezifikation?
  - Unspezifische Aspekte
  - Spezifikation der Benutzerschnittstelle
  - Spezifikation des Fehlerverhaltens
  - Spezifikation der Hardware
  - Spezifikation externer Schnittstellen

#### **Zuverlässigkeit:**

- Wahrscheinlichkeit, dass ein Programm während einer gegebenen Zeitdauer die (spezifizierten) Erwartungen erfüllt
- Programm mit niedriger Korrektheit kann trotzdem zuverlässig sein (z.B wenn fehlerhafte Teile selten benötigt werden) und umgekehrt (nur 1 Fehler, aber an wesentlicher Stelle).

#### **Effizienz:**

- Ausnutzung vorhandener Ressourcen:
  - Zeit
  - Speicher

- Übertragungskanäle
- periphere Einheiten

### **Portabilität:**

- Übertragbarkeit auf andere „Umgebungen“
  - Prozessormodule
  - Betriebssysteme
  - Netzwerkkonfigurationen
  - Datenbanken

### **Benutzerfreundlichkeit:**

- Adiquatheit
  - Wie gut „passt“ das Programm?
  - Funktionsumfang
- Erlernbarkeit
  - Wie leicht kann man sich einarbeiten?
  - Klarheit, Hilfestellung, Assistenten, Benutzerhandbuch
- Rubustheit
  - Wie können Fehler vermieden werden?
  - Bedienungs-, Hardware-, Software-Fehler
  - Sicherung der Konsistenz

### **Wartungsfreundlichkeit:**

- Lesbarkeit
  - Ist der Quelltext verständlich?
  - Struktur, Stil, Namenswahl, Kommentierung, ...
- Erweiterbarkeit
  - Welchen Aufwand verursachen neue Anforderungen?
  - Struktur, Modularisierung, Kapselung, ...
- Testbarkeit
  - Wie leicht können Fehler gefunden und korrigiert werden?
  - Erkennung, Lokalisierung, Behebung

## Testaufwand

(nach Sommerville)



kommerzielle Software

Wissenschaft

... - - - .

Steuerungssysteme

... - - - .

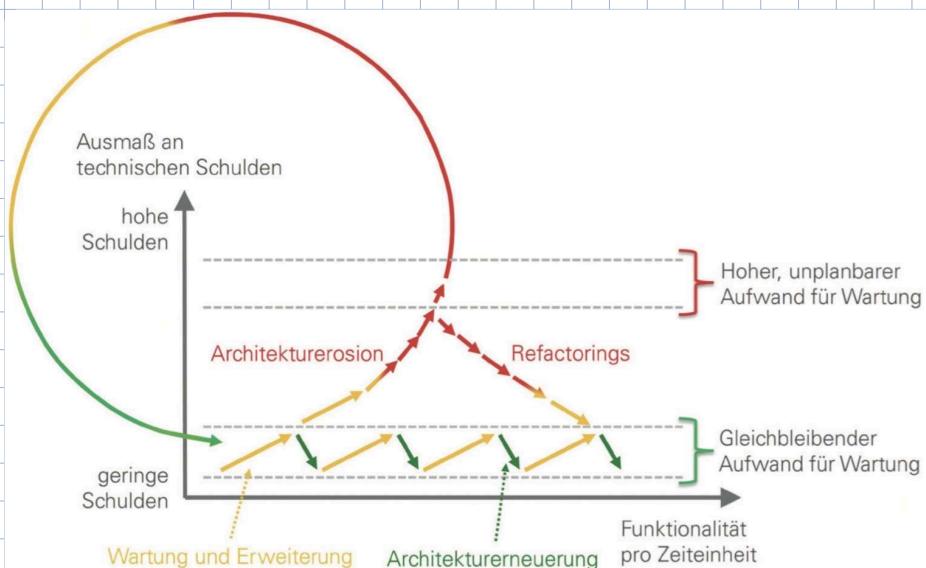
Raumfahrt

... - - - .

Betriebssysteme

... - - - .

Daten 40:20:40 Regel

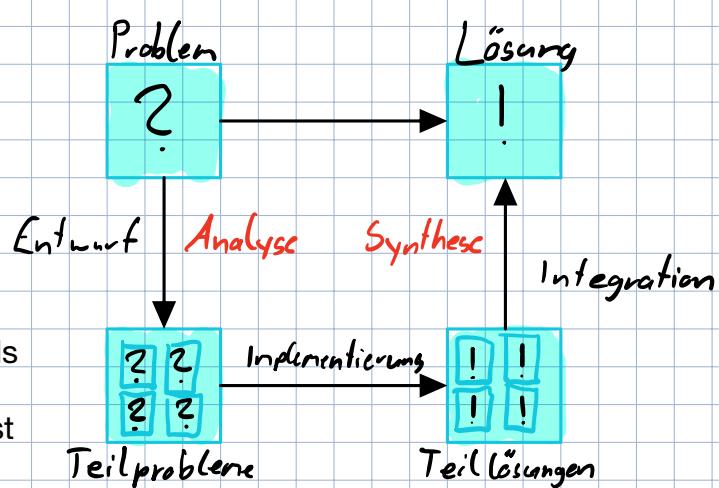


## Motivation:

- Häufiger Fehler: übereilter Beginn mit Detailentwurf und Implementierung

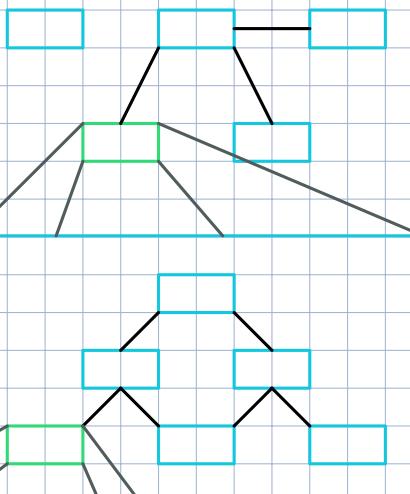
- Folgen daraus:

- unzureichendes Problemverständnis
- Selbstüberschätzung, Optimismus
- Hackertum
- Orientierung an Sprach- und Maschinendetails
- „Ad-Hoc-Struktur“: Entstandene Architektur ist
  - nicht dokumentiert, nicht überschaubar
  - nicht an die Anforderungen angepasst
  - für Wartungsarbeiten und Erweiterungen ungeeignet
  - kaum zu ändern

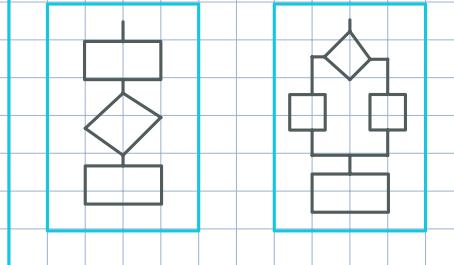


## Entwurfsebenen

1. Zerlegung in Subsysteme

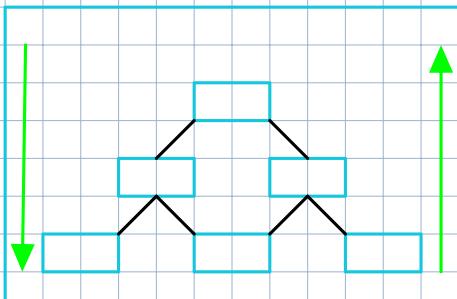


2. Modularisieren



3. Algorithmen

In allen Ebenen:  
parallel Behandlung  
von Algorithmen  
und Datenstrukturen



Zerlegung in Zusammensetzung  
Teilprobleme von Teillösungen  
(Analyse) (Synthese)

Ausgangspunkt: Ausgang  
Vision vom Bibliothek &  
Gesamtsystem Teilfunktionen

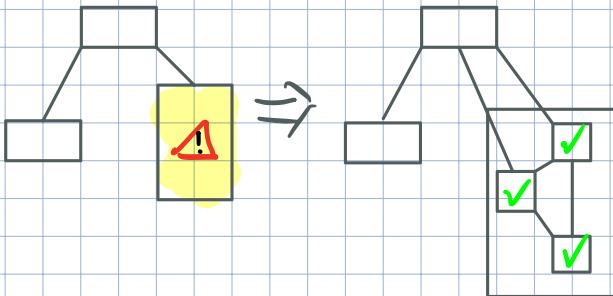
Top-Down      Bottom-Up

## Modularisierung:

- **Modul** = Zusammenfassung von Operationen und Daten zur Realisierung einer in sich abgeschlossenen Aufgabe.
- Kommunikation mit der Außenwelt nur über eine eindeutig definierte **Schnittstelle**.
- **Integration** in ein Programmsystem muss ohne Kenntnis des inneren Arbeitens möglich sein
- **Korrektheit** muss ohne Kenntnis der Einbettung in ein Programmsystem nachprüfbar sein.

## Komplexitätsreduktion:

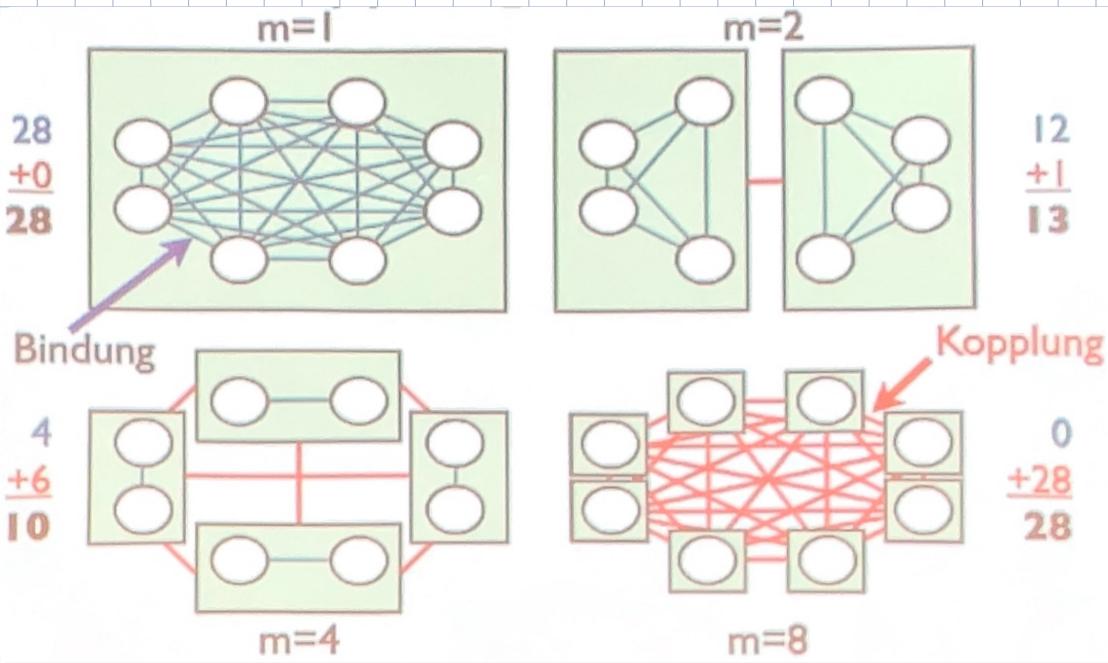
- Problem: Hohe (gemessene oder geschätzte) Komplexität eines Systemteils
- Abhilfe: Zerlegung großer Bausteine in mehrere kleine



- wenig große Module:
  - hohe innere Modulkomplexität
- viele kleine Module:
  - niedrige innere Modulkomplexität
  - aber viele Verknüpfungen zwischen Modulen

### Begriffe:

- **Modulbindung** = Innerer Zusammenhalt
- **Modulkopplung** = Verknüpfungen zwischen Modulen



Zerlegung in viele Module verlagert die Koppelkraft von der Modulbindung zum Modulkopplung

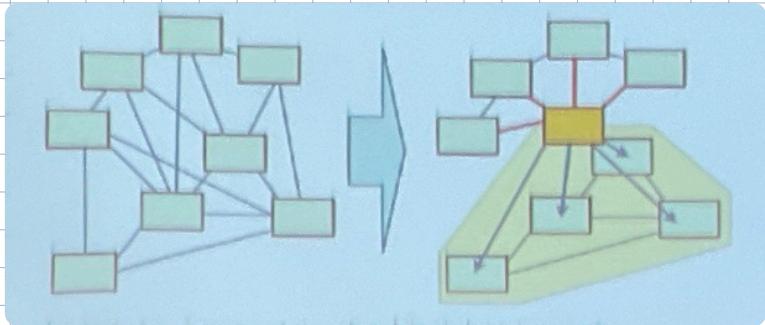
### Modularisierungskriterien:

- Modulgeschlossenheit
- Modulkopplung
- Minimalität der Schnittstelle
- Modulgröße
- Testbarkeit
- Interferenzfreiheit
- Verwendungszahl und Importzahl
- Moduhierarchie

### Modulkopplung:

- Stärke der Verbindung zwischen Modulen
- Niedrige Kopplung bewirkt, dass....
  - Module voneinander unabhängig sind,
  - Module durch andere Implementierungen ausgetauscht werden können,
  - Module einzeln getestet werden können,
- Idealfall: so locker wie möglich d.h:

- keine exportierten Daten
  - keine zyklische Verwendung (A benutzt B, B setzt A voraus)
  - enge Schnittstellen
  - kein „Modulgeflecht“ mit undurchschaubaren Wechselwirkungen
- Entflechtung bei hoher Kopplung durch „Clustering“ (d.h. Bildung von Subsystemen)

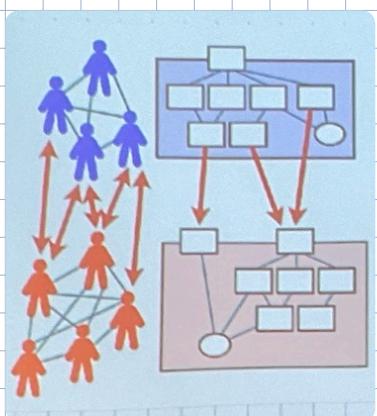


Rekursive Anwendung der Modularisierung!

### Minimierung von Schnittstellen:

#### „Schlankheitskur“ für Schnittstellen:

- Information Hiding:
  - interne Details (Daten und Datentypen) vor anderen Modulen verbergen
- lange Parameterlisten: in mehrere einfache Funktionen zerlegen
- Hilfsfunktionen geheim halten
- Clustering (eine schmale öffentliche Schnittstelle für eine Gruppe von Modulen)



Bindung:

Kommunikation inner der Teams; unproblematisch, Probleme werden werden informell gelöst.

Kopplung:

Kommunikation zwischen Teams; erfordert Koordination, Sitzungen, formelle Vereinbarungen, Berichtswesen, ....