Software-Engineering

Sebastian Iwanowski
FH Wedel

Kapitel 5: Systementwurf

Systemanalyse vs. Softwareentwurf

Systemanalyse

- **beschreibt das System der Anwendung, für das eine Aufgabe gelöst werden soll**
- **Modellierung orientiert sich an der Realität**

Softwareentwurf

- **beschreibt den Aufbau der Software, mit der die Aufgabe gelöst werden soll**
- **Modellierung orientiert sich an den technischen Möglichkeiten**

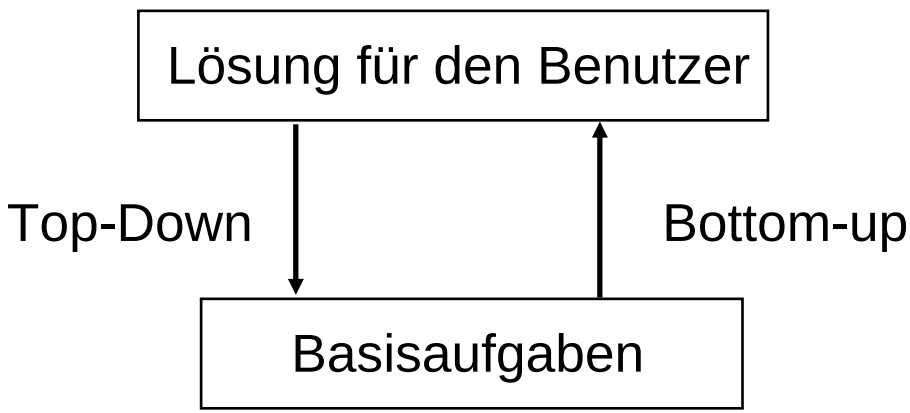
Die Modellierungshilfsmittel sind dieselben:

- **Abstraktion, verschiedene Sichten, hierarchische und modulare Zerlegung**
- **dieselben formalen Hilfsmittel: ERM, UML, ...**

Softwareentwurf: Allgemeine Prinzipien

Entwurfsrichtung

1) **Top-Down**
   - **vom Allgemeinen zum Konkreten**
2) **Bottom-Up**
   - **Vom Konkreten zum Allgemeinen**



Softwareentwurf: Top-Down

Vorgabe:

Abstrakte Zielvorgabe für die Funktionalität des Systems

Vorgehensweise:

Zerlegung der Zielvorgabe in Teilaufgaben und Konkretisierung

Vorteile

- Lösung steht im Mittelpunkt der Systementwicklung.
- Vollständige, exakte Spezifikationen führen genau zum gewünschten Ergebnis.
- Die Gesamtübersicht erleichtert die Bildung geeigneter Abstraktionsebenen und führt zu einer guten Systemstruktur.

Softwareentwurf: Top-Down

Vorgabe:

Abstrakte Zielvorgabe für die Funktionalität des Systems

Vorgehensweise:

Zerlegung der Zielvorgabe in Teilaufgaben und Konkretisierung

Nachteile

- Gute Fähigkeiten zum abstrakten Denken erforderlich.
- Realisierungsprobleme werden erst spät erkannt.
- Die Wiederverwendung vorhandener Teillösungen wird erschwert.

Softwareentwurf: Bottom-Up

Vorgabe:

Konkrete Bausteine für die Basislösung mit bekannter Funktionalität

Vorgehensweise:

Zusammensetzung dieser Bausteine zu größeren Einheiten

Vorteile:

- Realistischer Lösungsansatz durch sicheren Ausgangspunkt
- Suche nach vorhandenen Teillösungen wird begünstigt.

Softwareentwurf: Bottom-Up

Vorgabe:

Konkrete Bausteine für die Basislösung mit bekannter Funktionalität

Vorgehensweise:

Zusammensetzung dieser Bausteine zu größeren Einheiten

Nachteile:

- Schwierigkeiten, das Ziel im Auge zu behalten (und zu erreichen)
- Gefahr der Entwicklung von später nicht benötigten Softwarekomponenten auf unteren Ebenen des SW-Systems

Softwareentwurf: „Jo-Jo-Methode“

Vorgabe:

Konkrete Bausteine für die Basislösung mit bekannter Funktionalität

Zielvorgabe für das gesamte System

Vorgehensweise:

Zerlegung der Zielvorgabe in Teilaufgaben und Konkretisierung mit Hilfe der Bausteine für die Basislösung

Vorteile

- Bestmöglicher Kompromiss zwischen Wünschenswertem und Machbarem
- Erstellen einer möglichst ökonomischen Lösung
- Anforderungen an das Abstraktionsvermögen reduziert

Softwareentwurf: „Jo-Jo-Methode“

Vorgabe:

Konkrete Bausteine für die Basislösung mit bekannter Funktionalität

Zielvorgabe für das gesamte System

Vorgehensweise:

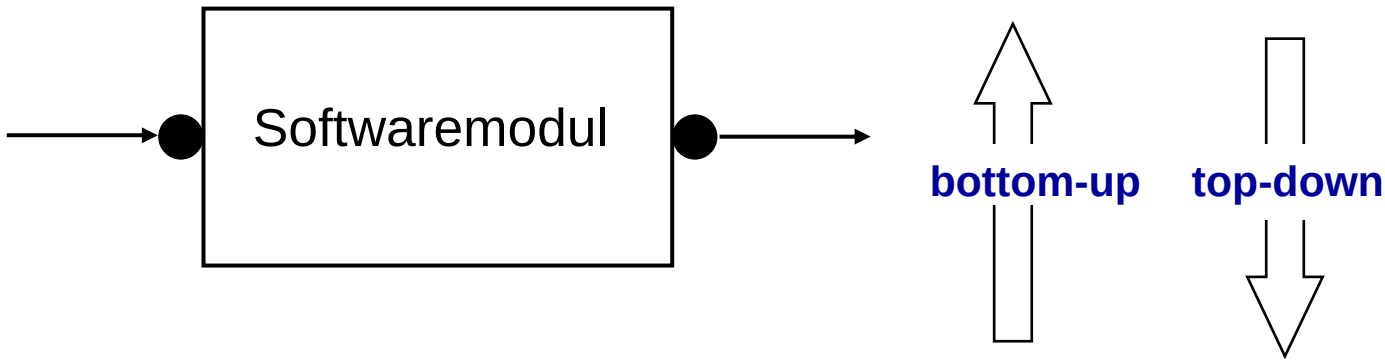
Zerlegung der Zielvorgabe in Teilaufgaben und Konkretisierung mit Hilfe der Bausteine für die Basislösung

Nachteile

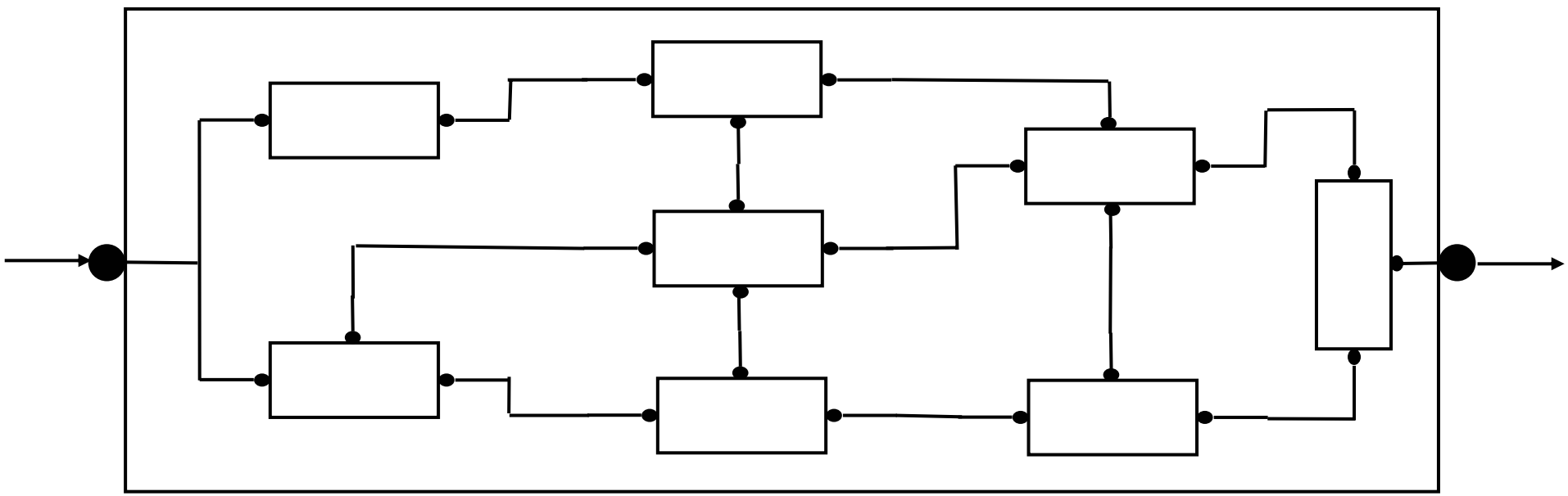
- Wechsel der Bearbeitungsrichtung willkürlich, daher Gefahr mangelnder Systematik

Softwareentwurf: Modularisierung

Entwurfsebene 1:



Entwurfsebene 2:



Softwareentwurf: Modularisierung

Der Begriff des Moduls (nach Goos / Dennis 1973):

- Ein **Modul** ist die Zusammenfassung von Operationen und Daten zur Realisierung einer **in sich geschlossenen Aufgabe**.
- Die **Kommunikation** eines Moduls mit der Außenwelt darf nur über eine **eindeutig spezifizierte Schnittstelle** erfolgen.
- Zur **Integration** eines Moduls in ein Programmsystem muss die **Kenntnis des inneren Aufbaus entbehrlich** sein.
- Zum **Nachweis der Korrektheit** eines Moduls muss die **Kenntnis der Einbettung** des Moduls in das Softwaresystem **entbehrlich** sein.

Leitlinien für die Modularisierung

Die Aspekte bei der Modularisierung:

- Modulbindung
- Modulkopplung
- Schnittstellen
- Modulgröße
- Interferenzeigenschaften
- Importzahl
- Verwendungszahl

Leitlinien für die Modularisierung

Modulbindung

- Grad des Zusammenhangs zwischen den einzelnen Daten und Operationen desselben Moduls
- eng ↔ lose

Modulkopplung

- Grad des Zusammenhangs zwischen verschiedenen Modulen
- eng ↔ lose

Schnittstellen

- Datenaustauschstellen zwischen Modul und dem umgebenden System
- viele ↔ wenige

Leitlinien für die Modularisierung

Modulgröße

- Anzahl der verwendeten Daten bzw. Operationen in einem Modul
- groß ↔ klein

Interferenzeigenschaften

- Grad der Beeinflussung anderer Module durch Seiteneffekte
- hoch ↔ niedrig

Importzahl

- Anzahl der in diesem Modul verwendeten Module
- viele ↔ wenige

Verwendungszahl

- Anzahl der Module, die diesen Modul verwenden
- viele ↔ wenige

Leitlinien für die Modularisierung

Folgende Kompromisse sollten geschlossen werden:

- Ausgewogenes Verhältnis zwischen Modulbindung und Modulkopplung
- Minimale Schnittstellen (Ausnahme: wenn sonst das Modul zu groß wird)
- Modulgröße maximal so groß, dass eine Person das Modul in einem überschaubaren Zeitraum bearbeitet
- Mittlere Import- und Verwendungszahlen

1. Bsp. für eine Modulart: Bibliotheksmodule

Bereitstellung einer Sammlung von Algorithmen / Funktionalitäten zu einem Aufgabenbereich

Besonderheiten:

- Umfangreiche Schnittstelle (eine Ausnahme zu den Leitlinien)
- Kein interner Zustand (Modul erfüllt keine Gesamtfunktionalität)

Beispiele:

- Module mit mathematischen Funktionen
- Module für Hilfsfunktionen, die nicht problemspezifisch sind (z.B. Modul Util mit Umwandlungsfunktionen, Datumsfunktionen, etc.)

2. Bsp. für eine Modulart: Datenkapselung

Zweck:

Verhinderung des direkten Zugriffs auf Daten von anderen Programmteilen aus

Aufbau:

Die Datenkapsel besteht aus abstrakten Datenstrukturen/-typen und zugehörigen Zugriffsfunktionen.

„Abstrakt“ heißt: Der konkrete Aufbau der Datenstrukturen/-typen ist außerhalb des Moduls unbekannt.

Bsp.: Wörterbuchproblem (Dictionary)

Abstrakter Datentyp: `Dictionary`

Zugriffsfunktionen:

```
getDictionary()
insert(dict, key, value)
delete(dict, key)
search(dict, key)
```

Softwareentwurf

hier noch nicht angesprochen:

Entwurfsmuster für die Programmierung

➔ ***Thema der Vorlesung Software-Design***

Entwurf von Bedienungsoberflächen

➔ ***Thema der Vorlesungen Software-Ergonomie / Medienkonzeption***

Verwendung von CASE-Tools

Reine UML-Tools: **Eher für Softwareentwurf geeignet**

ARIS: **Eher für Systemanalyse geeignet**

Was ist vorzuziehen, wenn beides erledigt werden soll ?

- **Für den nahtlosen Übergang von Systemanalyse zu Softwareentwurf ist ein einheitliches CASE-Tool von Vorteil**
- **Wähle das CASE-Tool, das besser geeignet ist für die schwierigere Aufgabe**

und nicht vergessen: **Kundenwünsche haben Vorrang !**