5. Entwurf

SW-Entwicklungsphasen

Planung:

Machbarkeitanalyse; Anforderungsanalyse.

Entwurf:

Systemarchitektur; (schrittweise) Zerlegung in Komponenten.

Implementierung und Modultest:

Codierung; Test Modul $\leftrightarrow$ Spezifikation.

Integration, Systemtest:

Zusammenbau der Komponenten;
Test Gesamtsystem $\leftrightarrow$ Spezifikation;
Abnahme durch den Kunden.

Wartung:

System ist in Betrieb; ggf. Fehlerkorrektur, Funktionserweiterung.

Entwurfsprozeß (I)

Ausgangspunkt:

Anforderungsspezifikation

(Zwischen-)Ziele:

Möglicherweise:
formalisierte Anforderungsspezifikation,

Spezifikation von Teilsystemen,

Spezifikation von Algorithmen,

Spezifikation von Datenstrukturen.

Entwurfsprozeß (II)

Phasen:

Architekturentwurf
$\leadsto$ welche Teilsysteme?

Abstrakte Spezifikation
$\leadsto$ was tun die Teilsysteme?
welche Rahmenbedingungen?

Schnittstellenentwurf
$\leadsto$ wie interagieren Teilsysteme?
welche Schnittstellen?

Komponentenentwurf
$\leadsto$ wie Teilsysteme weiter zerlegen?

Entwurf von Datenstrukturen und Algorithmen
$\leadsto$ wie im Detail?

Entwurfsprozeß (III)

Häufig empfohlen:

Top-down-Entwurf
$\leadsto$ Hierarchische Struktur
(in der Regel: DAG).

Aber Abweichungen möglich und sinnvoll:

Bisherige Erfahrungen des Designers
$\leadsto$ nicht jede Komponente muß streng hierarchisch zerlegt werden.

Schwierige Teile vorziehen
$\leadsto$ bessere Zeitabschätzung.

Wiederverwendung existierender Komponenten
$\leadsto$ teilweise: Bottom-up-Entwurf.

Entwurfsprozeß (IV)

Generelle Schritte:

Problem verstehen.

Mögliche Lösungen suchen (am besten mehrere) und beurteilen. Kriterien:

Erfahrung des Designers,

Wiederverwendung existierender Komponenten,

Einfachheit der Lösung,

Effizienz der Lösung.

In der Lösung verwendete Komponenten beschreiben.

Notationen

Graphische Notationen:

Beziehungen zwischen Komponenten und zur realen Welt darstellen.

$\leadsto$ Systemarchitektur, Datenstrukturen.

Programmbeschreibungssprachen:

Programmiersprachen + Erweiterungen (+ informeller Text).

$\leadsto$ Schnittstellen, Algorithmen.

Informeller Text:

$\leadsto$ generelle Ideen des Entwurfs,

$\leadsto$ nicht-funktionale Eigenschaften.

Entwurfsstrategien (I)

Funktionaler Entwurf:

System = Menge interagierender Funktionen.

Systemzustand: global (abgesehen von lokalen Daten der Funktionen)

Objektorientierter Entwurf:

System = Menge interagierender Objekte.

Systemzustand: dezentral,
Objekte haben eigenen Zustand.

Entwurfsstrategien (II)

In der Praxis Entwurfsstrategien häufig gemischt, z B.:

- obere Schichten: objektorientiert (Objekte $\approx$ Objekte der realen Welt),
- mittlere Schichten: funktional (z. B.: verwendete Algorithmen funktional spezifiziert),
- untere Schichten: objektorientiert (Objekte $\approx$ Objekte der realen Welt, Objekte $\approx$ Datenstrukturen).

Entwurfsqualität (I)

Kriterien:

- Effizienz:
  - Laufzeit,
  - Speicherplatz.
- Wartbarkeit:
  - Erweiterbarkeit,
  - Änderbarkeit,
  - Verständlichkeit.

Entwurfsqualität (II)

Wartbarkeit:

- Komponenten sollen für sich allein verstehbar sein.
- Änderungen sollen möglichst lokal durchgeführt werden können.
- $\rightsquigarrow$ starker Zusammenhalt der Komponenten.
- $\rightsquigarrow$ schwache Kopplung der Komponenten.
- $\rightsquigarrow$ Abgeschlossenheit der Komponenten.

Zusammenhalt (I)

Formen des Zusammenhalts:

- Logischer Zusammenhalt: Unterkomponenten verrichten ähnliche Aufgaben (z. B.: Eingabefunktionen).
- Zeitlicher Zusammenhalt: Unterkomponenten werden zur gleichen Zeit verwendet (z. B.: Initialisierungsroutinen).
- Prozeduraler Zusammenhalt: Unterkomponenten bilden zusammen eine Kommandosequenz.
- ...

Zusammenhalt (II)

Formen des Zusammenhalts:

...

Kommunikationszusammenhalt: Unterkomponenten benutzen Eingabedaten oder produzieren Ausgabedaten der gleichen Art.

Sequentieller Zusammenhalt: Die Ausgabe einer Unterkomponente bildet die Eingabe der nächsten Unterkomponente.

Funktionaler Zusammenhalt: Alle Unterkomponenten sind notwendig, um eine einzelne Funktion zu berechnen.

Objektzusammenhalt: Operationen gestatten Zugriff auf eine einzelne Datenstruktur.

Kopplung

Schwach gekoppelte Komponenten:

Wenig gemeinsame Daten,

Kleine Schnittstellen,

Kommunikation mit wenigen anderen Komponenten.

Dokumentation

Dokumentation des Entwurfs:

Vollständig.

Verständlich.

Konsistent mit der Implementierung (auch nach Änderungen oder Erweiterungen des Systems)

Gut änderbar.

Vererbung

Vorteile der Vererbung:

Bessere Wiederverwendbarkeit.

Statt Komponenten zu ändern reicht es möglicherweise aus, Unterklassen zu bilden.

Probleme der Vererbung:

Möglicherweise: schwächerer Zusammenhalt.

Starke Kopplung zwischen Ober- und Unterklasse.