3.6 Projektentwicklung/Phase 3: Eine Struktur entwerfen

3.6.1 Einführung

Das Ziel der Entwurfsphase ist es, aus dem Grobentwurf eine Systembeschreibung zu erstellen, die:

a) so kleinteilig ist, dass man in der folgenden Phase nicht noch einmal entwerfen muss,

b) so genau spezifiziert ist, dass der Entwickler eines Systemteils keine Rückfragen mehr stellen muss,

c) so genau spezifiziert ist, dass man mit dieser Beschreibung ein Systemteil testen kann,

d) so genau spezifiziert ist, dass man aus den Systemteilen das Gesamtsystem aufbauen kann.

Zu Beginn der Entwurfsphase liegen vor:

- Pflichtenheft
- Grobentwurf

Am Ende liegen vor:

- Fachlicher Anteil (z. B. Steuerrecht):
  - Beschreibung des UI
  - Beschreibung der Daten
- IT-Anteil:
  - Aufbau des Gesamtsystems
  - Definition der Systemteile (Module)
- Organisatorischer Anteil:
  - Einführungsplan

3.6.2 Aufgaben und Probleme bei Software-Projekten

Traditionell betrachtet man bei prozeduralen Programmiersprachen vor allem die Programmstruktur. Dafür stehen die Top-Down-Methode und Struktogramm und (Programm-) Flussdiagramm (PAP, DIN 66001). Für technische Problemstellungen ist das in Ordnung; aber bei Problemstellungen aus Wirtschaft und Verwaltung und bei Simulationen fällt auf, dass auch Daten und ihre Strukturierung eine Rolle spielen. Wie bekommt man nun die Daten unter? Eine Lösung ist die getrennte Erstellung eines Daten-Flussdiagramms (auch in DIN 66001). Diese Lösung befriedigt nicht ganz, weil man nun zwei getrennte Diagramme hat. Deshalb wurde die HIPO-Methode entwickelt, bei der man Programmaufbau (=H wie Hierarchie) und Eingabe, Verarbeitung und Ausgabe (=IPO wie Input, Processing und Output) in dasselbe Diagramm verlegt.

Mit der Modularen (C) und der Objektorientierten (C++) Programmierung wurde die Bottom-Up-Methode populär, bei der aus zunächst kleinen Einheiten (Modulen oder Klassen) ein immmer größeres System aufgebaut wird. Moduldiagramme und UML-Diagramme illustrieren diese Methode.

Als Beispiel soll hier der Entwurf einer App `preisvergleich` dienen, die zum Vergleichen von Preisen bei einem Wocheneinkauf dient. Solch eine App gibt es zwar bereits, jedoch ist der Datenschutz bei dieser App möglicherweise nicht gewährleistet. Deswegen soll sie neu entwickelt werden.

3.6.3 Top-Down-Methode

Bei der Top-Down-Methode wird das Gesamtproblem in mehrere Unterprobleme aufgeteilt. Diese können wieder aufgeteilt werden. Das passiert so oft, bis jedes Problem einfach lösbar ist. Anschließend werden die Teilprobleme gelöst und die Teil-Lösungen zusammengesetzt.

Das Aufteilen in Teilprobleme ist eventuell nicht ganz einfach. Die Teilprobleme sollen ja möglichst in sich abgeschlossen sein, so dass zum Schluss zwischen den Modulen möglichst wenig Datenkopplung besteht (Prinzip der schmalen Datenkopplung).

Oft hilft es, ein Programm anhand der Benutzer-Menüs aufzuteilen: Jedes Menü bekommt einen eigenen Programmteil. Oder man listet Anwendungsfälle auf (*use cases*); anschließend kann z. B. jeder Anwendungsfall ein Programmteil sein; oder man bildet Gruppen ähnlicher Anwendungsfälle. Bei verteilten Systemen (z. B. Client und Server) ist die Aufteilung leichter[1].

Abbildung 1 zeigt den Hierarchie-Baum für `preisvergleich`: Die oberste Ebene bekommt

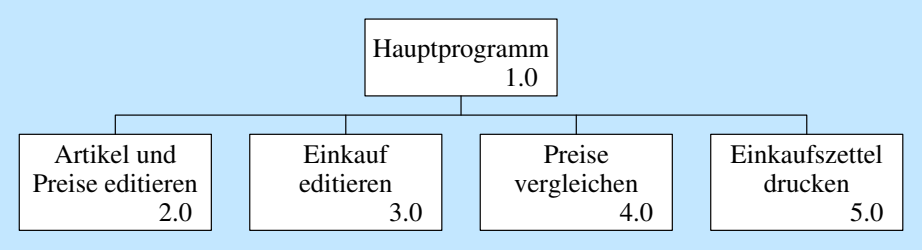

Abbildung 1: Beispiel eines Hierarchie-Baumes

die ID 1.0, die zweite Ebene 2.0 bis 9.0, die dritte Ebene 1.1 bis 9.9, darunter geht es weiter mit 1.1.1 bist 9.9.9.

3.6.4 HIPO-Methode

Bei der HIPO-Methode wird zusätzlich zum Hierarchie-Baum zu jedem Teilbaum ein IPO-Diagramm (Input-/Processing-/Output-Diagramm) gezeichnet, in dem die Ein- und Ausgabedaten dieses Programmteils benannt sind.

Abbildung 2 zeigt die IPO-Diagramme für `preisvergleich`. In jedem Diagramm soll die Beschreibung für die direkt untergeordneten Programmteile enthalten sein: 1.0 enthält 2.0 bis 5.0; 2.0 enthält 2.1 bis 2.x, 3.0 enthält 3.1 bis 3.x usw.; dies war hier natürlich nicht möglich, weil 2.1, 2.2 usw. im Hierarchie-Diagramm nicht existieren.

3.6.5 Modularer Entwurf

Man sieht, dass beim HIPO-Entwurf jeder Programmteil auf jedes E-/A-Element zugreifen kann. Das kann den Entwurf leider sehr unübersichtlich machen. Und man wird verführt, den Datenaustausch über Dateien oder globale Variablen zu realisieren.

Eine Abhilfe bietet der Modulare Entwurf. Er sieht vor, dass ein Datenfluss nur zwischen Programmteilen vorkommen kann, die einander aufrufen – genauso, wie es in C bei Funktionsaufrufen vorkommt. Abbildung 3 zeigt (anhand von drei Zeilen C-Code), wie der Datenfluss in das Baumdiagramm der Programmhierarchie eingebunden werden kann. Der Modulare Entwurf geht allerdings viel weiter und bietet deutlich mehr Möglichkeiten als hier abgebildet ist.

[1] Nicht aber das Systemdesign!

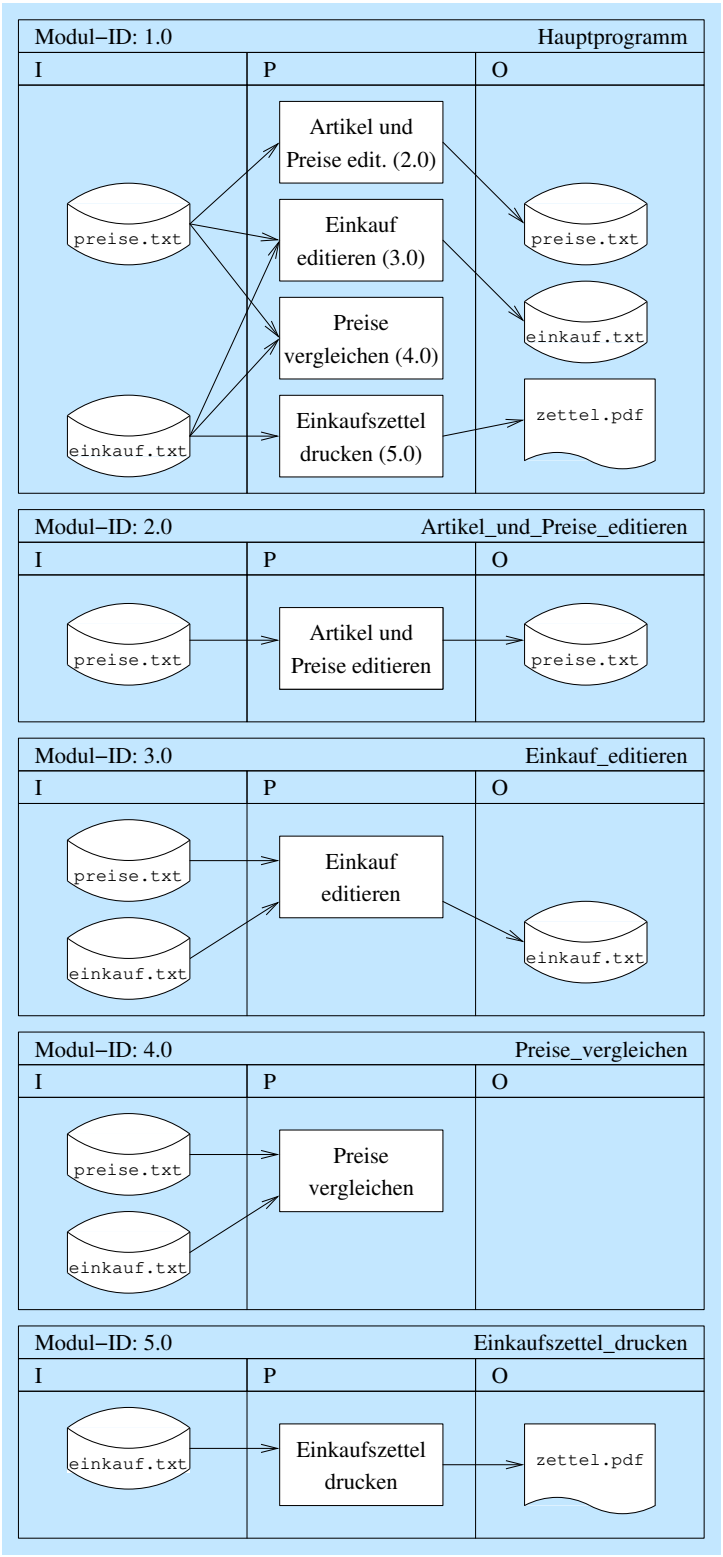

Abbildung 2: Beispiel eines IPO-Diagrammes

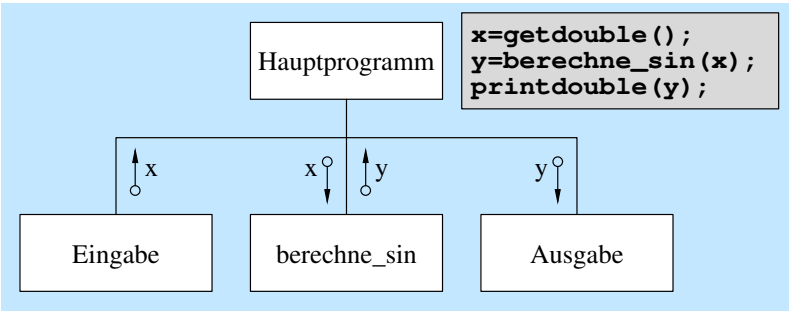

Abbildung 3: Beispiel eines Modularen Entwurfs