HANSER

Software Engineering

Gustav Pomberger, Wolfgang Pree

Architektur-Design und Prozessorientierung

ISBN 3-446-22429-7

Inhaltsverzeichnis

Inhaltsverzeichnis

Vorwort IX

Dank XI

1 Gegenstandsbestimmung – Einführung und Überblick 1

1.1 Was ist Software? 1

1.2 Was ist Software Engineering? 3

1.3 Inhalt und Aufbau des Buches 5

Teil I: Prozessorientierte Sicht – Organisation von Software-Projekten 9

2 Prozessmodelle 11

2.1 Das klassische sequenzielle Phasenmodell 11

2.2 Das V-Modell 18

2.3 Das Prototyping-orientierte Prozessmodell 26
2.3.1 Begriffe und Abgrenzung 26
2.3.2 Prozessmodell 29

2.4 Das Spiralmodell 32
2.4.1 Das Spiralmodell von Boehm 32
2.4.2 Das Spiralmodell von Pomberger und Pree 35

2.5 Der Unified Process 39

2.6 Ein objektorientiertes Phasenmodell 42

2.7 Leichtgewichtige (agile) Prozessmodelle 45

3 Software-Qualitätsmanagement 51

3.1 Was ist Softwarequalität? 51

3.2 Wo und wie entstehen Softwarequalität bzw. Qualitätsmängel 55

3.3 Grundzüge des Software-Qualitätsmanagements 57
3.3.1 Hauptaufgaben des Qualitätsmanagements 58
3.3.2 Objekte und Sichten des Qualitätsmanagements 58
3.3.3 Prinzipien des Qualitätsmanagements 60
3.3.4 Konstruktive und analytische Qualitätsmaßnahmen 61

Teil II: Konstruktions- und Architektur-orientierte Sicht 65

4 Elementare Konzepte und Konstrukte 67

4.1 Der Algorithmenbegriff 67

4.2 Datenobjekte, Datentypen und elementare Aktionen 68

4.3 Schnittstellen und Aktivierung von Algorithmen 70

4.4 Systematischer Entwurf von Algorithmen 72

4.5 Grundlegende Konzepte der objektorientierten Programmierung 76
4.5.1 Klassen, Objekte (= Instanzen von Klassen), Instanzvariablen, Methoden 77
4.5.2 Vererbung, Polymorphismus, statischer Variablentyp, dynamischer Variablentyp und dynamische Bindung 79

5 Konstruktion anpassbarer Software 85

5.1 Konfigurationsparameter als Basis für anpassbare Software 85
5.1.1 Parametereinstellungen über globale, statische Variable 85
5.1.2 Callback-Style of Programming – Funktionen und Prozeduren als Parameter 87

5.2 Konzepte und Konstruktionsprinzipien für anpassbare, objektorientierte Produktfamilien 89
5.2.1 Das Konzept der abstrakten Kopplung 91
5.2.2 Das Konzept der Template- und Hook-Methoden 97
5.2.3 Das Hook-Method-Konstruktionsprinzip 100
5.2.4 Das Hook-Object-Konstruktionsprinzip 103
5.2.5 Das Composite-Konstruktionsprinzip 109
5.2.6 Das Decorator-Konstruktionsprinzip 116
5.2.7 Zusammenfassung der Merkmale der Konstruktionsprinzipien 125

5.3 Konstruktionsprinzipien und Entwurfsmuster 125

6 Modularisierung und Software-Architekturen 131

6.1 Software-Module 132

6.2 Erwünschte Eigenschaften von Modulen 134
6.2.1 Stabile und verständliche Modulschnittstellen durch Information Hiding 135
6.2.2 Balance zwischen Kopplung und Kohäsion 137

6.3 Ausprägungen von Modulen 140
6.3.1 Modul als Abstrakte Datenstruktur (ADS) 140
6.3.2 Modul als Abstrakter Datentyp (ADT) 141
6.3.3 Module und Komponentenstandards 145

6.4 Beispiele für ausgewogene Modularisierungen 147
6.4.1 Kohäsionsverbesserung durch Aufteilung von Modulen 147
6.4.2 Module für die Simulation diskreter Ereignisse 149

6.5 Beschreibung von Software-Architekturen 156
6.5.1 Datenzentrierung 157
6.5.2 Datenflussorientierung 158
6.5.3 Call&Return 159
6.5.4 Virtuelle Maschine 160
6.5.5 Unabhängige Komponenten 161
6.5.6 Vor- und Nachteile der Architekturmuster 162

6.6 Analyse von Software-Architekturen 162
6.6.1 Die Software-Architektur-Analyse-Methode (SAAM) 162
6.6.2 Beispiel einer Anwendung der SAAM 167
6.6.3 Was ist bei einer Anwendung der SAAM zu beachten? Was sind die Vorteile und Risiken? 172

6.7 Mehrdimensionale Modularisierung durch Aspektorientierte Programmierung (AOP) 173
6.7.1 Das Problem einer einzigen, statischen Modularisierung 174
6.7.2 Grundlegende Sprachkonzepte und -konstrukte von AOP 176

6.8 Zusammenfassung wichtiger Modularisierungsprinzipien 180

Teil III: Ausgewählte Gebiete und Fallbeispiele 181

7 Transformationsorientierte Software 183

7.1 Grundlegende Konzepte 183
7.1.1 Kontextfreie Grammatiken 184
7.1.2 Konstruktion eines Sprachanalysators (Scanner und Parser) 186
7.1.3 Attributierte Grammatiken 192

7.2 Fallbeispiel: Transformation strukturierter Texte 196
7.2.1 Entwurf einer attributierten Grammatik mit Coco/R 199

7.3 Zusammenfassung 215

8 Web-Service-basierte Software 217

8.1 Was sind Web-Services 217

8.2 Vergleich mit anderen Technologien 219
8.2.1 CORBA 220
8.2.2 Java RMI 222
8.2.3 .NET Remoting Framework 222

8.3 Entwicklungsprozess für Web-Service-basierte Software 223

8.4 Grundlegende Konzepte und Standards 224
8.4.1 SOAP 224
8.4.2 Web-Service Description Language (WSDL) 226

8.5 Fallbeispiel: Elektronisches Telefonbuch 229
8.5.1 Ausgangssituation im Fallbeispiel 229
8.5.2 Beteiligte Komponenten 230
8.5.3 Schnittstelle des Web-Services 230
8.5.4 Implementierung des Web-Services mit Microsoft Visual Studio 231
8.5.5 Implementierung der Anwendungssoftware 232
8.5.6 Interaktionen zwischen Anwendungssoftware und Web-Service 234

8.6 Zusammenfassung 236

9 Eingebettete Echtzeitsoftware 237

9.1 Grundlegende Konzepte und Voraussetzungen 237
9.1.1 Charakteristika von Echtzeitsoftware 238
9.1.2 Logische versus tatsächliche Ausführungszeiten von Echtzeitsoftware 239

9.2 Fallbeispiel: Echtzeitsoftware eines autonom fliegenden Helikopters 241
9.2.1 Das OLGA-System 243
9.2.2 Die TDL-Spezifikation des Zeit- und Kommunikationsverhaltens 244
9.2.3 Die Übersetzung der TDL-Spezifikation in ein ausführbares Programm 247
9.2.4 Simulation eines TDL-basierten Regelungssystems 248

9.3 Zusammenfassung 250

Literaturverzeichnis 251

Stichwortverzeichnis 255