B 444/2082
~~Technische Hochschule Darmstadt
Fachbereich Informatik
Fachgebiet Praktische Informatik
Magdalenenstr. 11 c
6100 Darmstadt~~

Software Engineering with Student Project Guidance

BARBEE TEASLEY MYNATT

Bowling Green State University
Bowling Green, Ohio

Fachbereichsbibliothek Informatik
TU Darmstadt
59434004

Technische Universität Darmstadt
FACHBEREICH INFORMATIK
BIBLIOTHEK
Inventar-Nr.: M04-00262
Sachgebiete:
Standort: D.2/Myna

Prentice-Hall International, Inc.

Contents

Preface xiii

1. What Is Software Engineering? 1

1.1 Forces Behind the Emergence of Software Engineering 3
1.2 What Is Software Engineering? 5
1.2.1 Quality Software is a Primary Goal of Software Engineering 6
1.3 Software Life Cycle 9
1.3.1 A Closer Look at the Stages in the Software Development Cycle 10
1.3.2 Models, Approaches, and Methods in Software Engineering 12
1.3.3 An Alternative Development Model: Prototyping 15
1.4 Costs of Software Development and Ownership 17
1.5 Why is Software Development So Difficult? 21
1.5.1 Factor 1: Communications 21
1.5.2 Factor 2: Sequential Nature of System Development 22
1.5.3 Factor 3: Project Characteristics 23
1.5.4 Factor 4: Characteristics of Personnel 24
1.5.5 Factor 5: Management Issues 26
1.6 Understanding Systems 27
1.6.1 Understanding a System by Identifying Objects and Operations 28
1.7 Project Component of a Software Engineering Course 31
1.7.1 Getting Started on the Project 33
1.7.2 Team Organization 34

1.7.3 Guidelines for Holding Meetings 35
Summary 40
Checklist of Project Activities 41
Terms and Concepts Introduced 41
Exercises 42

2. Requirements Analysis and Specification 44
2.1 Requirements Analysis 45
2.1.1 Requirements Specification 46
2.1.2 Requirements Specification Document 46
2.2 Using Structured Analysis 48
2.2.1 Data-flow Diagram Notation 49
2.2.2 Abstraction and Decomposition in Systems Analysis 53
2.2.3 System Evolution 54
2.2.4 An Example: The Noncredit Courses Office 56
2.3 Creating the Data Dictionary 60
2.4 Activity Specifications Complete the Model 62
2.4.1 Using Pseudocode for Activity Specifications 63
2.4.2 Using Decision Tables for Activity Specifications 66
2.4.3 Comparison of Decision Tables to Pseudocode 68
2.5 Nonfunctional Requirements 70
2.6 Feasibility 74
2.7 Requirements Specification Document 74
2.8 Evaluating the Requirements Specification 76
2.8.1 Conducting a Formal Review 77
2.8.2 Criteria for Requirements Specification 79
2.9 Case Study: Rev-Pro — A Management/Pricing Tool for Arts Administrators 83
2.9.1 The Problem 84
2.9.2 Analysis of the Rev-Pro System 85
Summary 88
Checklist of Project Activities 89
Terms and Concepts Introduced 90
Exercises 91

3. Preliminary Design: Designing the User Interface 94
3.1 Preliminary Design Begins with the Design of the User Interface 94
3.2 Factors Affecting the Quality of Computer User Interfaces 95

3.3 Know the User Before Designing the Interface 97
3.4 Interaction Styles 98
3.5 Golden Rules of Interface Design 99
3.6 Menu Selection 101
3.6.1 Kinds of Menus 102
3.6.2 Other Menu Considerations 104
3.7 Form Fill-in Interface 105
3.8 Command Languages 106
3.8.1 Guidelines for Designing Command Languages 108
3.9 Direct Manipulation Interfaces 110
3.10 Designing Screen Displays 112
3.11 System Messages Design 116
3.11.1 Error Messages 117
3.11.2 Guidelines for Error Message Design 118
3.11.3 An Ounce of Prevention 119
3.12 User Training and Aid: Printed Manuals 120
3.12.1 Content 122
3.12.2 Organization and Format 123
3.12.3 Writing Style 124
3.13 User Training and Aid: On-line Facilities 125
3.14 User Interface Design Development and Implementation 126
3.14.1 Obtaining Background Information 127
3.14.2 Defining the Semantics of the User Interface 127
3.14.3 Defining the Syntax and Format of the Interface 128
3.14.4 Choosing the Physical Devices 128
3.14.5 Developing the Software 129
3.14.6 Integrating and Installing the System 129
3.14.7 Providing On-line and Product Support 129
3.15 Documenting the User Interface Design 129
3.16 Evaluating the User Interface Design 130
3.17 Case Study: Designing the User Interface for the Rev-Pro System 132
Summary 136
Checklist of Project Activities 137
Terms and Concepts Introduced 138
Exercises 138

4. Preliminary Design: Designing the Software Structure and Data Structures 143

4.1 The Concept of Modularity 144
4.1.1 Modules Should Be Cohesive 147
4.1.2 Modules Should Be Loosely Coupled 148
4.1.3 Modules Should Be Black Boxes 149

4.2 Designing the Software Structure 150
4.2.1 Structure Charts Are Used to Represent the Software Structure 151
4.2.2 Structured Design Uses the Top-down Approach 152
4.3 Using Transform Analysis to Create the Software Structure 156
4.3.1 Steps Involved in First-Level Factoring during Transform Analysis 157
4.3.2 Steps Involved in Lower-level Factoring during Transform Analysis 160
4.3.3 Some Characteristics of Structure Charts Using Transform Analysis 162
4.3.4 Transform Analysis Applied to the Concordance System Example 162
4.3.5 Reviewing the Structure Chart Created during Transform Analysis 166
4.4 Using Transaction Analysis to Derive a Structure Chart 166
4.5 Designing the Data Structures 169
4.5.1 Specification of Procedural Abstractions 170
4.5.2 Specification of Data Abstractions 171
4.5.3 Advantages of Using Abstraction by Specification 173
4.5.4 Some Criteria for Writing Good Specifications 174
4.5.5 Role of Abstraction by Specification in Software Design 174
4.5.6 Pragmatic Concerns in Designing Data Strucutres 175
4.6 Creating the Preliminary Design Document 176
4.6.1 Contents of the Design Description Section 177
4.6.2 Contents of the Module Designs Section 179
4.7 Evaluating the Preliminary Design 182
4.8 Case Study: Preliminary Design of the Rev-Pro System 183
Summary 187
Checklist of Project Activities 188
Terms and Concepts Introduced 189
Exercises 190

5. Detailed Design and Choosing a Programming Language 192

5.1 Selecting an Algorithm 193
5.1.1 Choosing a Correct Algorithm 193
5.1.2 Choosing an Efficient Algorithm 194
5.1.3 Choosing an Appropriate Algorithm 196

5.2 When Efficiency Is Particularly Important 197
5.3 Documenting the Algorithm Design 197
5.3.1 Representing Detailed Design with Nassi–Shneiderman Charts 198
5.3.2 Representing Detailed Design Using Pseudocode 202
5.4 Data Structure Design Revisited 203
5.5 Detailed Design Document 204
5.6 Evaluating the Detailed Design 205
5.7 Choosing a Programming Language: Pragmatic Concerns 206
5.8 Choosing a Programming Language: Programming-in-the-Small Concerns 207
5.8.1 Power and Suitability 208
5.8.2 Simplicity, Clarity, and Orthogonality 208
5.8.3 Language Syntax 210
5.8.4 Control Structures 212
5.8.5 Data Types and Strong Versus Weak Typing 216
5.8.6 Simple Data Types 217
5.8.7 Structured Data Types 218
5.8.8 Constants 220
5.8.9 Procedural Abstraction 221
5.8.10 Scoping and Information Hiding 222
5.9 Choosing a Programming Language: Programming-in-the-Large Concerns 224
5.9.1 Procedural and Data Abstraction 224
5.9.2 Generics 229
5.9.3 Separate Compilation 229
5.10 Case Study: Detailed Design and Language Choice for the Rev-Pro System 230
Summary 233
Checklist of Project Activities 234
Terms and Concepts Introduced 235
Exercises 235

6. Coding and Integration 239

6.1 Coding and Testing Are Carried on in Parallel 239
6.2 Some Alternative Orders of Module Integration 240
6.2.1 Big Bang Integration Method 241
6.2.2 Top-down Integration Method 243
6.2.3 Bottom-up Integration Method 243
6.2.4 Integration Using Threads 244
6.2.5 Example Using the Threads Approach 244
6.3 Producing and Documenting the Integration Plan 247

6.4 Factors That Affect the Quality of the Source Code 248
6.5 Structured Coding Techniques 249
6.6 Coding Style 251
6.6.1 Style Guideline 1: Shorter Is Simpler 252
6.6.2 Style Guideline 2: Fewer Decisions Are Better 255
6.6.3 Style Guideline 3: Nested Logic Should Be Avoided 255
6.7 Data Structure Considerations 258
6.8 Source Code Documentation 258
6.8.1 Program Header Comment 259
6.8.2 Module Headers as Comments 260
6.8.3 Line Comments 261
6.9 Source Code Format and Identifier Naming 263
6.10 Importance of Coding Standards 265
6.11 Case Study: Coding and Integrating the Rev-Pro System 265
Summary 269
Checklist of Project Activities 270
Terms and Concepts Introduced 271
Exercises 271

7. Testing 274
7.1 Testing Defined 274
7.2 Levels of Testing 276
7.3 Principles of Testing 277
7.4 Testing in the Small 279
7.5 White-box Testing in the Small 280
7.5.1 Representing Module Structure Using Logic-flow Diagrams 282
7.5.2 White-box Testing Is Achieved by Covering all Arcs in a Logic-flow Diagram 286
7.5.3 Why White-box Testing Cannot Be Complete 287
7.5.4 Carrying Out White-box Testing 288
7.6 Black-box Testing in the Small 292
7.6.1 Why Black-box Testing Cannot Be Complete 295
7.6.2 Carrying Out the Unit Black-box Tests 296
7.7 Testing in the Small: Data-structure-based Testing 297
7.8 Problem of Error-prone Modules 299
7.9 Testing in the Large 300
7.9.1 Testing in the Large: Integration Testing 300
7.9.2 Documenting Integration Testing 306
7.9.3 Testing in the Large: Acceptance Testing 308

7.10 Case Study: Testing the Rev-Pro System 309
Summary 313
Checklist of Project Activities 314
Terms and Concepts Introduced 316
Exercises 316

8. More about Software Engineering: Now and in the Future 319

8.1 Historical Influences and Trends in Software Engineering 319
8.2 Other Analysis and Design Methods 322
8.2.1 Structured Analysis and Design Technique 322
8.2.2 Warnier-Orr: A Data-structure-oriented Design Method 325
8.2.3 Jackson System of Programming 328
8.3 Software Maintenance 334
8.3.1 The Cost of Software Maintenance 335
8.3.2 Configuration Management: Managerial Concerns 336
8.3.3 Configuration Management: Technical Issues 337
8.3.4 A Special Configuration Management Problem: Multiple Releases 338
8.3.5 The Role of the Maintenance Programmer 339
8.4 Automating the Software Engineering Endeavor 340
8.4.1 CASE Tools: Framing Environments 343
8.4.2 CASE Tools: Programming Environments 346
8.4.3 CASE Tools: General Environments 349
8.4.4 An Example of a General CASE Environment: UNIX 350
8.4.5 Second Example of a General CASE Environment: APSE 352
8.4.6 Third Example of a General CASE Environment: Teamwork 354
8.4.7 Status of CASE 357
8.5 User Interface Management Systems 358
8.5.1 X Window: A Communication System to Support UIMSs 360
8.6 Object-Oriented Development 363
8.6.1 Object-oriented Design 364
8.6.2 Object-oriented Programming 368
8.6.3 Applying the Object-oriented Approach 370
8.7 Current Status of Software Engineering 371

8.8 Future of Software Engineering 374
8.9 Finishing Up the Project: System Delivery and the Operational/Maintenance Phase 378
Summary 380
Checklist of Project Activities 381
Terms and Concepts Introduced 382
Exercises 382

APPENDICES

A. Sample Documents from the Rev-Pro Case Study 385
Part I: Requirements and Specification Document Excerpts 386
Part II: User Interface Design Document Excerpts 404

B. Writing a User's Manual, *Katherine Nell Macfarlane* **408**

References 415

Index 425