



# The Unified Software Development Process

Ivar Jacobson

Grady Booch

James Rumbaugh

*Rational Software Corporation*



**ADDISON-WESLEY**

**An Imprint of Addison Wesley Longman, Inc.**

Reading, Massachusetts • Harlow, England • Menlo Park, California  
Berkeley, California • Don Mills, Ontario • Sydney  
Bonn • Amsterdam • Tokyo • Mexico City

---

---

# Contents

<b>Preface</b>	<b>xvii</b>
----------------	-------------

<b>Part I: The Unified Software Development Process</b>	<b>1</b>
---	----------

<b>Chapter 1: The Unified Process: Use-Case Driven, Architecture-Centric, Iterative, and Incremental</b>	<b>3</b>
--	----------

1.1	The Unified Process in a Nutshell	4
1.2	The Unified Process Is Use-Case Driven	5
1.3	The Unified Process Is Architecture-Centric	6
1.4	The Unified Process Is Iterative and Incremental	7
1.5	The Life of the Unified Process	8
1.5.1	The Product	9
1.5.2	Phases within a Cycle	11
1.6	An Integrated Process	13

## **Chapter 2: The Four Ps: People, Project, Product, and Process in Software Development 15**

- 2.1 People Are Crucial 16**
  - 2.1.1 Development Processes Affect People 16
  - 2.1.2 Roles Will Change 17
  - 2.1.3 Turning “Resources” into “Workers” 18
- 2.2 Projects Make the Product 19**
- 2.3 Product Is More Than Code 20**
  - 2.3.1 What Is a Software System? 20
  - 2.3.2 Artifacts 21
  - 2.3.3 A System Has a Collection of Models 21
  - 2.3.4 What Is a Model? 22
  - 2.3.5 Each Model Is a Self-Contained View of the System 22
  - 2.3.6 Inside a Model 23
  - 2.3.7 Relationships between Models 23
- 2.4 Process Directs Projects 24**
  - 2.4.1 Process: A Template 24
  - 2.4.2 Related Activities Make Up Workflows 25
  - 2.4.3 Specializing Process 26
  - 2.4.4 Merits of Process 27
- 2.5 Tools Are Integral to Process 28**
  - 2.5.1 Tools Impact Process 28
  - 2.5.2 Process Drives Tools 28
  - 2.5.3 Balance Process and Tools 29
  - 2.5.4 Visual Modeling Supports UML 29
  - 2.5.5 Tools Support the Whole Life Cycle 30
- 2.6 References 31**

## **Chapter 3: A Use-Case–Driven Process 33**

- 3.1 Use-Case–Driven Development in Brief 35**
- 3.2 Why Use Cases? 37**
  - 3.2.1 To Capture the Value Adding Requirements 37
  - 3.2.2 To Drive the Process 38
  - 3.2.3 To Devise the Architecture and More... 39
- 3.3 Capturing the Use Cases 40**
  - 3.3.1 The Use-Case Model Represents the Functional Requirements 40
  - 3.3.2 Actors Are the Environment of the System 41
  - 3.3.3 Use Cases Specify the System 41
- 3.4 Analysis, Design, and Implementation to Realize the Use Cases 42**
  - 3.4.1 Creating the Analysis Model from the Use Cases 43
  - 3.4.2 Each Class Must Fulfill All Its Collaboration Roles 48

3.4.3	Creating the Design Model from the Analysis Model	48
3.4.4	Subsystems Group Classes	51
3.4.5	Creating the Implementation Model from the Design Model	53
<b>3.5</b>	<b>Testing the Use Cases</b>	<b>55</b>
<b>3.6</b>	<b>Summing Up</b>	<b>57</b>
<b>3.7</b>	<b>References</b>	<b>57</b>

## **Chapter 4: An Architecture-Centric Process    59**

<b>4.1</b>	<b>Architecture in Brief</b>	<b>60</b>
<b>4.2</b>	<b>Why We Need Architecture</b>	<b>62</b>
4.2.1	Understanding the System	62
4.2.2	Organizing Development	63
4.2.3	Fostering Reuse	63
4.2.4	Evolving the System	64
<b>4.3</b>	<b>Use Cases and Architecture</b>	<b>65</b>
<b>4.4</b>	<b>The Steps to an Architecture</b>	<b>69</b>
4.4.1	The Architecture Baseline Is a “Small, Skinny” System	69
4.4.2	Using Architecture Patterns	71
4.4.3	Describing Architecture	74
4.4.4	The Architect Creates the Architecture	76
<b>4.5</b>	<b>Finally, an Architecture Description!</b>	<b>77</b>
4.5.1	The Architectural View of the Use-Case Model	78
4.5.2	The Architectural View of the Design Model	78
4.5.3	The Architectural View of the Deployment Model	81
4.5.4	The Architectural View of the Implementation Model	83
<b>4.6</b>	<b>Three Interesting Concepts</b>	<b>83</b>
4.6.1	What Is Architecture?	83
4.6.2	How Is It Obtained?	83
4.6.3	How Is It Described?	83
<b>4.7</b>	<b>References</b>	<b>84</b>

## **Chapter 5: An Iterative and Incremental Process    85**

<b>5.1</b>	<b>Iterative and Incremental in Brief</b>	<b>86</b>
5.1.1	Develop in Small Steps	87
5.1.2	What Iteration Is Not	88
<b>5.2</b>	<b>Why Iterative and Incremental Development?</b>	<b>89</b>
5.2.1	Mitigating Risks	89
5.2.2	Getting a Robust Architecture	91
5.2.3	Handling Changing Requirements	91
5.2.4	Allowing for Tactical Changes	92
5.2.5	Achieving Continuous Integration	92
5.2.6	Attaining Early Learning	93

<b>5.3</b>	<b>The Iterative Approach is Risk-Driven</b>	<b>94</b>
5.3.1	Iterations Alleviate Technical Risks	95
5.3.2	Management Is Responsible for Nontechnical Risks	97
5.3.3	Dealing with Risks	97
<b>5.4</b>	<b>The Generic Iteration</b>	<b>98</b>
5.4.1	What an Iteration Is	98
5.4.2	Planning the Iterations	100
5.4.3	Sequencing the Iterations	100
<b>5.5</b>	<b>The Result of an Iteration Is an Increment</b>	<b>101</b>
<b>5.6</b>	<b>Iterations over the Life Cycle</b>	<b>102</b>
<b>5.7</b>	<b>Models Evolve from Iterations</b>	<b>105</b>
<b>5.8</b>	<b>Iterations Challenge the Organization</b>	<b>106</b>
<b>5.9</b>	<b>References</b>	<b>106</b>

## **Part II: The Core Workflows 109**

### **Chapter 6: Requirements Capture: From Vision to Requirements 111**

<b>6.1</b>	<b>Why Requirements Capture Is Difficult</b>	<b>112</b>
<b>6.2</b>	<b>The Purpose of the Requirements Workflow</b>	<b>113</b>
<b>6.3</b>	<b>Overview of Requirements Capture</b>	<b>113</b>
<b>6.4</b>	<b>The Role of Requirements in the Software Life Cycle</b>	<b>118</b>
<b>6.5</b>	<b>Understanding the System Context Using a Domain Model</b>	<b>119</b>
6.5.1	What Is a Domain Model?	119
6.5.2	Developing a Domain Model	121
6.5.3	Use of the Domain Model	121
<b>6.6</b>	<b>Understanding the System Context Using a Business Model</b>	<b>122</b>
6.6.1	What Is a Business Model?	122
6.6.2	How to Develop a Business Model	124
6.6.3	Find Use Cases from a Business Model	126
<b>6.7</b>	<b>Supplementary Requirements</b>	<b>128</b>
<b>6.8</b>	<b>Summary</b>	<b>130</b>
<b>6.9</b>	<b>References</b>	<b>130</b>

### **Chapter 7: Capturing the Requirements as Use Cases 131**

<b>7.1</b>	<b>Introduction</b>	<b>131</b>
<b>7.2</b>	<b>Artifacts</b>	<b>133</b>
7.2.1	Artifact: Use-Case Model	133
7.2.2	Artifact: Actor	134
7.2.3	Use Case	135
7.2.4	Artifact: Architecture Description (View of the Use-Case Model)	139

7.2.5	Artifact: Glossary	139
7.2.6	Artifact: User-Interface Prototype	140
<b>7.3</b>	<b>Workers</b>	<b>140</b>
7.3.1	Worker: System Analyst	140
7.3.2	Worker: Use-Case Specifier	141
7.3.3	User-Interface Designer	142
7.3.4	Worker: Architect	142
<b>7.4</b>	<b>Workflow</b>	<b>143</b>
7.4.1	Activity: Find Actors and Use Cases	144
7.4.2	Activity: Prioritize Use Cases	153
7.4.3	Activity: Detail a Use Case	153
7.4.4	Activity: Prototype User Interface	160
7.4.5	Activity: Structure the Use-Case Model	166
<b>7.5</b>	<b>Summary of the Requirements Workflow</b>	<b>171</b>
<b>7.6</b>	<b>References</b>	<b>172</b>

## **Chapter 8: Analysis 173**

<b>8.1</b>	<b>Introduction</b>	<b>173</b>
<b>8.2</b>	<b>Analysis in Brief</b>	<b>176</b>
8.2.1	Why Analysis Is not Design or Implementation	176
8.2.2	The Purpose of Analysis: Summary	177
8.2.3	Concrete Examples of When to Employ Analysis	178
<b>8.3</b>	<b>The Role of Analysis in the Software Life Cycle</b>	<b>179</b>
<b>8.4</b>	<b>Artifacts</b>	<b>181</b>
8.4.1	Artifact: Analysis Model	181
8.4.2	Artifact: Analysis Class	181
8.4.3	Artifact: Use-Case Realization—Analysis	186
8.4.4	Artifact: Analysis Package	190
8.4.5	Artifact: Architecture Description (View of the Analysis Model)	193
<b>8.5</b>	<b>Workers</b>	<b>194</b>
8.5.1	Worker: Architect	194
8.5.2	Worker: Use-Case Engineer	194
8.5.3	Worker: Component Engineer	195
<b>8.6</b>	<b>Workflow</b>	<b>196</b>
8.6.1	Activity: Architectural Analysis	196
8.6.2	Activity: Analyze a Use Case	203
8.6.3	Activity: Analyze a Class	207
8.6.4	Activity: Analyze a Package	211
<b>8.7</b>	<b>Summary of Analysis</b>	<b>213</b>
<b>8.8</b>	<b>References</b>	<b>214</b>

## **Chapter 9: Design 215**

- 9.1 Introduction 215**
- 9.2 The Role of Design in the Software Life Cycle 216**
- 9.3 Artifacts 217**
  - 9.3.1 Artifact: Design Model 217
  - 9.3.2 Artifact: Design Class 218
  - 9.3.3 Artifact: Use-Case Realization—Design 221
  - 9.3.4 Artifact: Design Subsystem 224
  - 9.3.5 Artifact: Interface 226
  - 9.3.6 Artifact: Architecture Description (View of the Design Model) 226
  - 9.3.7 Artifact: Deployment Model 227
  - 9.3.8 Artifact: Architecture Description (View of the Deployment Model) 228
- 9.4 Workers 229**
  - 9.4.1 Worker: Architect 229
  - 9.4.2 Worker: Use-Case Engineer 230
  - 9.4.3 Worker: Component Engineer 230
- 9.5 Workflow 231**
  - 9.5.1 Activity: Architectural Design 232
  - 9.5.2 Activity: Design a Use Case 249
  - 9.5.3 Activity: Design a Class 255
  - 9.5.4 Activity: Design a Subsystem 263
- 9.6 Summary of Design 265**
- 9.7 References 266**

## **Chapter 10: Implementation 267**

- 10.1 Introduction 267**
- 10.2 The Role of Implementation in the Software Life Cycle 268**
- 10.3 Artifacts 269**
  - 10.3.1 Artifact: Implementation Model 269
  - 10.3.2 Artifact: Component 269
  - 10.3.3 Artifact: Implementation Subsystem 272
  - 10.3.4 Artifact: Interface 274
  - 10.3.5 Artifact: Architecture Description (View of the Implementation Model) 275
  - 10.3.6 Artifact: Integration Build Plan 276
- 10.4 Workers 277**
  - 10.4.1 Worker: Architect 277
  - 10.4.2 Worker: Component Engineer 277
  - 10.4.3 Worker: System Integrator 279
- 10.5 Workflow 279**
  - 10.5.1 Activity: Architectural Implementation 280
  - 10.5.2 Activity: Integrate System 283

10.5.3	Activity: Implement a Subsystem	285
10.5.4	Activity: Implement a Class	288
10.5.5	Activity: Perform Unit Test	289
10.6	Summary of Implementation	293
10.7	References	293

## **Chapter 11: Test 295**

11.1	Introduction	295
11.2	The Role of Testing in the Software Life Cycle	296
11.3	Artifacts	297
11.3.1	Artifact: Test Model	297
11.3.2	Artifact: Test Case	297
11.3.3	Artifact: Test Procedure	300
11.3.4	Artifact: Test Component	302
11.3.5	Artifact: Plan Test	302
11.3.6	Artifact: Defect	302
11.3.7	Artifact: Evaluate Test	302
11.4	Workers	303
11.4.1	Worker: Test Designer	303
11.4.2	Worker: Component Engineer	303
11.4.3	Worker: Integration Tester	303
11.4.4	Worker: System Tester	304
11.5	Workflow	304
11.5.1	Activity: Plan Test	305
11.5.2	Activity: Design Test	306
11.5.3	Activity: Implement Test	309
11.5.4	Activity: Perform Integration Test	310
11.5.5	Activity: Perform System Test	311
11.5.6	Activity: Evaluate Test	311
11.6	Summary of Testing	313
11.7	References	313

## **Part III: Iterative and Incremental Development 315**

### **Chapter 12: The Generic Iteration Workflow 317**

12.1	The Need for Balance	318
12.2	The Phases Are the First Division of Work	319
12.2.1	Inception Phase Establishes Feasibility	319
12.2.2	Elaboration Phase Focuses on "Do-Ability"	320
12.2.3	Construction Phase Builds the System	321
12.2.4	Transition Phase Moves into the User Environment	322



<b>12.3 The Generic Iteration Revisited</b>	<b>322</b>
12.3.1 Core Workflows Repeat in Each Iteration	322
12.3.2 Workers Participate in the Workflows	323
<b>12.4 Planning Precedes Doing</b>	<b>324</b>
12.4.1 Plan the Four Phases	325
12.4.2 Plan the Iterations	326
12.4.3 Think Long Term	327
12.4.4 Plan the Evaluation Criteria	327
<b>12.5 Risks Affect Project Planning</b>	<b>328</b>
12.5.1 Manage a Risk List	328
12.5.2 Risks Affect the Iteration Plan	329
12.5.3 Schedule Risk Action	329
<b>12.6 Use-Case Prioritization</b>	<b>330</b>
12.6.1 Risks Specific to a Particular Product	331
12.6.2 Risk of Not Getting the Architecture Right	331
12.6.3 Risk of Not Getting Requirements Right	332
<b>12.7 Resources Needed</b>	<b>333</b>
12.7.1 Projects Differ Widely	334
12.7.2 A Typical Project Looks Like This	335
12.7.3 Complex Projects Have Greater Needs	335
12.7.4 New Product Line Calls for Experience	336
12.7.5 Paying the Cost of the Resources Used	337
<b>12.8 Assess the Iterations and Phases</b>	<b>338</b>
12.8.1 Criteria Not Achieved	338
12.8.2 The Criteria Themselves	339
12.8.3 The Next Iteration	339
12.8.4 Evolution of the Model Set	340

## **Chapter 13: Inception Launches the Project 341**

<b>13.1 The Inception Phase in Brief</b>	<b>341</b>
<b>13.2 Early in the Inception Phase</b>	<b>342</b>
13.2.1 Before the Inception Phase Begins	342
13.2.2 Planning the Inception Phase	343
13.2.3 Expanding the System Vision	344
13.2.4 Setting the Evaluation Criteria	344
<b>13.3 The Archetypal Inception Iteration Workflow</b>	<b>346</b>
13.3.1 Introduction to the Five Core Workflows	346
13.3.2 Fitting the Project into the Development Environment	348
13.3.3 Finding Critical Risks	348
<b>13.4 Execute the Core Workflows, Requirements to Test</b>	<b>348</b>
13.4.1 Capture the Requirements	350
13.4.2 Analysis	352

13.4.3 Design	353
13.4.5 Test	354
<b>13.5 Make the Initial Business Case</b>	<b>354</b>
13.5.1 Outline Business Bid	354
13.5.2 Estimate Return on Investment	356
<b>13.6 Assess the Iteration(s) in the Inception Phase</b>	<b>356</b>
<b>13.7 Planning the Elaboration Phase</b>	<b>357</b>
<b>13.8 The Deliverables for the Inception Phase</b>	<b>358</b>

## **Chapter 14: The Elaboration Phase Makes the Architectural Baseline 359**

<b>14.1 The Elaboration Phase in Brief</b>	<b>359</b>
<b>14.2 Early in the Elaboration Phase</b>	<b>360</b>
14.2.1 Planning the Elaboration Phase	361
14.2.2 Building the Team	361
14.2.3 Modifying the Development Environment	361
14.2.4 Setting Evaluation Criteria	361
<b>14.3 The Archetypal Elaboration Iteration Workflow</b>	<b>362</b>
14.3.1 Capture and Refine Most of the Requirements	363
14.3.2 Develop the Architectural Baseline	364
14.3.3 Iterate While the Team Is Small	364
<b>14.4 Execute the Core Workflows—Requirements to Test</b>	<b>364</b>
14.4.1 Capture the Requirements	365
14.4.2 Analysis	367
14.4.3 Design	372
14.4.4 Implementation	374
14.4.5 Test	376
<b>14.5 Make the Business Case</b>	<b>377</b>
14.5.1 Prepare the Business Bid	378
14.5.2 Update Return on Investment	378
<b>14.6 Assess the Iterations in the Elaboration Phase</b>	<b>378</b>
<b>14.7 Planning the Construction Phase</b>	<b>379</b>
<b>14.8 The Key Deliverables</b>	<b>380</b>

## **Chapter 15: Construction Leads to Initial Operational Capability 381**

<b>15.1 The Construction Phase in Brief</b>	<b>382</b>
<b>15.2 Early in the Construction Phase</b>	<b>382</b>
15.2.1 Staffing the Phase	383
15.2.2 Setting the Evaluation Criteria	383
<b>15.3 The Archetypal Construction Iteration Workflow</b>	<b>384</b>

<b>15.4 Execute the Core Workflows—Requirements to Testing</b>	<b>385</b>
15.4.1 Requirements	387
15.4.2 Analysis	388
15.4.3 Design	389
15.4.4 Implementation	390
15.4.5 Test	391
<b>15.5 Controlling the Business Case</b>	<b>393</b>
<b>15.6 Assess the Iterations and the Construction Phase</b>	<b>393</b>
<b>15.7 Planning the Transition Phase</b>	<b>393</b>
<b>15.8 The Key Deliverables</b>	<b>394</b>

## **Chapter 16: Transition Completes Product Release      395**

<b>16.1 The Transition Phase in Brief</b>	<b>396</b>
<b>16.2 Early in the Transition Phase</b>	<b>397</b>
16.2.1 Planning the Transition Phase	397
16.2.2 Staffing the Transition Phase	399
16.2.3 Setting the Evaluation Criteria	399
<b>16.3 The Core Workflows Play a Small Role in this Phase</b>	<b>400</b>
<b>16.4 What We Do in the Transition Phase</b>	<b>401</b>
16.4.1 Getting the Beta Release Out	401
16.4.2 Installing the Beta Release	402
16.4.3 Responding to the Test Results	402
16.4.4 Adapting the Product to Varied User Environments	403
16.4.5 Completing the Artifacts	404
16.4.6 When Does the Project End?	404
<b>16.5 Completing the Business Case</b>	<b>405</b>
16.5.1 Controlling Progress	405
16.5.2 Review of the Business Plan	405
<b>16.6 Assess the Transition Phase</b>	<b>406</b>
16.6.1 Assess the Iterations and the Phase	406
16.6.2 Postmortem of the Project	407
<b>16.7 Planning the Next Release or Generation</b>	<b>407</b>
<b>16.8 The Key Deliverables</b>	<b>407</b>

## **Chapter 17: Making the Unified Process Work      409**

<b>17.1 The Unified Process Helps You Deal with Complexity</b>	<b>409</b>
17.1.1 The Life Cycle Objectives	410
17.1.2 The Life Cycle Architecture	410
17.1.3 Initial Operational Capability	411
17.1.4 Product Release	411

<b>17.2 The Major Themes</b>	<b>411</b>
<b>17.3 Management Leads Conversion to Unified Process</b>	<b>412</b>
17.3.1 The Case for Action	413
17.3.2 The Reengineering Directive Persuades	413
17.3.3 Implementing the Transition	414
<b>17.4 Specializing the Unified Process</b>	<b>416</b>
17.4.1 Tailoring the Process	416
17.4.2 Filling in the Process Framework	417
<b>17.5 Relate to the Broader Community</b>	<b>418</b>
<b>17.6 Get the Benefits of the Unified Process</b>	<b>418</b>
<b>17.7 References</b>	<b>419</b>

## **Appendix A: Overview of the UML**      **421**

<b>A.1 Introduction</b>	<b>421</b>
A.1.1 Vocabulary	422
A.1.2 Extensibility Mechanisms	422
<b>A.2 Graphical Notation</b>	<b>423</b>
A.2.1 Structural Things	423
A.2.2 Behavioral Things	424
A.2.3 Grouping Things	425
A.2.4 Annotational Things	425
A.2.5 Dependency Relationships	425
A.2.6 Association Relationships	425
A.2.7 Generalization Relationships	426
A.2.8 Extensibility Mechanisms	426
<b>A.3 Glossary of Terms</b>	<b>426</b>
<b>A.4 References</b>	<b>433</b>

## **Appendix B: The Unified Process-Specific Extensions of the UML**      **435**

<b>B.1 Introduction</b>	<b>435</b>
<b>B.2 Stereotypes</b>	<b>435</b>
<b>B.3 Tagged Values</b>	<b>438</b>
<b>B.4 Graphical Notation</b>	<b>439</b>
<b>B.5 References</b>	<b>439</b>

## **Appendix C: General Glossary**      **441**

<b>C.1 Introduction</b>	<b>441</b>
<b>C.2 Terms</b>	<b>441</b>

## **Index**      **451**