## Contents

**Foreword** xvii

**Preface** xxii

### Part I  Fundamentals of Requirements Engineering  1

1 Setting the Scene  3

1.1 What is requirements engineering?  3

1.1.1 The problem world and the machine solution  4

1.1.2 Introducing our running case studies  6

1.1.3 The WHY, WHAT and WHO dimensions of requirements engineering  12

1.1.4 Types of statements involved in requirements engineering  17

1.1.5 Categories of requirements  23

1.1.6 The requirements lifecycle: Processes, actors and products  30

1.1.7 Target qualities and defects to avoid  35

1.1.8 Types of software projects  40

1.1.9 Requirements in the software lifecycle  42

1.1.10 The relationship of requirements engineering to other disciplines  45

1.2 Why engineer requirements?  47

1.2.1 Facts, data and citations about the requirements problem  47

1.2.2 The role and stakes of requirements engineering  51

1.3 Obstacles to good requirements engineering practice  52

1.4 Agile development processes and requirements engineering  53

Summary  55

Notes and Further Reading  56

Exercises  58
## Domain Understanding and Requirements Elicitation

### 2.1 Identifying stakeholders and interacting with them

### 2.2 Artefact-driven elicitation techniques
- **2.2.1 Background study**
- **2.2.2 Data collection**
- **2.2.3 Questionnaires**
- **2.2.4 Repertory grids and card sorts for concept-driven acquisition**
- **2.2.5 Storyboards and scenarios for problem world exploration**
- **2.2.6 Mock-ups and prototypes for early feedback**
- **2.2.7 Knowledge reuse**

### 2.3 Stakeholder-driven elicitation techniques
- **2.3.1 Interviews**
- **2.3.2 Observation and ethnographic studies**
- **2.3.3 Group sessions**

### 2.4 Conclusion
- **Summary**
- **Notes and Further Reading**
- **Exercises**

## Requirements Evaluation

### 3.1 Inconsistency management
- **3.1.1 Types of inconsistency**
- **3.1.2 Handling inconsistencies**
- **3.1.3 Managing conflicts: A systematic process**

### 3.2 Risk analysis
- **3.2.1 Types of risk**
- **3.2.2 Risk management**
- **3.2.3 Risk documentation**
- **3.2.4 Integrating risk management in the requirements lifecycle**

### 3.3 Evaluating alternative options for decision making

### 3.4 Requirements prioritization

### 3.5 Conclusion
- **Summary**
- **Notes and Further Reading**
- **Exercises**

## Requirements Specification and Documentation

### 4.1 Free documentation in unrestricted natural language

### 4.2 Disciplined documentation in structured natural language
- **4.2.1 Local rules on writing statements**
- **4.2.2 Global rules on organizing the requirements document**
### 4.3 Use of diagrammatic notations

- **4.3.1 System scope: context, problem and frame diagrams**
- **4.3.2 Conceptual structures: entity–relationship diagrams**
- **4.3.3 Activities and data: SADT diagrams**
- **4.3.4 Information flows: dataflow diagrams**
- **4.3.5 System operations: use case diagrams**
- **4.3.6 Interaction scenarios: event trace diagrams**
- **4.3.7 System behaviours: state machine diagrams**
- **4.3.8 Stimuli and responses: R-net diagrams**
- **4.3.9 Integrating multiple system views and multiview specification in UML**
- **4.3.10 Diagrammatic notations: Strengths and limitations**

### 4.4 Formal specification

- **4.4.1 Logic as a basis for formalizing statements**
- **4.4.2 History-based specification**
- **4.4.3 State-based specification**
- **4.4.4 Event-based specification**
- **4.4.5 Algebraic specification**
- **4.4.6 Other specification paradigms**
- **4.4.7 Formal specification: strengths and limitations**

### 4.5 Conclusion

- Summary
- Notes and Further Reading
- Exercises

---

### 5 Requirements Quality Assurance

- **5.1 Requirements inspections and reviews**
  - **5.1.1 The requirements inspection process**
  - **5.1.2 Inspection guidelines**
  - **5.1.3 Requirements inspection checklists**
  - **5.1.4 Conclusion**

- **5.2 Queries on a requirements database**

- **5.3 Requirements validation by specification animation**
  - **5.3.1 Extracting an executable model from the specification**
  - **5.3.2 Simulating the model**
  - **5.3.3 Visualizing the simulation**
  - **5.3.4 Conclusion**

- **5.4 Requirements verification through formal checks**
  - **5.4.1 Language checks**
  - **5.4.2 Dedicated consistency and completeness checks**
  - **5.4.3 Model checking**
  - **5.4.4 Theorem proving**
### Contents

5.5 Conclusion 211  
Summary 213  
Notes and Further Reading 214  
Exercises 217

6 **Requirements Evolution** 219  
6.1 The time–space dimensions of evolution: Revisions and variants 220  
6.2 Change anticipation 223  
6.3 Traceability management for evolution support 225  
6.3.1 Traceability links 226  
6.3.2 The traceability management process, its benefits and cost 233  
6.3.3 Traceability management techniques 237  
6.3.4 Determining an adequate cost–benefit trade-off for traceability management 244  
6.4 Change control 246  
6.4.1 Change initiation 247  
6.4.2 Change evaluation and prioritization 248  
6.4.3 Change consolidation 249  
6.5 Runtime monitoring of requirements and assumptions for dynamic change 249  
6.6 Conclusion 251  
Summary 252  
Notes and Further Reading 254  
Exercises 256

7 **Goal Orientation in Requirements Engineering** 259  
7.1 What are goals? 260  
7.2 The granularity of goals and their relationship to requirements and assumptions 261  
7.3 Goal types and categories 265  
7.3.1 Types of goal: behavioural goals vs soft goals 265  
7.3.2 Goal categories: Functional vs non-functional goals 269  
7.4 The central role of goals in the requirements engineering process 272  
7.5 Where are goals coming from? 275  
7.6 The relationship of goals to other requirements-related products and processes 276  
7.6.1 Goals and scenarios 276  
7.6.2 Intentional and operational specifications 277  
7.6.3 Goals and use cases 277  
7.6.4 Goals and model-checked properties 277  
7.6.5 Goal orientation and agent orientation 278  
7.6.6 Goal orientation and object orientation 278  
7.6.7 Goal orientation and top-down analysis 279
10 Modelling Conceptual Objects with Class Diagrams 359

10.1 Representing domain concepts by conceptual objects 360
  10.1.1 What are conceptual objects? 360
  10.1.2 Object instantiation: classes and current instances 361
  10.1.3 Types of conceptual object 362
  10.1.4 Object models as UML class diagrams 363
  10.1.5 Object features as model annotations 364

10.2 Entities 366

10.3 Associations 366

10.4 Attributes 371

10.5 Built-in associations for structuring object models 373
  10.5.1 Object specialization 373
  10.5.2 Object aggregation 376

10.6 More on class diagrams 377
  10.6.1 Derived attributes and associations 377
  10.6.2 OR-associations 378
  10.6.3 Ordered associations 379
  10.6.4 Associations of associations 379

10.7 Heuristic rules for building object models 377
  10.7.1 Deriving pertinent and complete class diagrams from goal diagrams 380
  10.7.2 Object or attribute? 384
  10.7.3 Entity, association, agent or event? 384
  10.7.4 Attribute of a linked object or of the linking association? 385
  10.7.5 Aggregation or association? 386
  10.7.6 Specializing and generalizing concepts 386
  10.7.7 Avoiding common pitfalls 387

Summary 389

Notes and Further Reading 391
Exercises 392

11 Modelling System Agents and Responsibilities 395

11.1 What are agents? 396

11.2 Characterizing system agents 397
  11.2.1 Basic features 397
  11.2.2 Agent capabilities 397
  11.2.3 Agent responsibilities and goal realizability 399
  11.2.4 Agents as operation performers 401
  11.2.5 Agent wishes and beliefs 402
  11.2.6 Agent dependencies 403
11.3 Representing agent models 405
   11.3.1 Agent diagrams and instance declarations 405
   11.3.2 Context diagrams 406
   11.3.3 Dependency diagrams 407
11.4 Refinement of abstract agents 408
11.5 Building agent models 411
   11.5.1 Heuristics for building agent diagrams from goal models 411
   11.5.2 Generating context diagrams from goal models 413
Summary 415
Notes and Further Reading 417
Exercises 418

12 Modelling System Operations 421
12.1 What are operations? 422
12.2 Characterizing system operations 425
   12.2.1 Basic features 425
   12.2.2 Operation signature 425
   12.2.3 Domain pre- and post-conditions 426
   12.2.4 Operation performer 427
12.3 Goal operationalization 427
   12.3.1 Required pre-, post- and trigger conditions for goal satisfaction 427
   12.3.2 Agent commitments 430
   12.3.3 Goal operationalization and satisfaction arguments 432
12.4 Goals, agents, objects and operations: The semantic picture 434
12.5 Representing operation models 435
   12.5.1 Operationalization diagrams 435
   12.5.2 UML use case diagrams 435
12.6 Building operation models 437
   12.6.1 Heuristics for building operationalization diagrams 437
   12.6.2 Generating use case diagrams from operationalization diagrams 442
Summary 442
Notes and Further Reading 444
Exercises 445

13 Modelling System Behaviours 449
13.1 Modelling instance behaviours 450
   13.1.1 Scenarios as UML sequence diagrams 450
   13.1.2 Scenario refinement: Episodes and agent decomposition 452
13.2 Modelling class behaviours 454
   13.2.1 State machines as UML state diagrams 455
   13.2.2 State machine refinement: Sequential and concurrent sub-states 459
13.3 Building behaviour models 463
   13.3.1 Elaborating relevant scenarios for good coverage 465
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.3.2 Decorating scenarios with state conditions</td>
<td>467</td>
</tr>
<tr>
<td>13.3.3 From scenarios to state machines</td>
<td>469</td>
</tr>
<tr>
<td>13.3.4 From scenarios to goals</td>
<td>473</td>
</tr>
<tr>
<td>13.3.5 From operationalized goals to state machines</td>
<td>475</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>477</td>
</tr>
<tr>
<td><strong>Notes and Further Reading</strong></td>
<td>480</td>
</tr>
<tr>
<td><strong>Exercises</strong></td>
<td>481</td>
</tr>
<tr>
<td>14 Integrating Multiple System Views</td>
<td>485</td>
</tr>
<tr>
<td>14.1 A meta-model for view integration</td>
<td>485</td>
</tr>
<tr>
<td>14.1.1 Overall structure of the meta-model</td>
<td>487</td>
</tr>
<tr>
<td>14.1.2 The goal meta-model</td>
<td>488</td>
</tr>
<tr>
<td>14.1.3 The object meta-model</td>
<td>489</td>
</tr>
<tr>
<td>14.1.4 The agent meta-model</td>
<td>490</td>
</tr>
<tr>
<td>14.1.5 The operation meta-model</td>
<td>491</td>
</tr>
<tr>
<td>14.1.6 The behaviour meta-model</td>
<td>492</td>
</tr>
<tr>
<td>14.2 Inter-view consistency rules</td>
<td>493</td>
</tr>
<tr>
<td>14.3 Grouping related view fragments into packages</td>
<td>496</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>498</td>
</tr>
<tr>
<td><strong>Notes and Further Reading</strong></td>
<td>498</td>
</tr>
<tr>
<td><strong>Exercises</strong></td>
<td>499</td>
</tr>
<tr>
<td>15 A Goal-Oriented Model-Building Method in Action</td>
<td>501</td>
</tr>
<tr>
<td>15.1 Modelling the system-as-is</td>
<td>503</td>
</tr>
<tr>
<td>15.1.1 Step 1: Build a preliminary goal model illustrated by scenarios</td>
<td>503</td>
</tr>
<tr>
<td>15.1.2 Step 2: Derive a preliminary object model</td>
<td>506</td>
</tr>
<tr>
<td>15.2 Modelling the system-to-be</td>
<td>507</td>
</tr>
<tr>
<td>15.2.1 Step 3: Update the goal model with new goals illustrated by scenarios</td>
<td>507</td>
</tr>
<tr>
<td>15.2.2 Step 4: Derive the updated object model</td>
<td>510</td>
</tr>
<tr>
<td>15.2.3 Step 5: Analyse obstacles, threats and conflicts</td>
<td>512</td>
</tr>
<tr>
<td>15.2.4 Step 6: Analyse responsibilities and build the agent model</td>
<td>515</td>
</tr>
<tr>
<td>15.2.5 Step 7: Make choices among alternative options</td>
<td>517</td>
</tr>
<tr>
<td>15.2.6 Step 8: Operationalize goals in the operation model</td>
<td>518</td>
</tr>
<tr>
<td>15.2.7 Step 9: Build and analyse the behaviour model</td>
<td>521</td>
</tr>
<tr>
<td>15.3 Handling model variants for product lines</td>
<td>524</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>528</td>
</tr>
<tr>
<td><strong>Notes and Further Reading</strong></td>
<td>529</td>
</tr>
<tr>
<td><strong>Exercises</strong></td>
<td>529</td>
</tr>
<tr>
<td>Part III Reasoning About System Models</td>
<td>535</td>
</tr>
<tr>
<td>16 Semi-Formal Reasoning for Model Analysis and Exploitation</td>
<td>537</td>
</tr>
<tr>
<td>16.1 Query-based analysis of the model database</td>
<td>538</td>
</tr>
</tbody>
</table>
16.1 Checking the structural consistency and completeness of the model 538
16.1.2 Generation of other views for dedicated analyses 540
16.1.3 Traceability management 540
16.1.4 Analogical model reuse 541
16.2 Semi-formal analysis of goal-oriented models 544
16.2.1 Conflict analysis 544
16.2.2 Heuristic identification of obstacles 549
16.2.3 Threat analysis: From goal models to anti-goal models 551
16.3 Reasoning about alternative options 557
16.3.1 Qualitative reasoning about alternatives 557
16.3.2 Quantitative reasoning about alternatives 560
16.4 Model-driven generation of the requirements document 562
16.5 Beyond RE: From goal-oriented requirements to software architecture 566
16.5.1 Deriving a software data architecture from the object model 567
16.5.2 Deriving an abstract dataflow architecture from the agent and operation models 568
16.5.3 Selecting an architectural style from architectural requirements 570
16.5.4 Architectural refinement from quality requirements 571
Summary 574
Notes and Further Reading 576
Exercises 578

17 Formal Specification of System Models 583
17.1 A real-time temporal logic for specifying model annotations 584
17.1.1 State assertions 584
17.1.2 Temporal assertions 585
17.1.3 Real-time temporal constructs 586
17.2 Specifying goals in the goal model 588
17.3 Specifying descriptive properties in the object model 592
17.4 Specifying operationalizations in the operation model 594
17.5 Back to the system’s semantic picture 596
Summary 598
Notes and Further Reading 599
Exercises 599

18 Formal Reasoning for Specification Construction and Analysis 603
18.1 Checking goal refinements 604
18.1.1 Using a theorem prover 604
18.1.2 Formal refinement patterns 604
18.1.3 Using bounded SAT solvers 608
18.2 Deriving goal operationalizations 609
18.2.1 Using bounded SAT solvers 610
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.2.2</td>
<td>Formal operationalization patterns</td>
<td>610</td>
</tr>
<tr>
<td>18.3</td>
<td>Generating obstacles for risk analysis</td>
<td>613</td>
</tr>
<tr>
<td>18.3.1</td>
<td>Regressing obstructions through domain properties</td>
<td>614</td>
</tr>
<tr>
<td>18.3.2</td>
<td>Using formal obstruction patterns</td>
<td>617</td>
</tr>
<tr>
<td>18.4</td>
<td>Generating anti-goals for security analysis</td>
<td>618</td>
</tr>
<tr>
<td>18.4.1</td>
<td>Specifying security goals</td>
<td>618</td>
</tr>
<tr>
<td>18.4.2</td>
<td>Identifying security goals and initial anti-goals</td>
<td>620</td>
</tr>
<tr>
<td>18.4.3</td>
<td>Refining anti-goals</td>
<td>621</td>
</tr>
<tr>
<td>18.5</td>
<td>Formal conflict analysis</td>
<td>622</td>
</tr>
<tr>
<td>18.5.1</td>
<td>Deriving boundary conditions for conflict</td>
<td>623</td>
</tr>
<tr>
<td>18.5.2</td>
<td>Formal resolution of divergences</td>
<td>625</td>
</tr>
<tr>
<td>18.6</td>
<td>Synthesizing behaviour models for animation and model checking</td>
<td>627</td>
</tr>
<tr>
<td>18.6.1</td>
<td>Goal-driven model synthesis</td>
<td>628</td>
</tr>
<tr>
<td>18.6.2</td>
<td>Scenario-driven model synthesis</td>
<td>628</td>
</tr>
<tr>
<td>Summary</td>
<td>635</td>
<td></td>
</tr>
<tr>
<td>Notes and Further Reading</td>
<td>636</td>
<td></td>
</tr>
<tr>
<td>Exercises</td>
<td>637</td>
<td></td>
</tr>
</tbody>
</table>

**Bibliography** | 641 |

**Index** | 669 |