Dines Bjørner — bjorner@gmail.com

Guide to the 3-volume book

SOFTWARE ENGINEERING

Vol. 1: Abstraction and Modelling

Vol. 2: Specification of Systems and Languages

Vol. 3: Domains, Requirements and Software Design — in 9 Courses

November 21, 2005

Springer
Berlin Heidelberg New York
Hong Kong London
Milan Paris Tokyo

Why You Want to Lecture over This Book

- **The Texts:**
  - ⋆ Software engineering: the art, craft, discipline, logic, practice and science of developing large scale software products is in increasing need of a trustworthy, believable and professional base.
  - ⋆ This book is devoted to fill this need.
  - ⋆ The present series of strongly related volumes combine informal, engineeringly sound practices with the rigour of formal, mathematics based approaches.
  - ⋆ The three volumes, 2288 pages, are here suggested as the basis for quite a number of different undergraduate and graduate courses. In this document we outline nine (9) courses.
- **Problem Exercises and Solutions:**
  - ⋆ Each volume contains many exercises.
  - ⋆ Already now solutions can be provided to many of these exercises.
  - ⋆ Eventually all exercises will be so supported.
  - ⋆ A number of medium-to-large scale carefully annotated example — current and upcoming — specifications can serve as role models for lecturers and students alike. Helping lecturers to conduct semester projects and students to carry out such projects.
- **249 Lecture Topics and 2884 Slides:**
  - ⋆ The three volumes cover a great many carefully related topics.
  - ⋆ Each topic is supported by may lecture slides.
  - ⋆ These can — as well – be provided to lecturers.
  - ⋆ The present guide hints at combinations of topics into courses.
  - ⋆ Slide texts form a proper subset of volume and chapter texts.
  - ⋆ The following documents list topics and slide numbers:
    - · **Vol. 1: 71 Topics and 1790 Slides:**
      http://www.imm.dtu.dk/~db/Software_Engineering/ts1.pdf
    - · **Vol. 2: 95 Topics and 1531 Slides:**
      http://www.imm.dtu.dk/~db/Software_Engineering/ts2.pdf
    - · **Vol. 3: 83 Topics and 2563 Slides:**
      http://www.imm.dtu.dk/~db/Software_Engineering/ts3.pdf

- **More URLs:**
  - ⋆ **Software Engineering Book Home Page:**
    http://www.imm.dtu.dk/~db/Software_Engineering
  - ⋆ **Home Page for This Guide:**
    http://www.imm.dtu.dk/~db/Software_Engineering/se-guide.pdf
  - ⋆ **Vol. 1 Table of Contents:**
    http://www.imm.dtu.dk/~db/Software_Engineering/vol1/toc1.pdf
  - ⋆ **Vol. 2 Table of Contents:**
    http://www.imm.dtu.dk/~db/Software_Engineering/vol2/toc2.pdf
  - ⋆ **Vol. 3 Table of Contents:**
    http://www.imm.dtu.dk/~db/Software_Engineering/vol3/toc3.pdf
  - ⋆ **Springer's Vol. 1 URL:**
    http://www.springeronline.com/uk/3-540-21149-7
  - ⋆ **Springer's Vol. 2 URL:**
    http://www.springeronline.com/uk/3-540-21150-0
  - ⋆ **Springer's Vol. 3 URL:**
    http://www.springeronline.com/uk/3-540-21151-9

Contents

1 Background . . . 7
- 1.1 A Three-Volume Textbook on Software Engineering . . . 7
  - 1.1.1 Volume 1: Abstraction and Modelling . . . 7
  - 1.1.2 Volume 2: Specification of Systems and Languages . . . 7
  - 1.1.3 Volume 3: Domains, Requirements and Software Design . . . 8
- 1.2 Lecturers' and Students' Course Material . . . 8
- 1.3 Conclusion . . . 8

2 Course Proposals . . . 9
- 2.1 Undergraduate Courses . . . 9
  - 2.1.1 Two Courses in Software Engineering (SE I, SE II) . . . 9
    - 2.1.1.1 Software Engineering, SE I . . . 9
    - 2.1.1.2 Software Engineering, SE II . . . 10
  - 2.1.2 A Course in Discrete Mathematics for Software Engineering (DM) . . . 10
  - 2.1.3 A Course in Abstraction and Modelling (A&M) . . . 11
- 2.2 Graduate Courses . . . 11
  - 2.2.1 A Course in Language Design and Implementation (LD&I) . . . 11
  - 2.2.2 A Course in "UML"-ised Formal SE Techniques (SE/"UML") . . . 12
  - 2.2.3 A Course in Specification of Systems and Languages (SS&L) . . . 12
  - 2.2.4 Two Advanced Courses in Software Engineering (SE III A or B) . . . 13
- 2.3 "Single" Project Exercises . . . 13
  - 2.3.1 Vol. 1 Project Exercises . . . 13
  - 2.3.2 Vol. 2 Project Exercises . . . 13
  - 2.3.3 Vol. 3 Project Exercises . . . 14

3 Course and Volume Diagrams . . . 15
- 3.1 Course Diagram . . . 15
- 3.2 Brief Guide to Studies of Volume 1 . . . 15
- 3.3 Brief Guide to Studies of Volume 2 . . . 16
- 3.4 Brief Guide to Studies of Volume 3 . . . 18

4 Contents Listings: Volumes 1+2+3 . . . 21
- 4.1 Volume 1 . . . 22
- 4.2 Volume 2 . . . 30
- 4.3 Volume 3 . . . 38

References . . . 47

1

Background

1.1 A Three-Volume Textbook on Software Engineering

Chapter 4 lists three table of contents for this series of texts and handbooks on software engineering. From these the reader can see that together these volumes span 2284 pages.

1. Abstraction and Modelling	xxxvii + 711 pages
2. Specification of Systems and Languages	xxiv + 753 pages
3. Domains, Requirements and Software Design	xxx + 723 pages

1.1.1 Volume 1: Abstraction and Modelling

Volume 1 covers the basic principles and techniques of abstraction and modeling.

First Vol. 1 provides a sound, but simple basis of insight into discrete mathematics: numbers, sets, Cartesians, types, functions, the λ-calculus, algebras and mathematical logic.

Then Vol. 1 goes on to teach and train its readers in basic property- and model-oriented specification principles and techniques.

The model-oriented concepts that are common to such specification languages as B, VDM-SL, and Z will be propagated here through he use of the RAISE specification language RSL.

Finally Vol. 1 covers basic principles of functional, imperative and parallel specification programming — the latter based on the use of CSP, Hoare's language of communicating sequential processes.

1.1.2 Volume 2: Specification of Systems and Languages

Volume 2 covers the basic principles and techniques of specifying systems and languages.

First Vol. 2 teaches and trains its readers in advanced modelling principles and techniques: hierarchical versus compositional abstraction, denotational versus computational abstraction, and configurational abstraction, that is, abstracting and modeling contexts and states.

Then Vol. 2 goes on to teach and train its readers basic principles and techniques of modeling the semiotics: pragmatics, semantics and syntax of systems and languages.

An important part of Vol. 2 covers principles and techniques of modeling spatial and simple temporal phenomena.

A major part of Vol. 2 is devoted to such specialised topics as: modularity (incl. UML's class diagrams); Petri nets, message and live sequence charts, statecharts and temporal logics — including the duration calculus.

This part epitomises the adage: we "UML"-lise formal techniques. Thus the reader is brought, on a firm, precise, formal basis, on to the (trustworthy, believable and) professional use of such ideas as are the mainstay of UML — albeit the latter in informal, less precise disguises.

Finally Vol. 2 presents principles and techniques for developing the basis for sound and efficient interpreter and compiler development of functional, imperative, modular and parallel programming languages.

Volume 2 has Vol. 1 as prerequisite text.

1.1.3 Volume 3: Domains, Requirements and Software Design

Volume 3 covers the basic principles and techniques of overall software development: from domains via requirements to software designs.

Thus Vol. 3 advocates a novel approach to software engineering based on the adage: before requirements can be formulated one must understand the application domain.

Volume 3 therefore structured as follows: from (i) the principles and techniques for the development of domain descriptions, via (ii) principles and techniques for the derivation of requirements prescriptions from domain models, to (iii) principles and techniques for the refinement of requirements into software designs: architectures and component design.

Emphasis in the coverage of domain and requirements engineering is on "what goes into a proper domain description", respectively "requirements prescription", and on "how does one acquire and analyse the domain knowledge", respectively "the requirements expectations", and "how does one validate and verify domain and requirements models".

Volume 3 is structured so as to be targetable for two rather different audiences: in one clearly marked structure — focusing only on the informal parts — Vol. 3 is targeted at undergraduate students in a second semester course on software engineering, as well as at college lecturers in that field. In another – for the full version of Vol. 3 — it is targeted at advanced students, lecturers and researchers.

This is a novel feature for textbooks.

As such Vol. 3 — in the first, simple version — has no prerequisite texts. In the full version Vol. 3 has Vol. 2 as prerequisite text.

1.2 Lecturers' and Students' Course Material

In addition to the three-volume book, there is extensive course material:

- **Assignments/Exercises:** Most chapters contain assignments, typically 5–6 per chapter.
- **Problem Solutions:** The published volumes do not contain solutions to these problems. As times go by the author may post solutions to an increasing number of problem assignments on the Internet.
- **249 Lecture Topics Covering 5884 Slides:** Each chapter is the direct source of topic lecture slides. For each volume the numbers of separate topics and slides are:
  - ⋆ **Vol. 1: 71 topics, for a total of 1790 slides**
    http://www.imm.dtu.dk/~db/Software_Engineering/ts1.pdf
  - ⋆ **Vol. 2: 95 topics, for a total of 1531 slides**
    http://www.imm.dtu.dk/~db/Software_Engineering/ts2.pdf
  - ⋆ **Vol. 3: 83 topics, for a total of 2563 slides**
    http://www.imm.dtu.dk/~db/Software_Engineering/ts3.pdf

  Course lecturers will be able by means of, for example, `ghostview` to edit slides into any subsets suitable for individual lecture. All slides will be made available to bona fide course lecturers.

1.3 Conclusion

Given the extent of the course material already worked out in quite some detail, it was thought expedient to suggest how the volumes could be used in some nine courses. This is done next.

2

Course Proposals

The nine courses outlined below are typically thought of as one semester, typically 12–14 week courses with typically two lectures a week (of two times 45 minutes). Some courses are centered around solving exercises posed in respective volume chapters — ending with a written three to four hour examination. Such courses, except the examination, can be "patterned" by readers self-studying these volumes. Other courses are more project-oriented: groups of from three to eight students work throughout the semester on a set of related exercises which together "make up" a "single", comprehensive exercise. Appendix A of Vol. 1 and Sect. 1.7.1 of Vol. 3 contain outlines of three (3), respectively fifteen (15) "single", comprehensive project exercises. Chapter exercises in respective volumes then poses specific questions with reference to these 3+15 projects. Chapters 6–9 of Vol. 2 similarly poses a exercises which together form one comprehensive problem — which can, again, be subject to group projects. We refer to Sect. 2.3 for more details.

2.1 Undergraduate Courses

2.1.1 Two Courses in Software Engineering (SE I, SE II)

Volume 3 can be read and taught in basically two ways. The text of Vol. 3 is organised so that it is made syntactically very clear that some material can be skipped in a first reading. That "skippable" material constitutes the "formal stuff" of Vol. 3. Hence:

- **SE I + SE II, Vol. 3 Informally:** See Sects. 2.1.1.1 and 2.1.1.2.
  These two standard module courses, or one double module course, uses only the informal material of Vol. 3.
- **SE I + SE II, Vol. 3 Formally:** See Sect. 2.2.4 on page 13.
  This course may need the informal version of SE I + SE II as prerequisite course(s), depending on the maturity of the students.

2.1.1.1 Software Engineering, SE I

- **Aim:** To cover basic principles and techniques of domain and requirements engineering.
- **Objectives:** To set the reader free to engage with others in domain and requirements engineering, and to enable the reader to follow SE II.

This course is suggested to cover the following Vol. 3 material:

- Vol. 3: Chap. 1, Sects. 1.1, 1.2: The TripTych SE Paradigm
- Vol. 3: Chap. 2. Documentation
- Vol. 3: Chap. 5. Phenomena an Concepts
- Vol. 3: Chap. 8. Overview of Domain Engineering

- Vol. 3: Chap. 9. Domain Stake Holders
- Vol. 3: Chap. 11. Domain Facets
- Vol. 3: Chap. 16. The Domain Engineering Process Model
- Vol. 3: Chap. 17. Overview of Requirements Engineering
- Vol. 3: Chap. 19. Requirements Facets
- Vol. 3: Chap. 24. The Requirements Engineering Process Model

We refer to Sect. 2.3.3 on page 14.

2.1.1.2 Software Engineering, SE II

- **Aim:** To cover advanced principles and techniques of domain and requirements engineering, and to cover basic principles and techniques of software architecture and components design.
- **Objectives:** To set the reader free to lead other software engineers in software development.

This course is suggested to cover the following “intervening” Vol. 3 material:

- Vol. 3: Chap. 1, Sects. 1.3, 1.4, 1.5: The TripTych SE Paradigm
- Vol. 3: Chap. 3. Methods and Methodology
- Vol. 3: Chap. 4. Models and Modelling
- Vol. 3: Chap. 6. On Defining and on Definitions
- Vol. 3: Chap. 7. Jackson’s Description Principles
- Vol. 3: Chap. 10. Domain Attributes
- Vol. 3: Chap. 12. Domain Acquisition
- Vol. 3: Chap. 13. Domain Analysis and Concept Formation
- Vol. 3: Chap. 14. Domain Validation and Verification
- Vol. 3: Chap. 18. Requirements Stake Holders
- Vol. 3: Chap. 20. Requirements Acquisition
- Vol. 3: Chap. 21. Requirements Analysis and Concept Formation
- Vol. 3: Chap. 22. Requirements Validation and Verification
- Vol. 3: Chap. 23. Requirements Satisfiability and Feasibility
- Vol. 3: Chap. 25. Hardware/Software Co-Design
- Vol. 3: Chap. 26. Software Architecture Design
- Vol. 3: Chap. 27. A Case Study in Component Design

We refer to Sect. 2.3.3 on page 14.

2.1.2 A Course in Discrete Mathematics for Software Engineering (DM)

- **Aim:** To cover basic concepts of numbers, sets, Cartesians, types, functions, λ-calculus, algebras and mathematical logic.
- **Objectives:** To enable the reader to follow courses in and to self-study material on formal techniques in software engineering. That is, the remaining material on Vols. 1 and 2 as well as the formal parts of Vol. 3 (A&M, LD&I, SE/“UML”, SS&L and SE III (A or B)).

This course is suggested to cover the following (approximately 200-page) Vol. 1 chapters if not already taught in some other undergraduate course:

- Vol. 1: Chap. 1. Overview part of the Introduction
- Vol. 1: Chap. 2. Numbers
- Vol. 1: Chap. 3. Sets
- Vol. 1: Chap. 4. Cartesians
- Vol. 1: Chap. 5. Types
- Vol. 1: Chap. 6. Functions
- Vol. 1: Chap. 7. A λ–calculus
- Vol. 1: Chap. 8. Algebras
- Vol. 1: Chap. 9. Mathematical Logic

2.1.3 A Course in Abstraction and Modelling (A&M)

- **Aim:** To cover basic principles and techniques of abstraction and modelling (using the formal specification tool of the `RAISE` Specification Language, `RSL`).
- **Objectives:** To enable the reader to master basic concepts of and practical skills with respect to abstraction and modelling, and to serve as a prerequisite for courses LS&I, SS&L and SE/“UML”.

This course is suggested to cover selected parts of the following Vol. 1:

- Vol. 1: Chap. 1. Introduction
- Vol. 1: Chap. 10. Atomic Types and Values in `RSL`
- Vol. 1: Chap. 11. Function Definitions in `RSL`
- Vol. 1: Chap. 12. Property and Model Oriented Abstraction
- Vol. 1: Chap. 13. Sets in `RSL`
- Vol. 1: Chap. 14. Cartesians in `RSL`
- Vol. 1: Chap. 15. Lists in `RSL`
- Vol. 1: Chap. 16. Maps in `RSL`
- Vol. 1: Chap. 17. Higher-order Functions in `RSL`
- Vol. 1: Chap. 18. Types in `RSL`
- Vol. 1: Chap. 19. Applicative Specification Programming (lightly)
- Vol. 1: Chap. 20. Imperative Specification Programming (lightly)
- Vol. 1: Chap. 21. Parallel Specification Programming (lightly)
- Vol. 1: Chap. 22. Etcetera!

We refer to Sect. 2.3.1 on page 13.

2.2 Graduate Courses

2.2.1 A Course in Language Design and Implementation (LD&I)

This course has A&M as a prerequisite course. This course could (and should) precede a course on compiler development based on, for example, [1, 11]. The LD&I course could also be given jointly with a course on formal semantics using either of the books [2, 3, 9, 7, 8, 10, 12].

- **Aim:** To introduce the candidate to basic principles and techniques of semiotics: pragmatics, semantics and syntax, to cover basic abstraction and modelling principles and techniques of denotational (including continuation) and computational (i.e., mechanical) semantics, and to cover basic principles and techniques of designing typed, functional, imperative, modular and parallel programming languages.
- **Objectives:** To enable the reader to design reasonably advanced, yet simple and elegant programming languages, and to develop precise requirements for their interpretation and compilation.

This course is suggested to cover (selected parts of) the following Vol. 1 and Vol. 2 chapters:

- Vol. 2: Chap. 1. Introduction
- Vol. 1: Chap. 19. Applicative Specification Programming
- Vol. 1: Chap. 20. Imperative Specification Programming
- Vol. 1: Chap. 21. Parallel Specification Programming
- Vol. 2: Chap. 6. Pragmatics
- Vol. 2: Chap. 7. Semantics
- Vol. 2: Chap. 8. Syntax
- Vol. 2: Chap. 9. Semiotics
- Vol. 2: Chap. 16. Simple Applicative Language

- Vol. 2: Chap. 17. SIL: Simple Imperative Language
- Vol. 2: Chap. 18. SMIL: Simple Modular, Imperative Language
- Vol. 2: Chap. 19. SPIL: Simple Parallel, Imperative Language

We refer to Sect. 2.3.2 on the facing page.

2.2.2 A Course in "UML"-ised Formal SE Techniques (SE/"UML")

This course has course (a study of) A&M as a prerequisite course (study).

- **Aim:** To introduce the reader to basic diagrammatic specification principles and techniques of software development such as the latter (the diagrammatic) are found, albeit less precisely, in, for example, `UML`.
- **Objectives:** To enable the reader to master principles and techniques of Petri nets, message and live sequence charts and statecharts — and thus, as a beneficial side effect, to more professionally use `UML` critically and effectively.

This course is suggested to cover (selected parts of) the following Vol. 2 chapters:

- Vol. 2: Chap. 1. Introduction
- Vol. 2: Chap. 10. Modularisation, Class Diagrams
- Vol. 2: Chap. 11. State Automata and Machines
- Vol. 2: Chap. 12. Petri Nets
- Vol. 2: Chap. 13. Message and Live Sequence Charts
- Vol. 2: Chap. 14. Statecharts

• • •

The above course, SE/"UML", is a brief version of the next course, SS&L. For both courses the Vol. 2, Chap. 12 coverage of Petri nets is adequate. But a deeper, more satisfying study of Petri nets can be advised. We recommend [5, 4, 6]. Vol. 2, Chap. 12 is short on challenging exercises. Such can be found in [5, 4, 6].

2.2.3 A Course in Specification of Systems and Languages (SS&L)

This course has course (a study of) A&M as a prerequisite course (study).

- **Aim:** To introduce the reader to advanced abstraction and modelling principles and techniques for the development of complex systems including real-time embedded systems.
- **Objectives:** To enable the reader to tackle, with confidence, the specification of sophisticated computing systems.

This course is suggested to cover selected parts of the following Vol. 2 chapters:

- Vol. 2: Chap. 1. Introduction
- Vol. 2: Chap. 2. Hierarchies & Compositions
- Vol. 2: Chap. 3. Denotations & Computations
- Vol. 2: Chap. 4. Configurations: Contexts and States
- Vol. 2: Chap. 5. Time, Space and Space/Time
- Vol. 2: Chap. 10. Modularisation, Class Diagrams
- Vol. 2: Chap. 11. State Automata and Machines
- Vol. 2: Chap. 12. Petri Nets
- Vol. 2: Chap. 13. Message and Live Sequence Charts
- Vol. 2: Chap. 14. Statecharts
- Vol. 2: Chap. 15. Quantitative Models of Time

2.2.4 Two Advanced Courses in Software Engineering (SE III A or B)

This course is basically the formal version of courses SE I + SE II, see Sects.2.1.1.1 and 2.1.1.2.

SE III will cover the same chapters as courses SE I and SE II, but now emphasising the formal material.

There are two versions of this course:

- **SE III A:** A short, having the informal version of SE I and SE II as prerequisite.
  - ⋆ **Aim:** To introduce the reader to principles and techniques of formal software engineering: modelling domains, requirements and software design. The course contents follow that of course SE I, but present this material formally.
  - ⋆ **Objectives:**
- **SE III B:** A long, which does not have SE I + SE II as prerequisite — but instead covers the chapters mentioned in Sect. 2.1.1 on page 9 and 2.1.1 on page 10.
  - ⋆ **Aim:** To introduce the reader to principles and techniques of formal software engineering: modelling domains, requirements and software design, and additionally covers principles and techniques of acquisition, analysis, verification, verifications as well as other topics. The course contents follow that of courses SE I + SE II, but present this material formally.
  - ⋆ **Objectives:** To enable the reader to lead formal development of large software systems.

Course SE III A has courses (a study of) LD&I or SS&L as a prerequisite course (study). Course SE III B has both LD&I or SS&L as prerequisites.

2.3 "Single" Project Exercises

In the introduction to this chapter we briefly mentioned a notion of "single", comprehensive group project exercises. We repeat from there:

> *Appendix A of Vol. 1 and Sect. 1.7.1 of Vol. 3 contain outlines of three (3), respectively fifteen (15) "single", comprehensive project exercises. Chapter exercises in respective volumes then poses specific questions with reference to these 3+15 projects. Chapters 6–9 of Vol. 2 similarly poses a exercises which together form one comprehensive problem — which can, again, be subject to group projects.*

2.3.1 Vol. 1 Project Exercises

The three exercise clusters of Vol. 1 are:

1. Transportation Nets
2. Container Logistics
3. Financial Service Industry

Exercises section of chapters 2–5, 8–16, 18–21 then poses a total of nineteen (19) assignments for each of these three application areas.

2.3.2 Vol. 2 Project Exercises

Exercises sections of chapters 6–9 poses fourteen (14) assignments related to the pragmatics, semantics and syntax of an evolving set of programming languages: first (Chaps. 6–9), `STIL`, a simply typed imperative language, then (Chap. 9) three extensions to `STIL`, extensions that should result in a complete definition of an imperative programming language which allows the programmer to define types with dimensions and units (like length, respective either millimeter, centimeter, meter, kilometer, etc.). On the basis of Chaps. 16-19 of Vol. 2, the course lecturer can extend thus language design project into an interpreter and compiler development project.

2.3.3 Vol. 3 Project Exercises

The fifteen exercise clusters of Vol. 3 center around domain, requirements and software design engineering answers (including software related) to the following questions

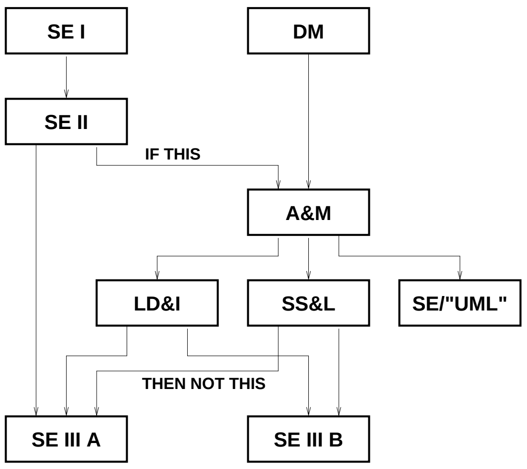
1. *What is administrative forms processing?*
2. *What is an airport?*
3. *What is air traffic?*
4. *What is a container harbour?*
5. *What is a document system?*
6. *What is a financial service system?*
7. *What is freight logistics?*
8. *What is a hospital?*
9. *What is a manufacturing company?*
10. *What is the market?*
11. *What is a metropolitan area tourism industry?*
12. *What is a railway system?*
13. *What is a university?*
14. *What is public administration?*
15. *What is a ministry of finance?*

Almost all exercise sections of Vol. 3 chapters contain almost 70 specific assignments related to any one of the above specific project clusters.

3 Course and Volume Diagrams

3.1 Course Diagram

The following diagram hints at possible prerequisite "arrangements" of the up towards nine different courses:



3.2 Brief Guide to Studies of Volume 1

Figure 3.1 abstracts a precedence relation between chapters. It is one that approximates suggested sequences of studying this volume.

- Chapter 1 is considered a prerequisite for the study of any chapter.
- Chapters 2–4 may be skipped by readers with some schooling in discrete mathematics.
- Chapters 5–6 may be skipped by readers with a bit more schooling in discrete mathematics.
- Chapters 7–9 can only be skipped by readers who have a reasonably firm grip on the topics mentioned.
- Chapters 10–16 from the core of Vol. 1.
- If, after Chap. 1, you continue with Chap. 2, then you should study all of Chaps. 2–9.
- If, after Chap. 1, you continue with Chap. 5, then you should study all of Chaps. 5–9.
- If, after Chap. 1, you continue with Chap. 7, then you should study all of Chaps. 7–9.
- You can skip Chaps. 17 and/or 18 before continuing with Chaps. 19–21.
- You can exit your study of this volume after any of Chaps. 16–21.
- It is no harm to study Chap. 22.

Within most chapters many sections can be skipped. Typically those with larger examples or towards the end of the chapters.

In this way a teacher or a reader can compose a number of suitable courses and studies.

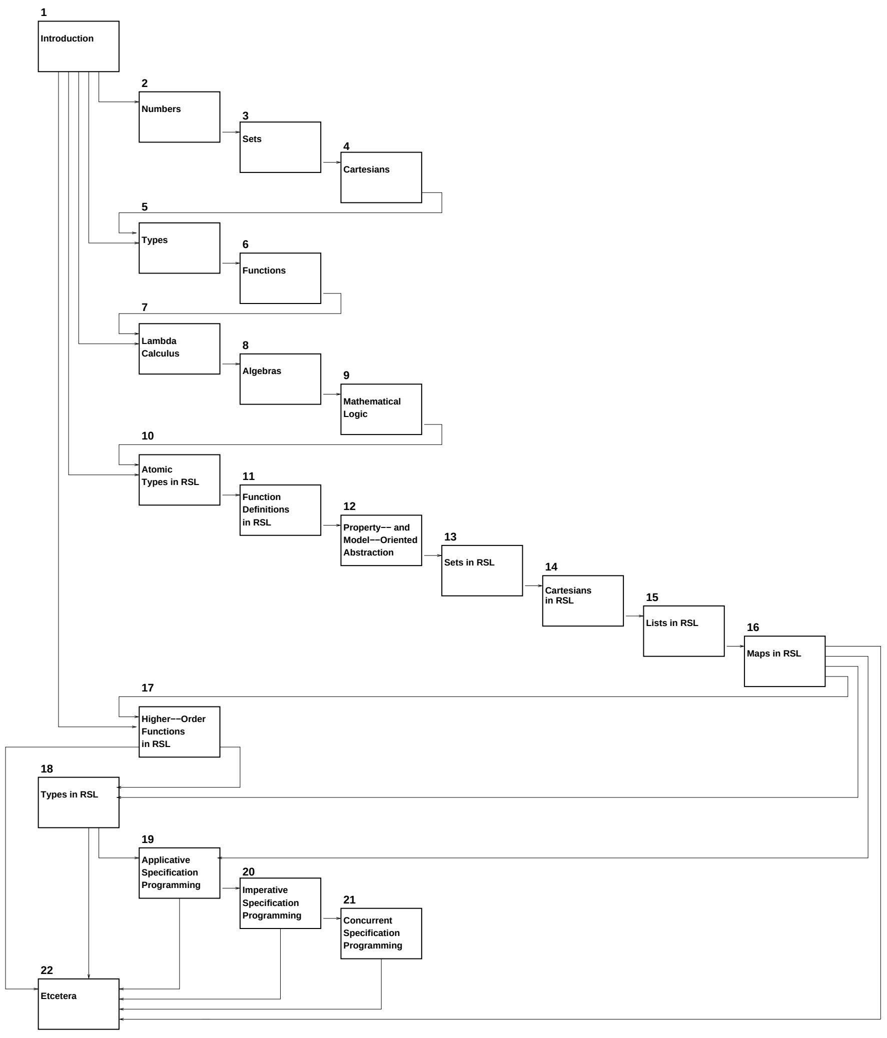

1 Introduction
2 Numbers
3 Sets
4 Cartesians
5 Types
6 Functions
7 Lambda Calculus
8 Algebras
9 Mathematical Logic
10 Atomic Types in RSL
11 Function Definitions in RSL
12 Property-- and Model--Oriented Abstraction
13 Sets in RSL
14 Cartesians in RSL
15 Lists in RSL
16 Maps in RSL
17 Higher--Order Functions in RSL
18 Types in RSL
19 Applicative Specification Programming
20 Imperative Specification Programming
21 Concurrent Specification Programming
22 Etcetera

Fig. 3.1. Volume 1 chapter precedence graph

3.3 Brief Guide to Studies of Volume 2

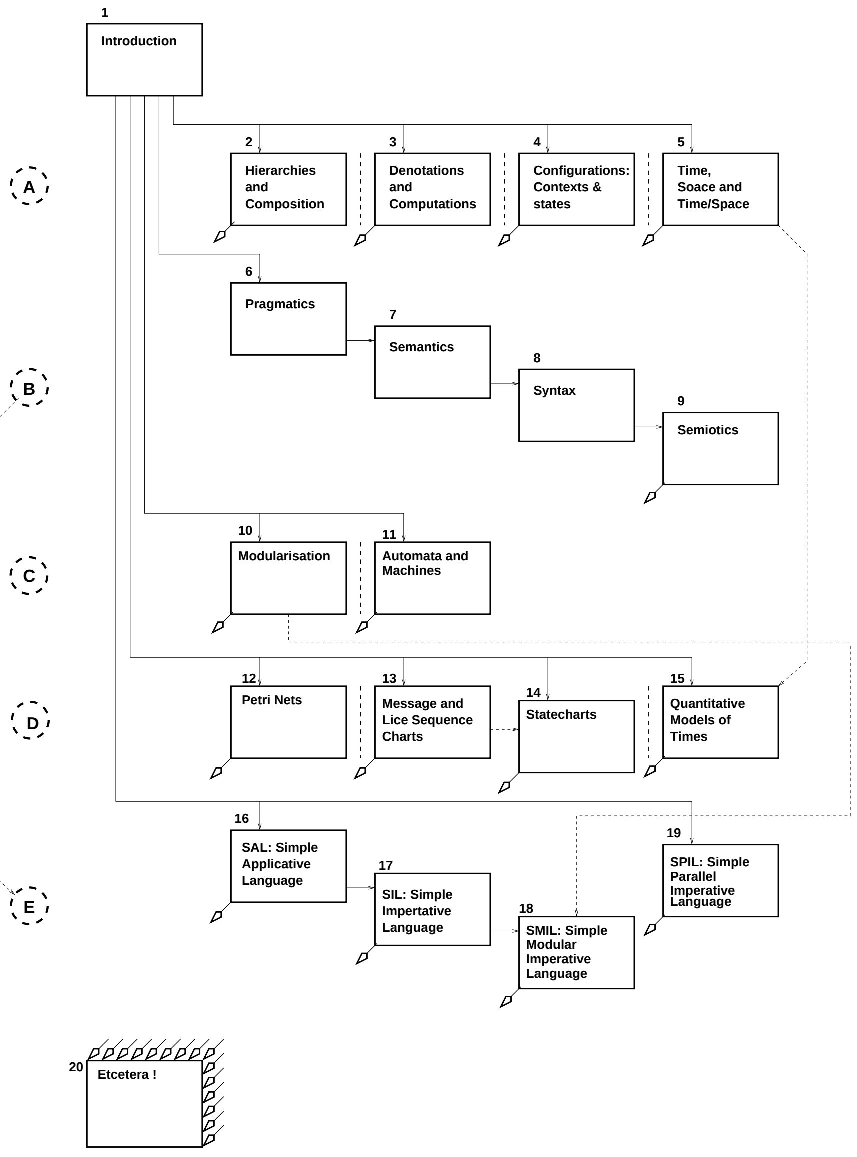
Figure 3.2 abstracts a precedence relation between chapters. It is one that approximates suggested sequences of studying this volume.

- Chapter 1 is considered a prerequisite for the study of any chapter.
- We group some parts into the dash-circled groups $A - E$.
- Group A consists of Chaps. 2–5 that can be studied in any order.
- Group B consists of Chaps. 6–9 that should be studied in order 6, 7, 8 and 9.
- Group C consists of Chaps. 10–11 that can be studied in any order.
- Group D consists of Chaps. 12–15 that can be studied in almost any order. Chaps. 14 does contain an example which requires having studied Chap. 13 first.

- Group E consists of Chaps. 16–19 that should be studied in order 16, 17 and 18. Chap. 19 can be studied in-between, before, or after. Preferably after.
- Groups $A - E$ can be studied in any order. But it might be useful to have studied Chap. 5 before Chap. 15, and Chap. 10 before Chap. 18, and to have studied Group B before Group E.
- It is no harm to study Chap. 20.

Within most chapters many sections can be skipped. Typically those with larger examples or towards the end of the chapters.

In this way a teacher or a reader can compose a number of suitable courses and studies.



1 Introduction
A
2 Hierarchies and Composition
3 Denotations and Computations
4 Configurations: Contexts & states
5 Time, Soace and Time/Space
6 Pragmatics
7 Semantics
8 Syntax
9 Semiotics
B
C
10 Modularisation
11 Automata and Machines
D
12 Petri Nets
13 Message and Lice Sequence Charts
14 Statecharts
15 Quantitative Models of Times
E
16 SAL: Simple Applicative Language
17 SIL: Simple Impertative Language
18 SMIL: Simple Modular Imperative Language
19 SPIL: Simple Parallel Imperative Language
20 Etcetera !

Fig. 3.2. Volume 2 chapter precedence graph

3.4 Brief Guide to Studies of Volume 3

Volume 3 can be studied in a number of ways. Any path — through chapters (i.e., nodes) — from the input node, labelled 1, to the output node, labelled 32, can form a course. Let us elaborate briefly on that:

Base Course in SE: A minimum course covers Chaps. 1, 2, 5, 8, 11, 16, 17, 19, 24–26, 30–32. That is, all the left column chapters!
Domain Engineering: A course focusing on domain engineering would additionally cover Chaps. 9, 10 and 12–15.
Requirements Engineering: A course focusing, instead, on requirements engineering would in addition to the base course cover Chaps. 18 and 20–23.
Software Design: A course focusing on software design would in addition to the base course cover Chaps. 17–29.

Any of the four courses outlined above can be given in either of two ways:

Informal: In this way of studying this volume the reader can skip the formalisation bits and focus just on the informal material. That is, one can study this volume, in principle and in reality, without first having studied Vol. 1 or Vols. 1 and 2.
Formal: In this way of studying this volume the reader cover all the informal material as well as the formal material — and thus a study of at least Vol. 1 is a prerequisite for studying the present volume.

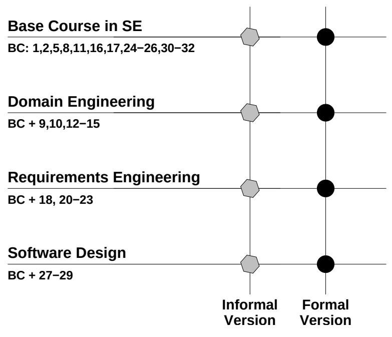


Fig. 3.3. Course alternatives

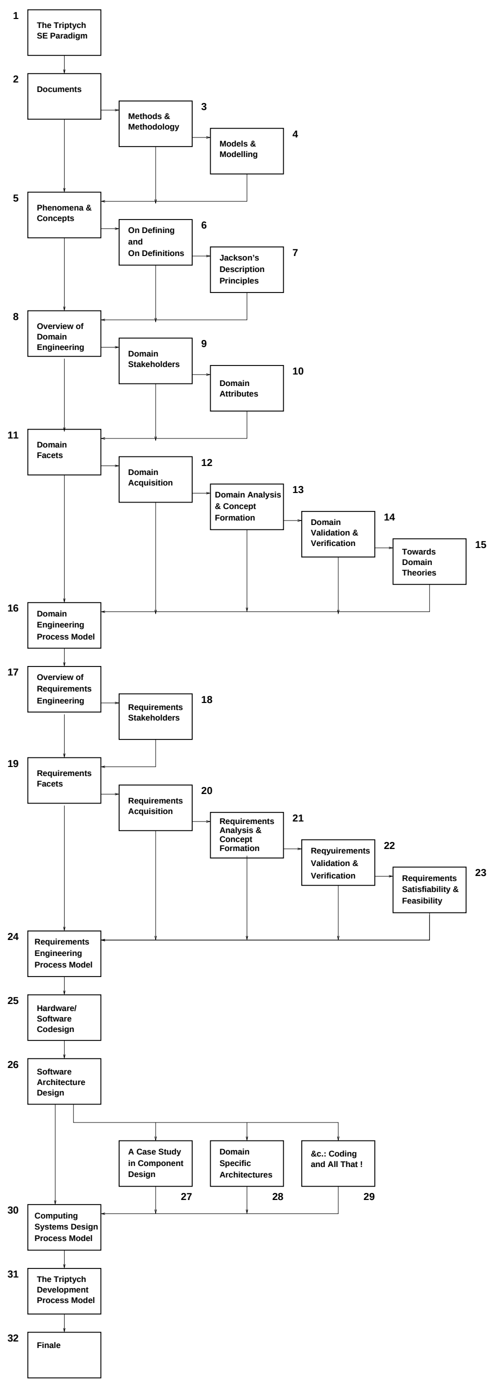

1 The Triptych SE Paradigm
2 Documents
3 Methods & Methodology
4 Models & Modelling
5 Phenomena & Concepts
6 On Defining and On Definitions
7 Jackson's Description Principles
8 Overview of Domain Engineering
9 Domain Stakeholders
10 Domain Attributes
11 Domain Facets
12 Domain Acquisition
13 Domain Analysis & Concept Formation
14 Domain Validation & Verification
15 Towards Domain Theories
16 Domain Engineering Process Model
17 Overview of Requirements Engineering
18 Requirements Stakeholders
19 Requirements Facets
20 Requirements Acquisition
21 Requirements Analysis & Concept Formation
22 Reqyuirements Validation & Verification
23 Requirements Satisfiability & Feasibility
24 Requirements Engineering Process Model
25 Hardware/ Software Codesign
26 Software Architecture Design
27 A Case Study in Component Design
28 Domain Specific Architectures
29 &c.: Coding and All That !
30 Computing Systems Design Process Model
31 The Triptych Development Process Model
32 Finale

Fig. 3.4. Volume 3 chapter precedence graph

Contents Listings: Volumes 1+2+3

- Volume 1: Pages 22–29
- Volume 2: Pages 30–37
- Volume 3: Pages 38–46

4.1 Volume 1

Volume 1: Abstraction and Modelling
http://www.imm.dtu.dk/~db/Software_Engineering/vol1/toc1.pdf

Preface v

Reasons for Writing These Volumes ... vii
Shortcomings of These Volumes ... viii
Methods of Approach ... x
A New Look at Software ... x
Formal Techniques "Light" ... xi
The "Super Programmer" ... xii
What Is Software Engineering? ... xiii
The Author's Aspirations ... xiv
Role of These Volumes in an SE Education Programme xiv
Why So Much Material? ... xviii
How to Use These Volumes in a Course ... xviii
Brief Guide to the Book ... xix
Guide to This Volume ... xix
Acknowledgments ... xxi

Part I OPENING

1 **Introduction** ... 3
1.1 Setting the Stage ... 3
1.2 A Software Engineering Triptych ... 7
1.2.1 Software Versus Systems Development ... 7
1.2.2 Motivating the Triptych ... 7
1.2.3 Domain Engineering ... 7
1.2.4 Requirements Engineering ... 9
1.2.5 Software Design ... 11
Software: Code and Documents ... 11
Software Design, I ... 11
Phases, Stages and Steps of Development ... 11
Software Design, II ... 12
1.2.6 Discussion ... 12
General Issues ... 12
1.3 Documentation ... 13
1.3.1 Document Kinds ... 14
1.3.2 Phase, Stage and Step Documents 14
1.3.3 Informative Documents ... 15
Current Situation Documentation . 15
Needs Documentation ... 15
Product Concepts and Facilities ... 16
Design Briefs ... 16
Scope and Span Delineations ... 16
Synopsis ... 16
Contracts ... 17
Discussion ... 17
1.3.4 Descriptive Documents ... 17
Descriptive Document Kinds and Types ... 18
Rough Sketches ... 18
Terminology ... 19
Narrative ... 19
Formal Model ... 21
Discussion ... 21
1.3.5 Analytic Documents ... 22
Rough Sketch Analysis and Concept Formation . 22
Validation of Descriptions, Prescriptions and Specifications ... 23
Verification of Properties of Specifications ... 23
Correctness of Development Phase, Stage or Step Transition ... 24
Discussion ... 25
1.4 Formal Techniques and Formal Tools ... 25
1.4.1 On Formal Techniques and Languages ... 25
1.4.2 Formal Techniques in SE Textbooks 26
1.4.3 Some Programming Languages ... 26
1.4.4 Some Formal Specification Languages ... 27
On Model-Oriented Specification Languages ... 27
On Property-Oriented Specification Languages ... 27
On Property-Oriented + Model-Oriented Specification Languages ... 28
More on Programming Languages . 28
Specification Languages Resumed . 28
1.4.5 Insufficiency of Current, Formal Languages ... 29
1.4.6 Other Formal Tools ... 30
1.4.7 Why Formal Techniques and Formal Tools? ... 30
Some Rationale ... 30
Anecdotal cum Analogical Evidence 31
1.5 Method and Methodology ... 31
1.5.1 Method ... 32
1.5.2 Methodology ... 32
1.5.3 Discussion ... 32
1.5.4 Meta-methodology ... 32
1.6 The Very Bases of Software ... 33
1.6.1 Didactics and Paradigms ... 34
1.6.2 Pragmatics, Semantics and Syntax 34
Pragmatics ... 34
Semantics ... 35
Syntax ... 37
1.6.3 On Specification and Programming Paradigms ... 38
1.6.4 Descriptions, Prescriptions and Specifications ... 38
Software Specifications, Requirements Prescriptions and Domain Descriptions 39
1.6.5 Metalanguages ... 39
1.6.6 Summary ... 39
1.7 Aims and Objectives ... 40
1.7.1 Aims ... 40
The Main Aims ... 40
Some Other Aims ... 40
1.7.2 Objectives ... 40
The Main Objectives ... 40
Some Other Objectives ... 40
1.7.3 Discussion ... 41
1.8 Bibliographical Notes ... 41
1.9 Exercises ... 41

Part II DISCRETE MATHEMATICS

2 **Numbers** 45
2.1 Introduction 45
2.2 Numerals and Numbers 46
2.3 Subsets of Numbers 46
2.3.1 Natural Numbers: **Nat** 46
Explanation of Some RSL Constructs 47
2.3.2 Integers: **Int** 48
Properties 48
The RSL Integer Algebra 48
2.3.3 Real Numbers: **Real** 49
Rational Numbers 49
Operations on Reals 49
2.3.4 Irrational Numbers 50
2.3.5 Algebraic Numbers 50
2.3.6 Transcendental Numbers 50
2.3.7 Complex and Imaginary Numbers 51
2.4 Type Definitions: Numbers 51
2.5 Summary 52
2.6 Bibliographical Notes 53
2.7 Exercises 53

3 **Sets** 55
3.1 Background 56
3.2 Mathematical Sets 56
3.3 Special Sets 58
3.3.1 Axiom of Extension 58
3.3.2 Partitions 58
3.3.3 Power Sets 58
3.4 Sorts and Type Definitions: Sets 58
3.4.1 Set Abstractions 58
3.4.2 Set Type Expressions and Type Definitions 59
3.4.3 Sorts 59
3.5 Sets in RSL 59
3.6 Bibliographical Notes 60
3.7 Exercises 60

4 **Cartesians** 63
4.1 The Issues 63
4.2 Cartesian-Valued Expressions 64
4.3 Cartesian Types 64
4.4 Cartesian Arity 65
4.5 Cartesian Equality 66
4.6 Some Construed Examples 66
4.7 Sorts and Type Definitions: Cartesians 68
4.7.1 Cartesian Abstractions 68
4.7.2 Cartesian Type Expressions and Type Definitions 68
4.8 Cartesians in RSL 69
4.9 Bibliographical Notes 69
4.10 Exercises 69

5 **Types** 71
5.1 Values and Types 72
5.2 Phenomena and Concept Types 73
5.2.1 Phenomena and Concepts 73
5.2.2 Entities: Atomic and Composite 73
5.2.3 Attributes and Values 74
Atomic Entity Attributes and Values 74
Composite Entity Attributes and Values 74
Discussion 76
5.3 Programming Language Type Concepts 77
Some Examples 77
Discussion 79
5.4 Sorts or Abstract Types 80
5.5 Built-in and Concrete Types 81
5.6 Type Checking 82
5.6.1 Typed Variables and Expressions 82
5.6.2 Type Errors 83
5.6.3 Detection of Type Errors 83
5.7 Types as Sets, Types as Lattices 84
5.8 Summary 84
5.9 Exercises 84

6 **Functions** 87
6.1 General Overview 89
Structure of This Chapter 89
6.1.1 Special Remarks 89
6.2 The Issues 90
6.2.1 Background 90
6.2.2 Some Concepts of Functions 90
Function Definitions 90
Function "Maps" (Graphs) 92
Types of Function Spaces and Function Signatures 92
Classes of Functions 94
6.3 How Do Functions Come About? 94
6.4 An Aside: On the Concept of Evaluation 96
6.4.1 [E]Valuation, Interpretation and Elaboration 96
6.4.2 Two Evaluation Examples 96
6.4.3 Function Invocation/"Function Call" 98
6.5 Function Algebras 98
6.5.1 Functions 98
6.5.2 Function Types 98
6.5.3 Higher-Order Function Types 99
6.5.4 Nondeterministic Functions 99
6.5.5 Constant Functions 100
6.5.6 Strict Functions 101
6.5.7 Strict Functions and Strict Function Invocation 101
6.5.8 Operations on Functions 101
6.6 Currying and λ-Notation 103
6.6.1 Currying 103
6.6.2 λ-Notation 103
6.6.3 Example of Currying and λ-Notation 104
6.7 Relations and Functions 104
6.7.1 Predicates 105
6.7.2 Function Evaluation by Relation Search 105
6.7.3 Nondeterministic Functions 106
6.8 Type Definitions 106
6.9 Conclusion 106
6.10 Bibliographical Notes 107
6.11 Exercises 107

7 **A λ-Calculus** 109
7.1 Informal Introduction 110
7.2 A "Pure" λ-Calculus Syntax 110
7.3 A λ-Calculus Pragmatics 112
7.4 A "Pure" λ-Calculus Semantics 112
7.4.1 Free and Bound Variables 113
7.4.2 Binding and Scope 113
7.4.3 Collision and Confusion of Variables 113
7.4.4 Substitution 114
7.4.5 α-Conversion and β-Conversion Rules 115
7.4.6 λ-Conversion 115
7.5 Call-by-Name Versus Call-by-Value 116
7.6 The Church–Rosser Theorems — Informal Version 117
7.7 The RSL λ-Notation 117
7.7.1 Extending λ-Expressions 117
7.7.2 The "**let ... in ... end**" Construct 118
7.8 Fix Points 119
7.8.1 The Issue 119
7.8.2 Informal Outline 119
7.8.3 The Fix Point Operator **Y** 120
7.8.4 Fix Point Evaluation 121
7.9 Discussion 122
7.9.1 General 122
7.9.2 On Minimal, Maximal and All Fix Points 122
7.9.3 Emphasis 122
7.9.4 Principles, Techniques and Tools 122
7.10 Bibliographical Notes 123
7.10.1 References 123
7.10.2 Alonzo Church, 1903–1995 123
7.11 Exercises 123

8 Algebras . . . 127
8.1 Introduction . . . 127
8.2 Formal Definition of the Algebra Concept . . . 128
8.3 How Do Algebras Come About? . . . 129
8.4 Kinds of Algebras . . . 130
8.4.1 Concrete Algebras . . . 130
8.4.2 Abstract Algebras . . . 130
8.4.3 Heterogeneous Algebras . . . 131
8.4.4 Universal Algebras . . . 132
The Morphism Concept . . . 132
Special Kinds of Morphisms . . . 133
8.5 Specification Algebras . . . 133
8.5.1 Syntactic Means of Expressing Algebras . . . 134
8.5.2 An Example Stack Algebra . . . 134
Informal Explanation of Some RSL Constructs . . . 135
8.5.3 An Example Queue Algebra . . . 135
Some Notation: **hide** . . . 136
8.5.4 Towards Semantic Models of "class" Expressions . . . 136
8.6 RSL Syntax for Algebra Specifications . . . 137
8.6.1 "class" Expressions . . . 137
8.6.2 "scheme" Declarations . . . 138
8.7 Discussion . . . 138
8.7.1 General . . . 138
8.7.2 Principles, Techniques and Tools . 139
8.8 Bibliographical Notes . . . 139
8.9 Exercises . . . 139

9 Mathematical Logic . . . 141
9.1 The Issues . . . 142
9.1.1 Language of Boolean Ground Terms . . . 142
9.1.2 Language of Propositional Expressions . . . 143
9.1.3 Language of Predicate Expressions 143
9.1.4 Boolean-Valued Expressions . . . 144
9.1.5 "chaos" — Undefined Expression Evaluations . . . 144
9.1.6 Axiom Systems and Inference Rules145
Axioms and Axiom Systems . . . 145
Inference Rules . . . 145
9.1.7 Proof Systems . . . 146
9.1.8 A Note on Two Axiom Systems . . 146
9.1.9 The "if ... then ... else ... end" Connective . . . 147
9.1.10 Discussion . . . 147
9.2 Proof Theory Versus Model Theory . . . 148
9.2.1 Syntax . . . 148
9.2.2 Semantics . . . 148
9.2.3 Syntax Versus Semantics . . . 149
9.2.4 Formal Logics: Syntax and Semantics . . . 149
More on the Semantics of Formal Logic Systems . . . 150
More on the Syntax of Formal Logic Systems . . . 150
On the Meaning of Material Implication, $\supset$ . . . 151
Metalinguistic Variables . . . 152
9.2.5 Issues Related to Proofs . . . 153
Proofs . . . 153
Theorems and Formal System Theories . . . 153
Consistency . . . 153
Decidability . . . 153
9.2.6 Relating Proof Theory to Model Theory . . . 153
Interpretation . . . 154
Models . . . 155
Satisfiability, Entailment: $\models$ and Validity . . . 155
Soundness and Completeness, $\vdash$ Versus $\models$ . . . 155
9.2.7 Discussion . . . 155
9.3 A Language of Boolean Ground Terms . . . 156
9.3.1 Syntax and Semantics . . . 156
9.3.2 The Connectives: $\sim$, $\wedge$, $\vee$, $\Rightarrow$, $=$, $\neq$, $\equiv$ . . . 157
Negation, $\sim$ . . . 157
Conjunction, $\wedge$ . . . 157
Disjunction, $\vee$ . . . 157
Equality, $=$ . . . 158
Implication, $\Rightarrow$ . . . 158
Identity, $\equiv$ . . . 158
9.3.3 Three-Valued Logic . . . 158
$\equiv$ Versus $=$ . . . 159
Form of Inference Rule . . . 159
Truth and Falsity (Syntactic) Designators and Semantic Values . . . 160
Non-commutativity of Boolean Connective . . . 160
9.3.4 Ground Terms and Their Evaluation161
Syntax of Boolean Ground Terms, BGT . . . 161
Boolean Ground Term Evaluation, Eval_BGT . . . 163
9.3.5 "Syntactic" Versus "Semantic Semantics" . . . 164
9.3.6 Discussion . . . 165
9.4 Languages of Propositional Logic . . . 165
9.4.1 Propositional Expressions, PRO . . 166
Examples of Propositional Expressions . . . 166
Syntax of Propositional Expressions, PRO . . 166
9.4.2 Examples . . . 167
9.4.3 Proposition Evaluation, Eval_PRO 168
9.4.4 Two-Valued Propositional Calculi . 169
Preliminaries . . . 169
Some Proof Concepts . . . 169
Axioms and a Rule of Inference, I . 170
Axioms and Inference Rules, II . . . 170
9.4.5 Discussion . . . 171
9.5 Languages of Predicate Logic . . . 171
9.5.1 Motivation . . . 172
9.5.2 Informal Presentation . . . 172
9.5.3 Examples . . . 173
9.5.4 Quantifiers and Quantified Expressions . . . 176
Syntax . . . 176
Free and Bound Variables . . . 177
Compound Quantified Expressions 177
9.5.5 Syntax of Predicate Expressions, PRE . . . 178
The Symbols of a Predicate Calculus . . . 178
The Term Language of a Predicate Calculus . 178
The Atomic Formula Language of a Predicate Calculus . . . 179
The Well-formed Formulas of a Predicate Calculus . 179
An Informal BNF Grammar for Predicate Expressions . . . 179
9.5.6 A Predicate Calculus . . . 180
Axiom Schemes . . . 180
Rules of Inference . . . 180
9.5.7 Predicate Expression Evaluation . . 181
Evaluation Contexts . . . 181
Meaning Versus Values of Predicate Expressions . . . 182
Evaluation Procedure, Eval_PRE . . 183
9.5.8 First-Order and Higher-Order Logics . . . 184
9.5.9 Validity, Satisfiability and Models . 184
Contexts and Interpretations . . . 184
Validity and Satisfiability . . . 185
Models . . . 185
9.5.10 Discussion . . . 186
9.6 Axiom Systems . . . 186
9.6.1 General . . . 187
9.6.2 Axioms . . . 187
9.6.3 Axiom System . . . 188

9.6.4 Consistency and Completeness . . . 189
9.6.5 Property-Oriented Specifications . . 189
9.6.6 Discussion 196
9.7 Summary 196
9.8 Bibliographical Notes 197
9.9 Exercises 197

Part III SIMPLE RSL

General 201
RSL Versus VDM-SL, Z and B 201
What, Syntactically, Constitutes a Specification? 202
Towards an RSL "Standard" 203
RSL Tools 203

10 Atomic Types and Values in RSL 205
10.1 Introduction 205
10.1.1 Mathematical Versus Enterprise Modelling 206
10.1.2 The "Primitive" Model Building Blocks 206
10.2 The RSL Numbers 206
10.2.1 Three Types of Numbers 207
Natural Numbers: Nat 207
Integers: Int 207
Real Numbers: Real 207
10.2.2 Operations on RSL Numbers 207
10.3 Enumerated Tokens 208
10.3.1 Motivation 208
10.3.2 General Theory 208
10.3.3 Operations on Tokens 209
10.3.4 Enumerated Tokens in Abstract Models 210
10.3.5 Modelling Using Enumerated Tokens 211
Enumerated Tokens and Finite State Devices 211
Enumerated Tokens and Linux Commands 212
10.4 Characters and Texts 212
10.4.1 Motivation 212
10.4.2 The Character and Text Data Types 212
10.5 Identifiers and General Tokens 213
10.5.1 Identifiers 213
RSL Identifiers 214
Universe of Discourse Identifiers . . 214
10.5.2 Operations on General Tokens 214
10.5.3 General Tokens 215
10.6 Discussion 216
10.6.1 General 216
10.6.2 Modelling Atomic Entities 216
10.7 Exercises 217

11 Function Definitions in RSL 221
11.1 The Function Type 221
11.1.1 Syntax of Function Types 221
11.1.2 Informal Semantics of Total and Partial Function Spaces 222
11.2 Model-Oriented Explicit Definitions 222
11.3 Model-Oriented Axiomatic Definitions 223
11.4 Model-Oriented pre/post-Condition Definitions 224
11.5 Property-Oriented Axiomatic Definitions 226
11.6 Property-Oriented Algebraic Definitions 227
11.7 Summary of RSL Function Definition Styles . . 228
11.8 Discussion 229
11.9 Exercises 229

12 Property-Oriented and Model-Oriented Abstraction . 231
12.1 Abstraction 232
12.1.1 The Issues 232
Modelling and Models 232
12.1.2 Abstraction and Specification 233
12.1.3 An Essay on Abstraction 233
12.2 Property-Oriented Abstractions 235
12.2.1 Pragmatics of Property-Oriented Specifications 235
12.2.2 Syntactics of Property-Oriented Specifications 236
Informal Documentation 236
Formal Description 239
12.2.3 Semantics of Property-Oriented Specifications 240
12.2.4 Discussion 240
General 240
Principles, Techniques and Tools . 240
12.3 Model Versus Property Abstractions 241
12.3.1 Representation and Operation Abstraction 241
12.3.2 Property- Versus Model-Oriented Abstractions 241
The Issues 242
Further Characterisations 242
12.3.3 Definitions 242
12.3.4 Representation Abstraction Examples 243
12.3.5 Operation Abstraction Examples . . 246
Property-Oriented Specification . . 247
Model-Oriented Specification 248
12.3.6 Discussion 248
General 248
Specific: *"What's the Difference Anyway?"* 249
12.4 Model-Oriented Abstractions 250
12.4.1 Ultrashort Overview of the Next Six Chapters 250
12.4.2 Models and Models 250
Models of Property-Oriented Specifications 250
Models of Model-Oriented Specifications 251
12.4.3 Underspecification 251
The Issue 251
Why Underspecifications? 251
12.4.4 Determinism and Nondeterminism 252
Deterministic Expressions 252
Nondeterministic Expressions 252
12.4.5 Why Loose Specifications? 253
12.4.6 Discussion 253
General 253
12.5 Principles, Techniques and Tools 254
12.5.1 Property- Versus Model-Oriented Specification? 254
When Property-Orientation? 254
When Model-Orientation? 254
12.5.2 Property-Oriented Specification Style 255
Blending Specification Styles 256
12.5.3 Model-Oriented Specification Style 256
Blending Specification Styles 257
12.5.4 Implicit and Explicit Functions . . . 257
12.5.5 No Confusion, Please! 257
12.5.6 A Note on Observer Functions . . . 258
First Principle: Postulation 258
Second Principle: No "Self-reference" 258
Third Principle: Identification 259
12.6 Exercises 260

13 Sets in RSL . . . 263
- 13.1 Sets: The Issues . . . 264
- 13.2 The Set Data Type . . . 265
  - 13.2.1 Set Types: Definitions and Expressions . . . 265
  - 13.2.2 Set Value Expressions . . . 266
    - Set Value Operator/Operand Expressions . . . 267
    - Mathematical Meaning of the Set Operators . . . 269
    - Set Comprehension . . . 270
    - Sets — Determinism and Nondeterminism Revisited . . . 271
    - Sets — Models, Values and the ≡ Operation . . . 271
  - 13.2.3 Set Binding Patterns and Matching 271
  - 13.2.4 Nondeterminism . . . 272
- 13.3 Examples of Set-Based Abstractions . . . 273
  - 13.3.1 Representation I . . . 273
  - 13.3.2 File Systems I . . . 273
  - 13.3.3 Representation II . . . 275
- 13.4 Abstracting and Modelling With Sets . . . 276
  - 13.4.1 Modelling Networks . . . 276
    - Formalisation of Flat Networks . . . 276
    - Narrative of Hyper-networks . . . 277
    - Formalisation of Hyper-networks . . . 277
  - 13.4.2 Modelling Pseudo-hierarchies . . . 277
    - Formalisation of Simple Graphs . . . 278
    - Narrative of Hyper-graphs . . . 278
    - Formalisation of Hyper-Graphs . . . 279
  - 13.4.3 Modelling a Telephone System . . . 280
    - Narrative of *the State* . . . 281
    - Formalisation of *the State* . . . 281
    - Narrative of *Efficient States* . . . 281
    - Formalisation of *Efficient States* . . . 281
    - Narrative of *Subscriber Actions* . . . 282
    - Formalisation of Action Types . . . 282
    - An Aside on Type Union and Variant Records . . . 282
    - Narrative: *Multi-party Call* . . . 283
    - Narrative: *Call Termination* . . . 283
    - Narrative: *Subscriber Busy* . . . 284
- 13.5 Inductive Set Definitions . . . 284
  - 13.5.1 Inductive Set Type Definitions . . . 284
  - 13.5.2 Inductive Set Value Definitions . . . 285
- 13.6 A Comment on Varying Sets . . . 287
- 13.7 Principles, Techniques and Tools . . . 288
- 13.8 Discussion . . . 289
- 13.9 Bibliographical Notes . . . 289
- 13.10 Exercises . . . 289

14 Cartesians in RSL . . . 295
- 14.1 Cartesians: The Issues . . . 295
- 14.2 The Cartesian Data Type . . . 296
  - 14.2.1 Cartesian Types and Type Expressions . . . 296
  - 14.2.2 Cartesian Value Expressions . . . 298
  - 14.2.3 Cartesian Operations, I . . . 299
  - 14.2.4 Cartesian Binding Patterns and Matching . . . 299
  - 14.2.5 Cartesian Operations, II . . . 300
- 14.3 Examples of Cartesian Abstractions . . . 300
  - 14.3.1 File Systems II . . . 300
  - 14.3.2 Kuratowski: Pairs as Sets . . . 301
- 14.4 Abstracting and Modelling with Cartesians . . . 303
  - 14.4.1 Modelling Syntactic Structures . . . 303
    - An Aside—The Union Type Operator: | . . . 304
    - Another Aside — Cartesian Text Types . . . 304
    - A Detour: The RSL "case" Construct . . . 306
  - 14.4.2 Cartesian "**let** ... **in** ... **end**" Bindings . . . 308
  - 14.4.3 Modelling Semantic Structures . . . 308
    - Narrative — Semantic Types . . . 308
    - Program Pointer Stacks . . . 309
    - Formalisation — Semantic Types . 310
    - Narrative — Computer Program Interpretation . . . 310
    - Formalisation — Semantic Functions . . . 310
  - 14.4.4 Cartesians: A First Discussion . . . 312
- 14.5 Inductive Cartesian Definitions . . . 312
  - 14.5.1 Inductive Cartesian Type Definitions . . . 312
  - 14.5.2 Inductive Cartesian Value Definitions . . . 313
- 14.6 Discussion . . . 315
  - 14.6.1 General . . . 315
  - 14.6.2 Principles, Techniques and Tools . 315
- 14.7 Exercises . . . 316

15 Lists in RSL . . . 321
- 15.1 Issues Related to Lists . . . 322
- 15.2 The List Data Type . . . 322
  - 15.2.1 List Types . . . 322
  - 15.2.2 List Value Expressions . . . 323
    - List Enumerations . . . 323
    - List Value Operator/Operand Expressions . . . 324
    - List Comprehension . . . 326
  - 15.2.3 List Binding-Patterns and Matching 327
  - 15.2.4 Lists: Determinism and Nondeterminism Revisited . . . 328
- 15.3 Small Examples of List-Based Abstractions . . . 328
  - 15.3.1 Representations . . . 328
  - 15.3.2 Stacks and Queues . . . 329
  - 15.3.3 File Systems III . . . 330
  - 15.3.4 Sorting Algorithms . . . 332
- 15.4 Abstracting and Modelling with Lists . . . 333
  - 15.4.1 Modelling Books Using Lists . . . 334
    - A Narrative of Entities . . . 334
    - Formalisation . . . 334
    - An Operations Narrative . . . 334
    - Formalisation . . . 335
  - 15.4.2 Modelling "KeyWord-In-Context, KWIC" . . . 335
- 15.5 Inductive List Definitions . . . 340
  - 15.5.1 Inductive List Type Definitions . . . 340
  - 15.5.2 Inductive List Value Definitions . . . 341
- 15.6 A Review of List Abstractions and Models . . . 342
- 15.7 Lists: A Discussion . . . 343
- 15.8 Exercises . . . 343

16 Maps in RSL . . . 349
- 16.1 The Issues . . . 350
- 16.2 The Map Data Type . . . 350
  - 16.2.1 Map Types: Definitions and Expressions . . . 350
  - 16.2.2 Map Value Expressions . . . 351
    - Map Enumerations . . . 351
    - Deterministic and Nondeterministic Map Values . . . 352
    - Map Operator/Operand Expressions . . . 352
    - Map Comprehension . . . 354
  - 16.2.3 Map Binding Patterns and Matching . . . 355
  - 16.2.4 Nondeterminism . . . 356
- 16.3 Examples of Map-Based Abstractions . . . 356
  - 16.3.1 Sorting . . . 356
  - 16.3.2 Equivalence Relations . . . 357
- 16.4 Abstracting and Modelling with Maps . . . 358
  - 16.4.1 Graphs . . . 358
  - 16.4.2 Structured Tables . . . 360
  - 16.4.3 Hierarchies . . . 362
  - 16.4.4 Relational File Systems (IV) and Databases . . . 366
  - 16.4.5 Complex Pointer Data Structures . 369
    - Discussion . . . 378
  - 16.4.6 Well-formedness of Data Structures 378
    - Discussion . . . 382
  - 16.4.7 Discussion . . . 382
- 16.5 Inductive Map Definitions . . . 383
  - 16.5.1 Inductive Map Type Definitions . . 383
  - 16.5.2 Inductive Map Value Definitions . . 384

16.6 A Review of Map Abstractions and Models . . . 386
16.7 Maps: A Discussion . . . 388
16.8 Exercises . . . 388

17 Higher-Order Functions in RSL . . . 393
17.1 Functions: The Issues . . . 393
17.2 Examples Using Function-Based Abstractions . 394
17.2.1 Functionals . . . 394
17.2.2 Discussion . . . 395
17.3 Abstracting and Modelling With Functions . . . 395
17.3.1 Concepts as Functions . . . 396
17.3.2 Operator Lifting . . . 399
Examples of Operator Lifting . . . 399
17.4 Inductive Function Definitions . . . 406
17.4.1 Inductive Function Type Definitions406
17.4.2 Inductive Function Value Definitions . . . 407
17.5 Review of Function Abstractions and Models . 407
17.6 Discussion . . . 408
17.7 Exercises . . . 408

Part IV SPECIFICATION TYPES

18 Types in RSL . . . 413
18.1 The Issues . . . 413
18.2 Type Categories . . . 415
18.2.1 Abstract Types: Sorts . . . 415
18.2.2 Concrete Types . . . 415
18.2.3 Discussion . . . 416
18.3 Enumerated Token Types Revisited . . . 416
18.4 Records: Constructors and Destructors . . . 417
18.4.1 General . . . 417
18.4.2 Variant Record Value Induction Axioms . . . 418
Non-recursive Record Type Definitions . . . 418
Recursive Record Type Definitions 419
18.4.3 An Example . . . 419
18.5 Union Type Definitions . . . 420
18.6 Short Record Type Definitions . . . 421
18.7 Type Expressions, Revisited . . . 421
18.8 Subtypes . . . 422
18.9 Type Definitions, Revisited . . . 422
18.10 On Recursive Type Definitions . . . 423
18.11 Discussion . . . 423
18.11.1 General . . . 423
18.11.2 Principles, Techniques and Tools . 423
18.12 Bibliographical Notes . . . 424
18.13 Exercises . . . 424

Part V SPECIFICATION PROGRAMMING

On Specification Programming . . . 427
On Problems and Exercises . . . 428

19 Applicative Specification Programming . . . 429
19.1 Scope and Binding . . . 430
19.1.1 Binding Patterns — An Informal Exposition . . . 430
19.1.2 "let" Construct Scope and Binding [1] . . . 432
19.1.3 Function Definition Scope and Binding [2] . . . 432
19.1.4 "case" Construct Scope and Binding [3] . . . 433
19.1.5 Comprehensions: Scope and Binding [4] . . . 434
19.1.6 Quantifications: Scope and Binding [5] . . . 435
19.2 Intuition . . . 435
19.2.1 Simple "let a = E_d in E_b(a) end" 435
19.2.2 Recursive "let f(a) = E_d(f) in E_b(f,a) end" . . . 436
19.2.3 Predicative "let a:A •P(a) in E(a) end" . . . 436
19.2.4 Multiple "let a_i = E_{d_i} in $E_b(a_i)$ end" . . . 436
19.2.5 Literals and Identifiers . . . 437
Literals . . . 437
Identifiers . . . 437
19.3 Operator/Operand Expressions . . . 438
19.4 Enumerated and Comprehended Expressions . . 438
19.5 Conditional Expressions . . . 439
19.6 Bindings, Typings, Patterns and Matching . . . 440
19.6.1 The Issues . . . 441
Syntactic Issues of Convenience . . 441
19.6.2 An Essence of Bindings and Patterns . . . 441
Bindings . . . 441
Patterns . . . 441
Bindings and Patterns: Apparent Differences . . . 442
Binding and Typing . . . 442
19.6.3 Binding Patterns . . . 443
Review of Earlier Material . . . 443
Record Binding Patterns . . . 444
General Forms of Binding Patterns 445
19.6.4 Typings . . . 448
19.6.5 Choice Patterns and Bindings . . . 448
Some Observations . . . 453
19.6.6 Summary . . . 454
19.7 Review and Discussion . . . 455
19.7.1 General . . . 455
19.7.2 Principles and Techniques . . . 455
19.8 Bibliographical Notes . . . 455
19.9 Exercises . . . 456

20 Imperative Specification Programming . . . 467
20.1 Intuition . . . 468
20.2 Imperative Combinators: A λ-Calculus . . . 468
20.2.1 [0] "variable" Declarations . . . 468
20.2.2 [1] Assignments: "var := expression" . . . 470
20.2.3 [9] State Expressions . . . 471
20.2.4 [2] "skip": No-Action . . . 471
20.2.5 [3] Statement Sequencing (;) . . . 471
20.2.6 [4] "if ... then ... else ... end" . . . 472
20.2.7 [5–6] "while ... do ... end", and "do ... until ... end" . . . 472
20.2.8 [7] "case ... of ... end" . . . 472
20.2.9 [8] "for... in ... do... end" . . . 473
20.3 Variable References: Pointers . . . 473
20.3.1 A Discourse on Simple References 473
20.3.2 Dynamic Allocation and Referencing . . . 474
Intuition . . . 475
Formalisation — Storage, Locations and Values475
Location and Value Types . . . 476
Storage Invariant . . . 477
Semantic Operations . . . 477
The Syntactic Forms . . . 478
Semantics . . . 478
20.3.3 Discussion: Semantics First, Then Syntax . . . 479
20.3.4 Discussion: Type Homomorphisms 480
20.3.5 The Notion of State . . . 480
20.4 Function Definitions and Expressions . . . 480
20.4.1 The **Unit** Type Expression, I . . . 481
20.4.2 Imperative Functions . . . 481
20.4.3 Read/Write Access Descriptions . . 481
20.4.4 Local Variables . . . 482
20.4.5 The **Unit** Type Expression, II . . . 482
20.4.6 Pure Expressions . . . 483
20.4.7 Read-Only Expressions . . . 483
Quantification over States (□) . . . 484
20.4.8 Equivalence (≡) and Equality (=) 484
Equivalence (≡) . . . 485
Conditional Equivalence . . . 485
Equality (=) . . . 485
20.5 Translations: Applicative to Imperative . . . 486
20.5.1 Applicative to Imperative Translations . . . 486
20.5.2 Recursive to Iterative Translations 487
20.5.3 Applicative to Imperative Schemas 488
Schemas . . . 489

20.5.4 Correctness, Principles, Techniques and Tools 495
20.6 Styles of Configuration Modelling 495
20.6.1 Applicative Contexts and States . . 495
Syntactic and Semantic Types . . . 495
Auxiliary Functions 496
Simple Interpretation Functions . . 497
Composite Interpretation Functions 497
Context-Creating Interpretation Functions 497
Summary, Applicative Contexts and States 498
20.6.2 Applicative Contexts and Imperative States 499
Metastate and Auxiliary Functions 500
Simple Interpretation Functions . . 500
Composite Interpretation Functions 500
Context-Creating Interpretation Functions 501
Summary, Applicative Contexts and Imperative States 501
20.6.3 Imperative Contexts and States . . 502
Metastate 502
Auxiliary Functions 502
Simple Interpretation Functions . . 503
Composite Interpretation Functions 503
Context-Creating Interpretation Functions 503
Summary, Imperative Contexts and States 504
20.6.4 Summary of Sequential Models . . . 505
20.7 Review and Discussion 505
20.7.1 Review 505
20.7.2 Discussion 506
20.8 Bibliographical Notes 506
20.8.1 Theory of Computation 506
20.8.2 A Type Theory for the λ-Calculus . 506
20.8.3 Source Program Transformation Works 507
20.8.4 Laws of Imperative Programming . 507
20.9 Exercises 508

21 Concurrent Specification Programming 511
21.1 Behaviour and Process Abstractions 512
21.1.1 Introduction 513
21.1.2 On Process and Other Abstractions 513
21.2 Intuition 514
21.2.1 Illustrative Rendezvous Scenarios . 514
Schematic "Rendezvous" Specifications 1–2–3 517
Schematic "Rendezvous" Specification 4 518
21.2.2 Diagram and Notation Summary . . 518
21.2.3 On a Trace Semantics 518
21.2.4 Some Characterisations: Processes, Etcetera 520
21.2.5 Principle of Process Modelling . . . 521
Components ≡ Processes 522
21.2.6 Informal Examples 522
21.2.7 Some Modelling Comments — An Aside 527
21.2.8 Examples Continued 528
21.2.9 Some System Channel Configurations 529
21.2.10 Concurrency Concepts — A Summary 530
21.3 Communicating Sequential Processes, CSP . . . 532
21.3.1 Preliminaries: Processes and Events 532
21.3.2 Process Combinators, Etcetera . . . 533
stop: A Basic Process 533
Prefix 533
Definitions 533
External Nondeterministic Choice . 533
Internal Nondeterministic Choice . 534
CSP Law (I) 534
Compound Events 534
Input and Output 535
Parallel Composition 535
Shared Events 535
CSP Law (II) 536
21.3.3 Discussion 536
21.4 The RSL/CSP Process Combinators 537
21.4.1 RSL-like Channels 537
21.4.2 RSL Communication Clauses 538
Simple Input/Output Clauses 538
21.4.3 RSL Processes 539
Simple Process Definitions 539
Processes and Their Definitions . . 539
Process Invocations 540
Array Channel Process Definitions 540
21.4.4 Parallel Process Combinator 541
21.4.5 Nondeterministic External Choice . 541
21.4.6 Nondeterministic Internal Choice . 542
21.4.7 Interlock Combinator 542
21.4.8 Summary 542
21.4.9 A Note of Caution 543
21.5 Translation Schemas 543
21.5.1 Stage I: An Applicative Schema . . 543
21.5.2 Stage II: A Simple Reformulation . 544
21.5.3 Stage III: Introducing Parallelism . 544
21.5.4 Stage IV: A Simple Reformulation 545
21.5.5 Stage Relations 546
21.5.6 Stage V: An Imperative Reformulation 547
21.5.7 Some Remarks 547
21.6 Parallelism and Concurrency: A Discussion . . . 547
21.6.1 CSP and RSL/CSP 547
21.6.2 Modelling Techniques 548
21.7 Bibliographical Notes 548
21.8 Exercises 548

Part VI AND SO ON!

22 Etcetera! 557
22.1 What Have We Covered? 557
22.2 What Is Next? 557
22.3 What Is Next-Next? 558
22.4 A Caveat 559
22.5 Formal Methods *"Lite"* 559
22.6 Bibliographical Notes 560

Part VII APPENDIXES

A Common Exercise Topics 563
A.1 Transportation Nets 563
A.2 Container Logistics 564
A.3 Financial Service Industry 564
A.4 Summary References to Exercises 566

B Glossary 567
B.1 Categories of Reference Lists 568
B.1.1 Glossary 568
B.1.2 Dictionary 568
B.1.3 Encyclopædia 569
B.1.4 Ontology 569
B.1.5 Taxonomy 569
B.1.6 Terminology 569
B.1.7 Thesaurus 569
B.2 Typography and Spelling 569
B.3 The Glosses 570

C Indexes 649
C.1 Symbols Index 650
C.1.1 Operators 650
Literal Operators 650
Relational Operators 650
Arithmetic Operators 651
Boolean Connectives 651
Set Operators *&c.* 651
Cartesian Composition 651
List Operators *&c.* 652
Map Operators *&c.* 652
Process Combinators 652
Comprehension 652
C.1.2 Constructors 652
Atomic Types 652
Composite Types 652
Function Constructs 653
Deconstructors 653
C.1.3 Constant Value Literals 653
C.1.4 Combinators 653
Statement Combinators 653
Clause Combinators 653
Specification Combinators 654
C.1.5 Calculi 654
λ The λ Calculus, 109–125 654
The Predicate Calculi 654
C.1.6 Abbreviations 654
C.2 Concepts Index 656
C.3 Characterisations and Definitions Index 680
C.4 Authors Index 682
References 687

4.2 Volume 2

Volume 2: Specification of Systems and Languages
http://www.imm.dtu.dk/˜db/Software_Engineering/vol2/toc2.pdf

PREFACE vii
Overview vii
"UML"-ising Formal Techniques viii
The RAISE Specification Language: RSL viii
Acknowledgments viii
Brief Guide to Volume 2 ix

Part I OPENING

1 Introduction 3
1.1 Introduction 3
1.1.1 Why This Volume? 3
1.1.2 Why Master These Principles, Techniques and Tools? 4
1.1.3 What Does This Volume "Contain"? 4
1.1.4 How Does This Volume "Deliver"? 5
1.2 Formal Techniques "Lite" 6
1.3 An RSL Primer 8
1.3.1 Types 8
Type Expressions 8
Type Definitions 10
1.3.2 The RSL Predicate Calculus 11
Propositional Expressions 11
Simple Predicate Expressions 12
Quantified Expressions 12
1.3.3 Concrete RSL Types 12
Set Enumerations 12
Cartesian Enumerations 13
List Enumerations 13
Map Enumerations 14
Set Operations 15
Cartesian Operations 17
List Operations 17
Map Operations 19
1.3.4 λ-Calculus+Functions 21
The λ-Calculus Syntax 21
Free and Bound Variables 21
Substitution 22
α-Renaming and β-Reduction 22
Function Signatures 22
Function Definitions 23
1.3.5 Other Applicative Expressions 23
Let Expressions 23
Conditionals 25
Operator/Operand Expressions 25
1.3.6 Imperative Constructs 26
Variables and Assignment 26
Statement Sequences and **skip** 26
Imperative Conditionals 26
Iterative Conditionals 27
Iterative Sequencing 27
1.3.7 Process Constructs 27
Process Channels 27
Process Composition 27
Input/Output Events 28
Process Definitions 28
1.3.8 Simple RSL Specifications 29
1.4 Bibliographical Notes 29

Part II SPECIFICATION FACETS

2 Hierarchies and Compositions 35
2.1 The Issues 35
2.1.1 Informal Illustrations 36
2.1.2 Formal Illustrations 36
2.2 Initial Methodological Consequences 37
2.2.1 Some Definitions 37
"Top-down" 38
"Bottom-up" 38
2.2.2 Principles and Techniques 38
2.3 The Main Example 39
2.3.1 A Hierarchical, Narrative Presentation 40
2.3.2 A Hierarchical, Formal Presentation 42
2.3.3 A Compositional, Narrative Presentation 45
2.3.4 A Compositional, Formal Presentation 47
2.4 Discussion 48
2.5 Bibliographical Notes 49
2.6 Exercises 50

3 Denotations and Computations 55
3.1 Introduction 55
3.1.1 Computations and Denotations 56
3.1.2 Syntax and Semantics 56
3.1.3 Characterisations 56
3.2 Denotational Semantics 57
3.2.1 A Simple Example: Numerals 57
3.2.2 The Denotational Principle 58
3.2.3 Expression Denotations 58
An Extension 61
3.2.4 GOTO Continuations 62
The Problem 64
Syntactic Types 66
Semantic Types 67
The Main Semantic Functions 67
A Profound Problem 70
Further Remarks on Example 3.3 . 70
The Remaining Semantic Functions 70
Discussion 72
3.2.5 Discussion of Denotational Semantics 72
3.3 Computational Semantics 74
3.3.1 The Issues 74
3.3.2 Two Examples 74
3.3.3 Expression Computations 74
A Computational State 75
Motivating the Control Stack 75
The Elaboration Functions 76
Discussion 78
3.3.4 Computational Semantics of GOTO Programs 78
The Syntax 78
Some Semantic Observations 79
Semantic Types 80
3.3.5 Computational Semantics of Coroutine Programs 83
"Coroutine" Program Text I 83
Coroutine Program Text II 84
3.3.6 Discussion 85
On the Mechanisation of Continuations 85
On the "Power" of GOTOs via Label Variables 85
On Computational Semantics 85
3.4 Review: Denotations and Computations 86
3.5 Some Pioneers of Semantics 86
3.5.1 John McCarthy 86
3.5.2 Peter Landin 88
3.6 Exercises 90

4 Configurations: Contexts and States ... 93
- 4.1 Introduction ... 94
- 4.2 The Issues ... 97
- 4.3 "Real-World" Contexts and States ... 98
  - 4.3.1 A Physical System: Context and State ... 99
  - 4.3.2 Configurations of Contexts and States ... 99
  - 4.3.3 Nonphysical System: Context and State ... 100
    - The Context ... 100
    - The State ... 101
    - A Model ... 101
  - 4.3.4 Discussion, I ... 101
  - 4.3.5 Discussion, II ... 102
- 4.4 First Summary: Contexts and States ... 102
  - 4.4.1 General ... 102
    - Model versus Specification States ... 103
  - 4.4.2 Development Principles and Techniques ... 103
- 4.5 Programming Language Configurations ... 104
- 4.6 Concurrent Process Configurations ... 104
  - 4.6.1 The Example ... 104
    - Comments on the Semantics of Fig. 4.2[A] ... 106
    - Comments on the Semantics of Fig. 4.2[B] ... 106
    - Comments on the Semantics of Fig. 4.2[C] ... 106
    - Comments on the Semantics of Fig. 4.2[D] ... 106
  - 4.6.2 Summary ... 110
- 4.7 Second Summary: Contexts and States ... 111
- 4.8 Information States and Behaviour States ... 112
  - 4.8.1 Program Flowcharts as State Machine Data ... 112
  - 4.8.2 Flowcharts ≡ Machines ... 113
  - 4.8.3 Flowchart Machines ... 113
  - 4.8.4 Observations ... 114
  - 4.8.5 Conclusion ... 115
- 4.9 Final Summary: Contexts and States ... 115
- 4.10 Exercises ... 116

Part III A CRUCIAL DOMAIN AND COMPUTING FACET

5 Time, Space and Space/Time ... 121
- 5.1 Time ... 122
  - 5.1.1 Time — The Basics ... 122
    - Time-Varying Entities = Dynamic Entities ... 122
    - Time and Dynamicity ... 124
  - 5.1.2 Time — General Issues ... 124
  - 5.1.3 *"A–Series"* and *"B–Series"* Models of Time ... 125
  - 5.1.4 A Continuum Theory of Time ... 125
  - 5.1.5 Temporal Events ... 126
  - 5.1.6 Temporal Behaviour ... 127
  - 5.1.7 Representation of Time ... 127
  - 5.1.8 Operations "on" Time ... 128
- 5.2 Space ... 129
  - 5.2.1 Space — The Basics ... 129
  - 5.2.2 Location-Varying Entities ... 129
  - 5.2.3 Locations and Dynamicity ... 131
  - 5.2.4 Space — General Issues ... 132
    - Point, Curve, Surface and Volume 132
    - Spatial "Events" ... 133
    - Spatial "Behaviours" ... 133
    - Representation of Spatial Bodies ... 133
    - Operations on Space ... 134
- 5.3 Space/Time ... 135
  - 5.3.1 A Guiding Example ... 135
  - 5.3.2 Representation of Space-Time ... 135
  - 5.3.3 Blizard's Theory of Time-Space ... 136
    - Discussion of the *Blizard* Model of Space-Time ... 137
- 5.4 Discussion ... 137
- 5.5 Bibliographical Notes ... 137
- 5.6 Exercises ... 137

Part IV LINGUISTICS

6 Pragmatics ... 145
- 6.1 Introduction ... 145
- 6.2 Everyday Pragmatics ... 146
- 6.3 "Formal" Pragmatics ... 146
- 6.4 Discussion ... 147
  - 6.4.1 General ... 147
  - 6.4.2 Principles and Techniques ... 148
- 6.5 Bibliographical Note ... 148
- 6.6 Exercises ... 149

7 Semantics ... 151
- 7.1 Introduction ... 151
- 7.2 Concrete Semantics ... 152
- 7.3 "Abstract" Semantics ... 152
- 7.4 Preliminary Semantics Concepts ... 152
  - 7.4.1 Syntactic and Semantic Types ... 153
  - 7.4.2 Contexts ... 153
  - 7.4.3 States ... 154
  - 7.4.4 Configurations ... 154
  - 7.4.5 Interpretation, Evaluation and Elaboration ... 154
- 7.5 Denotational Semantics ... 155
  - 7.5.1 Simple Case ... 156
  - 7.5.2 Composite Case ... 156
- 7.6 Macro-Expansion Semantics ... 157
  - 7.6.1 Rewriting ... 157
  - 7.6.2 Macro-Expansion ... 158
  - 7.6.3 Inductive Rewritings ... 158
    - Static Inductive Semantics ... 159
    - Dynamic Inductive Semantics ... 159
  - 7.6.4 Fix Point Evaluation ... 161
- 7.7 Operational and Computational Semantics ... 161
  - 7.7.1 Stack Semantics ... 162
  - 7.7.2 Attribute Grammar Semantics ... 162
    - A Symbolic Attributed Parse Tree Example ... 165
- 7.8 Proof Rule Semantics ... 166
- 7.9 Discussion ... 169
  - 7.9.1 General ... 169
  - 7.9.2 Principles, Techniques and Tools ... 169
- 7.10 Bibliographical Notes ... 170
- 7.11 Exercises ... 170

8 Syntax ... 173
- 8.1 The Issues ... 174
  - 8.1.1 Form and Content: Syntax and Semantics ... 174
  - 8.1.2 Structure and Contents of This Chapter ... 175
- 8.2 Sentential Versus Semantical Structures ... 175
  - 8.2.1 General ... 175
    - Syntax of Sentential Structures ... 175
    - Syntax of Semantical Structures ... 176
  - 8.2.2 Examples of Sentential Structures ... 176
    - Modelling Simple Sentential Structures ... 177
  - 8.2.3 Examples of Semantical Structures 178
- 8.3 The First Abstract Syntax, John McCarthy ... 181
  - 8.3.1 Analytic Grammars: Observers and Selectors ... 181
  - 8.3.2 Synthetic Grammars: Generators ... 182
- 8.4 BNF Grammars ≈ Concrete Syntax ... 183
  - 8.4.1 BNF Grammars ... 183
  - 8.4.2 BNF↔RSL Parse Trees Relations ... 184
- 8.5 Structure Generators and Recognisers ... 186
  - 8.5.1 Context-Free Grammars and Languages ... 186
  - 8.5.2 Parse Trees ... 188
  - 8.5.3 Regular Expressions and Languages 189
  - 8.5.4 Language Recognisers ... 190
- 8.6 XML: Extensible Markup Language ... 190
  - 8.6.1 An Example ... 191
  - 8.6.2 Discussion ... 192
  - 8.6.3 Historical Background ... 192
  - 8.6.4 The Current XML "Craze" ... 193
  - 8.6.5 XML Expressions ... 193
  - 8.6.6 XML Schemas ... 195

8.6.7 References . . . 197
8.7 Abstract Syntaxes . . . 197
8.7.1 Abstract Syntax of a Storage Model . . . 197
Values and Value Types . . . 197
Locations and Location Types . . . 198
Storages . . . 199
Type Constraints . . . 199
8.7.2 Abstract Syntaxes of other Storage Models . . . 200
Informal Exposition . . . 200
Formal Exposition . . . 201
8.8 Converting RSL Types to BNF . . . 202
8.8.1 The Problem . . . 202
8.8.2 A Possible Solution . . . 202
8.9 Discussion of Informal and Formal Syntax . . . 203
8.9.1 General . . . 203
8.9.2 Principles, Techniques and Tools . 204
8.10 Bibliographical Notes . . . 204
8.11 Exercises . . . 205

9 Semiotics . . . 213
9.1 Semiotics = Syntax ⊕ Semantics ⊕ Pragmatics . . . 213
9.2 Semiotics . . . 214
9.3 Language Components . . . 215
9.4 Linguistics . . . 216
9.5 Languages and Systems . . . 217
9.5.1 Professional Languages . . . 218
9.5.2 Metalanguages . . . 219
9.5.3 Systems . . . 219
A Physical System View . . . 219
Examples of Physical and Nonphysical Systems 220
A Linguistic Systems View . . . 222
Example System Languages . . . 223
A Flowchart Language . . . 226
Informal Syntax of Simple Flowcharts . . . 227
Formal Syntax of Simple Flow Charts . . . 228
Structured Flowcharts . . . 229
System Diagrams Versus Formal Specifications . . . 230
9.5.4 System Diagram Languages . . . 232
9.5.5 Discussion of System Concepts . . . 232
9.5.6 Systems as Languages . . . 233
9.6 Discussion . . . 233
9.6.1 General . . . 233
9.6.2 Principles, Techniques and Tools . 234
9.7 Charles Sanders Peirce . . . 234
9.8 Bibliographical Notes . . . 234
9.9 Exercises . . . 235

Part V FURTHER SPECIFICATION TECHNIQUES

10 Modularisation . . . 243
10.1 Introduction . . . 244
10.1.1 Some Examples . . . 244
Review of Examples . . . 248
10.1.2 Preparatory Discussion . . . 249
Software Devices . . . 250
Abstract Data Types → Algebraic Semantics 250
The Frames Approach . . . 250
The Entity-Relationship (ER) Approach . . . 251
General Pragmatics of Modularisation . . . 252
General Semantics of Modularisation . . . 252
General Syntax of Modularisation . 252
General Module Specification Method . . . 252
10.1.3 Structure of Chapter . . . 252
10.2 RSL Classes, Objects and Schemes . . . 253
10.2.1 Introducing the RSL "class" Concept . . . 253
Meaning of RSL Declarations — A Review . . . 253
First Motivation of the RSL "class" Concept: Focus . . . 253
Second Motivation of the RSL "class" Concept: Semantic Algebras . 253
Third Motivation of the RSL "class" Concept . . . 254
Fourth Motivation of the RSL "class" Concept: Named schemes . . . 256
10.2.2 The RSL "class" Concept . . . 257
10.2.3 The RSL "object" Concept . . . 257
10.2.4 The RSL "scheme" Concept . . . 257
Simple Schemes . . . 257
Scheme Extensions . . . 259
Hiding . . . 260
Etcetera . . . 262
10.2.5 RSL "scheme" Parameterisation . . 263
Motivation: Why and How Scheme Parameters? 263
The Syntax and Semantics of Parameterised Schemes . . . 265
10.2.6 A "Large-Scale" Example . . . 265
The Contrasting Background Example . . . 266
Narrative . . . 266
Formalisation . . . 266
The Schematised Example . . . 268
10.2.7 Definitions: Class, Scheme and Object . . . 270
10.3 UML and RSL . . . 271
10.3.1 Overview of UML Diagrams . . . 271
Use Case Diagrams . . . 271
Sequence/Collaboration Diagrams 272
Statechart Diagrams . . . 272
10.3.2 Class Diagrams . . . 272
UML "Standardisation" . . . 273
10.3.3 Class Diagrams . . . 273
Classes . . . 273
Association . . . 274
Links . . . 275
Generalisations . . . 276
10.3.4 Example: Railway Nets . . . 276
10.3.5 Comparison of UML and RSL OO Constructs . . . 278
10.3.6 References . . . 279
10.3.7 Class Diagram Limitations . . . 279
10.4 Discussion . . . 280
10.4.1 Modularity Issues . . . 280
Modular Specification and Programming . . . 280
Stability of Modularity Concepts . . 280
Whither Object-Oriented (OO) "Programming" . . . 280
Schema, Object and Module Calculi281
Formalisations of UML's Class Concept . . . 281
10.4.2 Principles, Techniques and Tools . 281
10.5 Bibliographical Notes . . . 282
10.6 Exercises . . . 282

11 Automata and Machines . . . 283
11.1 Discrete State Automata . . . 284
11.1.1 Intuition . . . 286
11.1.2 Motivation . . . 286
11.1.3 Pragmatics . . . 286
11.2 Discrete State Machines . . . 288
11.3 Finite State Automata . . . 289
11.3.1 Regular Expression Language Recognisers . . . 290
11.3.2 Regular Expressions . . . 291
11.3.3 Formal Languages and Automata . 292
11.3.4 Automaton Completion . . . 293
11.3.5 Nondeterministic Automata . . . 293
11.3.6 Minimal State Finite Automata . . . 294
11.3.7 Finite State Automata Formalisation, I . . . 295
11.3.8 Finite State Automata Realisation, I . . . 295
11.3.9 Finite State Automaton Formalisation, II . . . 296
11.3.10 Finite State Automata Realisation, II . . . 297
11.3.11 Finite State Automata — A Summary . . . 297
11.4 Finite State Machines . . . 298
11.4.1 Finite State Machine Controllers . . 299
11.4.2 Finite State Machine Parsers . . . 301
11.4.3 Finite State Machine Formalisation 303
11.4.4 Finite State Machine Realisation . . 303
11.4.5 Finite State Machines — A Summary . . . 305
11.5 Pushdown Stack Devices . . . 305
11.5.1 Pushdown Stack Automata and Machines . . . 305
11.5.2 Formalisation of Pushdown Stack Machines . . . 307
11.5.3 Pushdown Stack Device Summary 309
11.6 Bibliographical Notes: Automata and Machines 309
11.7 Exercises . . . 310

Part VI CONCURRENCY AND TEMPORALITY

12 Petri Nets . . . 313
Christian Krog Madsen is chief author of this chapter
12.1 The Issues . . . 313
12.2 Condition Event Nets (CENs) . . . 314
12.2.1 Description . . . 314
12.2.2 Small CEN Examples . . . 315
12.2.3 An RSL Model of Condition Event Nets . . . 319
Syntax of CENs and A Static Semantics . . . 319
A Dynamic Semantics . . . 320
12.3 Place Transition Nets (PTNs) . . . 321
12.3.1 Description . . . 321
12.3.2 Small PTN Examples . . . 321
12.3.3 An RSL Model of Place Transition Nets . . . 322
Syntax of PTPNs and A Static Semantics . . . 322
A Dynamic Semantics . . . 324
12.3.4 Railway Domain Petri Net Examples . . . 326
Route Descriptions . . . 326
Interlocking Tables . . . 327
Petri Net for Units . . . 328
Petri Net for Switches . . . 328
Petri Net for Signals . . . 329
Petri Net for Routes . . . 329
Construction of Petri Net for Interlocking Tables . 329
Summary . . . 330
12.4 Coloured Petri Nets (CPNs) . . . 331
12.4.1 Description . . . 331
12.4.2 A CPN Example . . . 332
12.4.3 An RSL Model of Coloured Petri Nets . . . 334
Syntax of CPNs and A Static Semantics . . . 334
Dynamic Semantics of Coloured Petri Nets . . . 337
12.4.4 Timed Coloured Petri Nets . . . 339
12.5 CEN Example: Work Flow System . . . 340
12.5.1 Project Planning . . . 340
Project Plans . . . 340
Project Plan Construction . . . 343
12.5.2 Project Activities . . . 344
Project Flow of Control: "Waves" and Traces 344
12.5.3 Project Generation . . . 351
Process Generation . . . 351
Channel Allocation . . . 352
12.6 CPN and RSL Examples: Superscalar Processor 354
12.6.1 Description . . . 354
12.6.2 Coloured Petri Net Model . . . 355
12.6.3 RSL Model: Superscalar Processor . 360
12.7 Discussion . . . 368
12.8 Bibliographical Notes . . . 369
12.9 Exercises . . . 370

13 MESSAGE AND LIVE SEQUENCE CHARTS . . . 371
Christian Krog Madsen is chief author of this chapter
13.1 Message Sequence Charts . . . 372
13.1.1 The Issues . . . 372
13.1.2 Basic MSCs (BMSCs) . . . 372
Informal Presentation . . . 372
An Example BMSC . . . 376
An RSL Model of BMSC Syntax . 377
13.1.3 High Level MSCs (HMSCs) . . . 379
An Informal Presentation . . . 379
An Example HMSC . . . 380
13.1.4 An RSL Model of HMSC Syntax . 380
13.1.5 MSCs are HMSCs . . . 381
13.1.6 Syntactic Well-formedness of MSCs 381
13.1.7 An Example: IEEE 802.11 Wireless Network . . . 387
Description . . . 387
An RSL Model of the IEEE 802.11 Example . . . 389
13.1.8 Semantics of Basic Message Sequence Charts . . . 395
13.1.9 Semantics of High Level Message Sequence Charts . . . 397
13.2 Live Sequence Charts: Informal Presentation . . 397
13.2.1 Live Sequence Chart Syntax . . . 397
Graphical Syntax of Live Sequence Charts . . . 397
13.2.2 A Live Sequence Chart Example, I 403
13.3 Process Algebra . . . 404
13.3.1 The Process Algebra PA_ϵ . . . 405
Signature . . . 405
Equations . . . 406
Derivability . . . 407
Reduction to Basic Terms . . . 408
13.3.2 Semantics of PA_ϵ . . . 412
13.3.3 The Process Algebra $PAc.\epsilon$ . . . 415
13.3.4 Semantics for PAc_ϵ . . . 418
13.4 Algebraic Semantics of Live Sequence Charts . 422
13.4.1 Textual Syntax of Live Sequence Charts . . . 422
13.4.2 Semantics of Live Sequence Charts 422
13.4.3 The Live Sequence Chart Example, II . . . 426
13.5 Relating Message Charts to RSL . . . 427
13.5.1 Types of Integration . . . 427
13.5.2 An RSL Subset . . . 428
Syntax . . . 428
Operational Semantics with Communication Behaviour . . . 429
13.5.3 Relating Live Sequence Charts to RSL . . . 430
Syntactical Restrictions . . . 430
Satisfaction Relation . . . 431
13.5.4 Checking Satisfaction . . . 437
13.5.5 Tool Support . . . 438
13.6 Communicating Transaction Processes (CTP) 438
13.6.1 Intuition . . . 438
13.6.2 Narration of CTPs . . . 439

CTP Diagrams: 439
CTP Processes 440
CTP Transaction Schemas 441
CTP Transaction Charts 441
Simple CTP Message Sequence Charts 441
Enabled CTP Transaction Charts 443
CTP Transitions 443
Enabled Versus Invoked Schemas and Charts 444
Details of Invocation and Execution 444
13.6.3 Formalisation of CTPs 445
Syntax of CTP Programs 445
Types of CTP Interpretation States 446
Interpretation Functions 447
13.7 Discussion 447
13.7.1 General 447
13.7.2 Principles, Techniques and Tools 448
13.8 Bibliographical Notes 449
13.9 Exercises 450

14 Statecharts 455
Christian Krog Madsen is chief author of this chapter
14.1 Introduction 455
14.2 A Narrative Description of Statecharts 456
14.3 An RSL Model of the Syntax of Statecharts 461
14.4 Examples 464
14.4.1 Railway Line Automatic Blocking 464
Narrative 465
Statecharts 466
14.4.2 Railway Line Direction Agreement System 468
Narrative 468
Internal Behaviour of LDAS (Statechart) 469
Internal Behaviour of Station A (Statechart) 470
Internal behaviour of Station B (Statechart) 470
External Behaviour (Live Sequence Chart) 471
14.4.3 Wireless Rain Gauge 471
Description 472
Statechart Model 473
RSL Model 473
14.5 A Process Algebra for Statecharts 477
14.5.1 SPL: The Statechart Process Language 478
14.5.2 Semantics of SPL 479
14.5.3 Equivalence for SPL Terms 479
14.6 Semantics of Statecharts 481
14.6.1 An SPL Semantics for Statecharts 481
14.6.2 Statechart Example 483
14.7 Relating Statecharts to RSL 485
14.7.1 Syntactical Restrictions 485
14.7.2 Satisfaction Relation 485
14.7.3 Checking Satisfaction 486
14.7.4 Tool Support 486
14.8 Discussion 487
14.8.1 General 487
14.8.2 Principles, Techniques and Tools 487
14.9 Bibliographical Notes 488
14.10 Exercises 488

15 QUANTITATIVE MODELS OF TIME 495
15.1 The Issues 495
15.1.1 Soft Temporalities 495
15.1.2 Hard Temporalities 496
15.1.3 Soft and Hard Real-Time 496
15.1.4 Examples — "Ye Olde Way"! 496
15.1.5 Structure of This Chapter 498
15.2 Temporal Logic 498
15.2.1 The Issues 499
15.2.2 A Philosophical Linguistics Background 499
15.2.3 Interval Temporal Logic, ITL 500
Intervals and Subintervals: $[c,d] \sqsubseteq [b,e]$ 500
Length of an Interval: $\lceil\cdot\rceil$ 501
The "Sometime" Modality: ◇ 501
The "'Always" Modality: □ 502
The Right Neighbourhood Expanding Modalities 502
The Left Neighbourhood Expanding Modalities 503
The "Chop" Modality 504
Defining ◇ in Terms of Chop 504
15.2.4 The Classic Temporal Operators: ◇, □ 505
15.3 The Duration Calculus 506
15.3.1 Examples, Part I 506
15.3.2 Some Basic Notions 507
Boolean States, State Assertions and Characteristic Functions 507
State Durations 508
15.3.3 Examples, Part II 510
Problem Description 510
Requirements: **L_Req** 512
15.3.4 The Syntax 514
Simple Expressions 514
State Expressions and Assertions 514
Durations and Duration Terms 515
Duration Formulas 515
Common Duration Formula Abbreviations 515
15.3.5 The Informal Semantics 516
15.3.6 Examples, Part III 517
The Elevator cum Lift Example: Design 517
The Railway Road-Rail Level Crossing Examples 519
Problem Description 520
Formalisation 521
Problem Description 524
Some Observations 527
15.3.7 Transitions and Events 528
15.3.8 Discussion: From Domains to Designs 533
15.4 TRSL: RSL with Timing 533
15.4.1 TRSL Design Criteria 533
The First TRSL Design Decision 534
The Second TRSL Design Decision 534
15.4.2 The TRSL Language 536
Syntax 536
Semantics 536
15.4.3 Another Gas Burner Example 537
15.4.4 Discussion 540
15.5 RSL with Timing and Durations 541
15.5.1 Review of TRSL 541
15.5.2 TRSL and Duration Calculus 542
Problem Description 543
DC Requirements 543
TRSL Specification 544
Satisfaction Relation 544
15.6 Discussion 545
15.6.1 General 545
15.6.2 Principles, Techniques and Tools 545
15.7 Bibliographical Notes 546
15.8 Exercises 546

Part VII LANGUAGE DEFINITIONS

16 SAL: Simple Applicative Language 551
16.1 A Caveat 552
16.2 The SAL Syntax 552
16.2.1 Informal Exposition of SAL Syntax 552
16.2.2 Formal Exposition of SAL Syntax 553
16.2.3 Comments 554
16.3 A Denotational Semantics 554
16.3.1 An Informal Semantics 554
16.3.2 A Formal Semantics 555
Semantic Types 555
Operator Meanings 555
Semantic Functions 556
16.3.3 Review of SAL Semantics, 1 557
16.3.4 Two Asides 558
Of Things to Come! 558
The Most Recent Error 558
16.4 A First-Order Applicative Semantics 560
16.4.1 Syntactic Types 560
16.4.2 Semantic Types 560
16.4.3 Abstraction Functions 561
16.4.4 Auxiliary Functions 562
16.4.5 Semantic Functions 563
16.4.6 Review 566
16.4.7 Review of SAL Semantics, 2 566
16.5 An Abstract, Imperative Stack Semantics 567
16.5.1 Design Decisions — Informal Motivation 567
16.5.2 Semantics Style Observations 568
16.5.3 Syntactic Types 568
16.5.4 Semantic Types 569
16.5.5 Abstraction Functions 569
16.5.6 Run-Time Functions 569
16.5.7 Semantic Functions 570
Two Invariants 570
[0] Interpret Programs 571
[1] Interpret Constant Expressions . 571
[2] Interpret Variable Expressions . 572
[3] Interpret Prefix Expressions . . . 572
[4] Interpret Infix Expressions 573
[5] Interpret Conditional Expressions 573
[6] Interpret Lambda Expressions . 574
[7] Interpret Simple Let Expressions 574
[8] Interpret Recursive Let Expressions 575
[9] Interpret Function Application Expressions 575
16.5.8 Review of SAL Semantics, 3 576
16.6 A Macro-Expansion Semantics 576
16.6.1 Analysis of Stack Semantics 577
Informal Design Description 580
[1] The Run-Time State 581
[2] Macro-expansion 581
[3] Realisation of CLOSures 582
[4] The Compile State — Compile-Time Specification 582
[5] The Compile State — The Dictionary 583
Execution 583
16.6.2 Syntactic Types 584
16.6.3 Compile–Time Types 584
16.6.4 Run-Time Semantic Types 584
16.6.5 Run-Time State 584
16.6.6 Run-Time Stack Operations 585
16.6.7 Run-Time Stack Search for Variable Values 585
16.6.8 Macro-Expansion Functions 586
[0] Program Macro-Expansion 586
[1] Constant Expression Macro-Expansion . . 587
[2] Variable Expression Macro-Expansion . . 587
[3] Prefix Expression Macro-Expansion . . 588
[4] Infix Expression Macro-Expansion . . 588
[5] Conditional Expression Macro-Expansion . . 588
[6] Lambda-Expression Macro-Expansion . . 589
[7] Simple Let Expression Macro-Expansion . . 590
Block Macro-Expansion 590
[8] Recursive Function/Let Expression Macro-Expansion . . 592
[9] Function Application Expression Macro-Expansion . . 593
16.6.9 Review of SAL Semantics, 4 593
16.7 ASM: An Assembler Language 594
16.7.1 Semantic Types 594
16.7.2 The Computer State 595
16.7.3 The Address Concept 595
16.7.4 Machine Instructions 596
16.7.5 Machine Semantics 598
Interpreting Code 598
Find Label Index 598
The Store Immediate and Store Instructions 599
The Load Immediate and Load Instructions 599
The Apply Function Instructions . . 599
The Unconditional Jump Instruction 601
The Conditional Jump Instruction . 601
The Register Move and Adjust Instructions 602
The Pack and Unpack Instructions 602
The Output Instruction 602
The Finish Instruction 603
16.7.6 Review of ASM 603
16.8 A Compiling Algorithm 603
16.8.1 Syntactic Types 604
16.8.2 Compile Time Types and State . . . 604
16.8.3 Compile-Time Dynamic Function . 604
16.8.4 Compile-Time Static Function 604
16.8.5 Run-Time Constant Values 605
16.8.6 Compilation Functions 605
[0] Program Compilation 605
[1] Constant Expression Compilation 606
[2] Variable Expression Compilation 606
[3] Prefix Expression Compilation . 608
[4] Infix Expression Compilation . . 608
[5] Conditional Expression Compilation 608
[6] Lambda-Expression Compilation 609
[7] Simple Let Expression Compilation 610
[*] Block Expression 610
[8] Recursive Function/Let Expression Compilation 611
[9] Function Application Expression Compilation 612
16.8.7 Review of Compiling Algorithm . . . 613
16.9 An Attribute Grammar Semantics 613
16.9.1 Abstract Syntactic Types 614
16.9.2 SAL BNF Grammar, 1 614
16.9.3 Node Attributes 615
16.9.4 Constants 615
16.9.5 Some Typographical Distinctions . 615
16.9.6 Compilation Functions 616
[0] Program Compilation 616
[1] Constant Expression Compilation 616
[2] Variable Expression Compilation 617
[3] Prefix Expression Compilation . 617
[4] Infix Expression Compilation . . 618
[5] Conditional Expression Compilation 618
[6] Lambda-Expression Compilation 619
[7] Simple Let Expression Compilation 619
[*] Block Expression Compilation . 620

[8] Recursive Function/Let Expression Compilation 620
[9] Function Application Expression Compilation 621
16.9.7 Review of Attribute Semantics, 1 . 622
16.10 Another Attribute Grammar Semantics 623
16.10.1 Abstract Syntactic Types........ 623
16.10.2 SAL BNF Grammar, 2 623
16.10.3 Global Variables 624
16.10.4 Constants 624
16.10.5 Node Attributes 624
16.10.6 Compilation Functions 625
[0] Program Compilation 625
[1] Constant Expression Compilation 625
[2] Variable Expression Compilation 625
[3] Prefix Expression Compilation . 626
[4] Infix Expression Compilation . . 626
[5] Conditional Expression Compilation 627
[6] Lambda-Expression Compilation 627
[7] Simple Let Expression Compilation 628
[•] Block Compilation 628
[8] Recursive Function / Let Expression Compilation 628
[9] Function Application Expression Compilation 629
16.10.7 Review of Attribute Semantics, 2 . 630
16.11 Discussion 630
16.11.1 General 630
16.11.2 Principles, Techniques and Tools . 631
16.12 Review and Bibliographical Notes 631
16.13 Exercises 633

17 SIL: Simple Imperative Language 635
17.1 The Background 635
17.2 Syntactic Types 636
17.2.1 Concrete, Schematic Syntax 636
17.2.2 Abstract Syntax 636
17.3 Imperative Denotational Semantics 637
17.3.1 Semantic Types 637
17.3.2 Auxiliary Semantic Functions 638
17.3.3 Semantic Functions 638
Procedure Denotations 638
Statement and Expression Function 639
17.4 Macro-Expansion Semantics.............. 639
17.4.1 Syntactic Types 640
17.4.2 Compile-Time Semantic Types ... 640
17.4.3 Run-Time Semantic Types 640
"Snapshot" of a Run-time State . . 640
Semantic Types 641
17.4.4 Run-Time State Declaration and Initialisation 641
17.4.5 Abstraction Functions 642
17.4.6 Macros 643
17.5 Discussion 644
17.5.1 General 644
17.5.2 Principles, Techniques and Tools . 644
17.6 Bibliographical Notes 645
17.7 Exercises 645

18 SMIL: Simple Modular, Imperative Language 647
18.1 Syntactic Types 647
18.2 A Denotational Semantics 648
18.2.1 Semantic Types 648
18.2.2 Auxiliary Functions 649
Static Functions 649
Temporal Functions 649
18.2.3 Semantic Functions 649
Semantic Function Types 650
Semantic Function Definitions.... 650
18.3 A Macro-Expansion Semantics............. 651
18.3.1 Run-Time Semantic Types 651
18.3.2 Compile/Run–time Semantic Types 652
18.3.3 Compile-Time Semantic Types ... 653
18.3.4 Semantic Functions 653
18.4 Discussion 655
18.4.1 General 655
18.4.2 Principles, Techniques and Tools . 655
18.5 Bibliographical Notes 656
18.6 Exercises 656

19 SPIL: Simple Parallel, Imperative Language 657
19.1 The Problem 657
19.2 Syntax 658
19.2.1 Informal Syntax 658
Process Expressions 658
Expressions and Statements 659
System Processes.............. 659
19.2.2 Formal Syntax 660
19.3 Process Concepts and Semantic Types 660
19.3.1 Syntactic Notions 661
Textual 661
19.3.2 Machines and Interpreters 662
19.3.3 Semantic Notions and Types..... 662
Actions 662
19.4 Process-Oriented Semantic Types 664
19.4.1 Unique Process Identifiers $\pi : \Pi$. 664
19.4.2 The Heap $\xi : \Xi$ 665
19.4.3 Input/Output Channel Bindings . . 666
19.4.4 Environments ρ : ENV 666
19.4.5 State Composition $\Psi, \Gamma, \Xi, \Sigma, \Omega$. 667
The Global State 668
19.5 Initial and Auxiliary Semantic Functions 668
19.5.1 Start Function 668
19.5.2 System Function 669
19.5.3 Bind and Allocate Functions 669
19.5.4 Free and Bound Functions....... 670
19.5.5 Distribute Function 670
19.5.6 Transition Loop 671
19.6 Semantic Functions 671
19.6.1 The *Next*-State Transition Function 671
19.6.2 The Assignment Statement...... 672
19.6.3 The *case* Statement 672
19.6.4 The *while* Loop 673
19.6.5 The *repeat until* Loop 673
19.6.6 Simple Input/Output Processes . . 674
19.6.7 The Parallel Process Command, || 675
19.6.8 The *stop* Process Technicality.... 675
19.6.9 The Process *call* Command...... 676
19.6.10 Internal Nondeterministic Processes 676
19.6.11 External Nondeterministic Processes 676
19.6.12 Nondeterministic Input/Output Processes 677
19.6.13 The Embedded System Process Command.................... 678
19.6.14 A *finish* Process Technicality 678
19.7 Discussion 678
19.7.1 General 678
19.7.2 Principles, Techniques and Tools . 679
19.8 Bibliographical Notes 679
19.9 Exercises 680

Part VIII CLOSING

20 Closing 685
20.1 A Summary 685
20.2 Conclusion: Volumes 1 and 2 686
20.3 Preview of Volume 3 686
20.4 "UML"-ising Formal Techniques 688

Part IX APPENDIXES

A Naming Convention 693

B Indexes 697
B.1 Symbols Index 698
B.1.1 General 698
B.1.2 Duration Calculus 698
B.1.3 Other Temporal Logics 698
B.1.4 Live Sequence Charts 699
B.1.5 Message Sequence Charts 699
B.1.6 Petri Nets 699
B.1.7 RSL 699
B.1.8 Statecharts 699
B.1.9 Time/Space 699
B.1.10 Abbreviations 700
B.2 Concepts Index 701
B.3 Characteriations and Definitions Index 720
B.4 Authors Index 722

References 727

4.3 Volume 3

Volume 3: Domains, Requirements and Software Design
http://www.imm.dtu.dk/~db/Software_Engineering/vol3/toc3.pdf

Preface vii
General vii
Brief Guide to Volume 3 vii

Part I OPENING

1 **The TripTych Paradigm** 3
1.1 Delineations of Software Engineering 3
1.1.1 "Old" Delineations 3
Friedrich L. Bauer, 1968 3
Ian Sommerville, 1980–2000 4
IEEE Std. 610.12–1990 4
David Lorge Parnas 4
Shari Lawrence Pfleeger, 2001 5
Carlo Ghezzi, Mehdi Jazayeri and Dino Mandrioli, 2002 5
Accreditation Board for Engineering and Technology (ABET) 5
Hans van Vliet, 2000 6
1.1.2 Our View: What Is Software Engineering? 6
1.2 The Triptych of Software Engineering 7
1.2.1 On Universes of Discourse and Domains 7
1.2.2 Domain Engineering 9
General 9
What Do We Expect From a Domain Description? 10
What a Domain Description Does Not Describe . 10
What a Domain Description Does Describe 10
Domain Phenomena and Concepts 10
Further Expectations From Domain Descriptions 21
Domain Descriptions as Bases for Domain Theories 21
More on Domain Engineering 21
1.2.3 Requirements Engineering 22
The Machine 22
The Machine Environment 22
General 23
Different Kinds of Requirements . . 23
More on Requirements Engineering 23
1.2.4 Software 24
1.2.5 Software Design 24
Software Architecture and Software Architecture Design 24
Component Structure and Component Design . 25
Software Architecture versus Component + Module Structure . . 25
Modules, Components and Systems 25
Systems, Design and Refinement . 27
Components and Modules, Design and Refinement 28
Code Design 29
More on Software Design 29
1.2.6 Discussion 29
1.3 Phases, Stages and Steps of Development . . . 30
Three Terms 30
A Principle of Separation of Concerns" 30
Linear, Cyclic and Parallel Development Activities 30
1.3.1 Phases of Software Development . 31
1.3.2 Stages and Steps of Development . 31
1.3.3 Domain Development 33
Stages of Domain Development . . 33
Steps of Domain Development . . . 36
1.3.4 Requirements Development 36
Stages of Requirements Development 36
Steps of Requirements Development 38
1.3.5 Computing Systems Design 38
Stages and Steps of Hardware Design 38
Stages of Software Design 39
Steps of Software Design 39
1.3.6 Discussion: Phases, Stages and Steps 39
Iterations of Phases, Stages and Steps 39
Concurrency of Phases, Stages and Steps 40
1.4 The Triptych Process Model — A First View . 41
1.4.1 The Concept of a Process Model . 41
1.4.2 The Triptych Process Model 42
1.5 Conclusion to Chapter 1 42
1.5.1 Summary 42
1.5.2 What Will Be Covered Later? 43
1.6 Bibliographical Notes 43
1.7 Exercises 43
1.7.1 On a Series of Software Developments 43
1.7.2 Introductory Remarks 49
1.7.3 The Exercises 49

2 **Documents** 51
2.1 Documentation Is All! 51
2.2 Kinds of Document Parts 52
2.2.1 General 52
2.2.2 What Is a Description? 52
Information Versus Description Versus Analysis 52
Descriptions, Prescriptions and Specifications 54
2.3 Deliverables 54
2.4 Informative Document Parts 55
2.4.1 Name, Place and Date 55
2.4.2 Partners 55
Clients 56
Developers 56
2.4.3 Current Situation, Needs, Ideas and Concepts 57
Current Situation 58
Needs 58
Ideas 58
Concepts and Facilities 59
2.4.4 Scope, Span and Synopsis 60
Scope 60
Span 60
Synopsis 61
2.4.5 Assumptions and Dependencies . . . 62
2.4.6 Implicit/Derivative Goals 63
2.4.7 Standards 63
Development Standards 63
Documentation Standards 64
Standards Versus Recommendations 64
Specific Standards 64
2.4.8 Contracts and Design Briefs 66
Contracts 66
Design Briefs 66
2.4.9 Logbook 67
2.4.10 Discussion of Informative Documentation 67
General 67

Methodological Consequences: Principle, Techniques and Tools 68
2.5 Descriptive Document Parts 68
Generalities 69
Specifics 69
Informal and Formal Document Parts 70
2.5.1 Rough Sketches 71
Pragmatics, Semantics and Syntax of Rough Sketches 71
Discussion 71
Methodological Consequences: Principles, Techniques and Tools 72
2.5.2 Terminologies 73
General 73
Pragmatics, Semantics and Syntax of Terminologies 73
Terminologisation — Continued 74
On Formal Terminologies cum Ontologies 74
Methodological Consequences: Principles, Techniques and Tools 75
2.5.3 Narratives 76
Pragmatics, Semantics and Syntax of Narratives 76
Discussion 78
Methodological Consequences: Principles, Techniques and Tools 78
2.5.4 Formal Descriptions 79
Pragmatics, Semantics and Syntax of Formalisations 79
On Formalisation 79
Methodological Consequences: Principles, Techniques and Tools 81
2.5.5 Discussion of Descriptive Documentation 82
2.6 Analytic Document Parts 82
Pragmatics, Semantics and Syntax 82
Categories of Analytical Document Parts 83
2.6.1 Concept Formation 83
Pragmatics, Semantics and Syntax 84
2.6.2 Validation 84
2.6.3 Verification, Model Checking, Testing 84
2.6.4 Theory Formation 85
2.6.5 Discussion of Analytic Documentation 85
General 85
Methodological Consequences: Principles, Techniques and Tools 85
2.7 Discussion 86
2.7.1 General 86
2.7.2 Summary of Chapter 86
Methodological Consequences: Principles, Techniques and Tools 87
2.8 Exercises 88
2.8.1 A Preamble 88
2.8.2 The Exercises 88

Part II CONCEPTUAL FRAMEWORK

3 Methods and Methodology 93
3.1 Method 93
3.2 Methodology 94
3.3 Method Constituents 95
3.3.1 Principle 95
3.3.2 Analysis 95
3.3.3 Construction (or Synthesis) 96
3.3.4 Techniques 96
3.3.5 Tools 96
3.4 Development Principles, Techniques and Tools 97
3.4.1 Some Metaprinciples 97
3.4.2 Some Principles, Techniques and Tools 98
Type Principle, Techniques and Tools 98
Function Principle, Techniques and Tools 98
Relation Principle, Techniques and Tools 99
Algebra Principle, Techniques and Tools 100
Logic Principle, Techniques and Tools 101
3.5 Discussion 101
3.6 Exercises 102

4 Models and Modelling 103
4.1 Introductory, Context-setting Remarks 103
4.1.1 Models Versus "Possible Worlds" . 103
4.1.2 On Models of a Specification 104
4.1.3 Modelling 104
4.1.4 Universes of Discourse 105
4.2 Model Attributes 105
4.2.1 Analogic, Analytic and Iconic Models 105
Principles and Techniques 106
Discussion 108
4.2.2 Descriptive and Prescriptive Models109
Examples 109
Principles and Techniques 110
Discussion 111
4.2.3 Extensional and Intensional Models 111
Principles and Techniques 112
Discussion 113
4.3 Roles of Models 113
4.4 The Modelling Principle 114
4.5 Discussion 114
4.6 Exercises 115

Part III DESCRIPTIONS: THEORY AND PRACTICE

5 Phenomena and Concepts 119
5.1 Introduction 119
5.2 Phenomena and Concepts 119
5.2.1 Physically Manifest Phenomena 120
5.2.2 Mentally Conceived Concepts 120
5.2.3 Categories of Phenomena and Concepts 120
5.2.4 Concrete and Abstract Concepts 121
5.2.5 Categories of Descriptions 121
5.2.6 What Is a Description? 122
5.3 Entities 123
5.3.1 Atomic Entities 123
5.3.2 Composite Entities 124
5.3.3 Subentities 124
5.3.4 Values, Mereology and Attributes . 124
5.3.5 Entity Mereology 125
5.3.6 Mereologies and Attributes 126
5.3.7 Model-Oriented Mereologies 126
5.3.8 Model-Oriented Attributes — An Aside 126
Atomic Types and Values 126
Other Attributes 126
5.3.9 Entity Properties 126
General 127
"Thing" Properties 127
5.3.10 Real Examples and Our Type System 127
Set Compounds 127

Cartesian Compounds 129
List Compounds 131
Map Compounds 132
5.3.11 A Type System 133
5.3.12 Type Constraints 134
5.3.13 Summary: Principles, Techniques and Tools 135
5.4 Functions 136
5.4.1 Function Signatures 137
5.4.2 Function Definition 138
5.4.3 Algorithms 140
5.5 Events and Behaviours 142
5.5.1 States, Actions, Events and Behaviours 142
5.5.2 Synchronisation and Communication 143
5.5.3 Processes 145
5.5.4 Traces 146
5.5.5 Process Definition Languages 146
5.6 Choice On Modelling Phenomena and Concepts 147
5.6.1 Qualitative Characteristics 147
5.6.2 Quantitative Characteristics 147
Information-Intensive Universe of Discourse 148
Function-Intensive Universe of Discourse 148
Event Intensive Universe of Discourse 148
Process Intensive Universe of Discourse 149
5.6.3 Principles, Techniques and Tools . 149
5.7 Discussion 151
5.7.1 Entities, Functions, Events and Behaviours 151
5.7.2 Intensity and Problem Frames 151
5.8 Bibliographical Notes 152
5.9 Exercises 152
5.9.1 A Preamble 152
5.9.2 The Exercises 152

6 On Defining and On Definitions 153
6.1 A Pragmatics of Definitions 155
6.1.1 Phenomena, Artifacts and Concepts 155
6.1.2 What Are Definitions? 156
6.1.3 The Nature of Concepts being Defined 156
6.1.4 Mathematical Definitions 157
6.1.5 Physical World Definitions 157
6.1.6 Formal Definitions 158
6.2 Varieties of Philosophy Definitions 158
6.2.1 Six Varietal Characterisations of Art 159
6.2.2 Discussion 160
6.2.3 A Possible Objection 161
6.3 Preliminary Discussion 161
6.4 A Syntax of Formal Definitions 161
6.4.1 Recognition and Reproduction . . . 163
6.4.2 Uniqueness and Identification 164
6.4.3 Ontological Terms 165
6.5 A Semantics of Formal Definitions 165
6.6 Discussion 166
6.6.1 General 166
6.6.2 Principles, Techniques and Tools . 166
6.7 Exercises 167

7 Jackson's Description Principles 171
7.1 Phenomena, Facts and Individuals 171
7.2 Designations 172
7.2.1 Some Observations 173
7.2.2 Formalisation 175
7.2.3 Observer Functions and Identification 176
7.2.4 Mathematical and Computing Entities 177
Mathematical Entities 177
Computing Entities 178
Awareness 178
Some Guidelines 178
A "Large" Example 179
Formal Modelling 181
7.2.5 Discussion: Designations 181
7.3 Explicit Definitions 182
7.3.1 Definitions: "The Narrow Bridge" . 182
7.3.2 Definition of Abstract, Intangible Concepts 183
7.3.3 How Much, How Little to Define? 184
7.3.4 Discussion: Definitions 184
7.4 Refutable Assertions 185
7.4.1 Designation and Definition Assertions 185
7.4.2 Analysis 186
7.4.3 "Dangling" Assertions 186
7.4.4 Discussion: Refutable Assertions . . 187
7.5 Discussion: Description Principles 188
7.6 Bibliographical Notes 188
7.7 Exercises 188
7.7.1 A Preamble 188
7.7.2 The Exercises 188

Part IV DOMAIN ENGINEERING

8 Overview of Domain Engineering 191
8.1 Introduction 191
8.2 A Review of *Why Domain Engineering?* 192
8.3 Overview of Part and Chapter 192
8.4 Domain Stakeholders and Their Perspectives . 193
8.5 Domain Acquisition and Validation 194
8.6 Domain Analysis and Concept Formation 194
8.7 Domain Facets 194
8.8 Auxiliary Stages of Domain Development 195
8.9 The Domain Model Document 195
8.9.1 A Preview of Things to Come 195
8.9.2 Contents of a Domain Model Document 196
8.10 Further Structure of This Part 196
8.11 Bibliographical Notes 197
8.12 Exercises 197

9 Domain Stakeholders 199
9.1 Introduction 199
9.2 Stakeholders 199
9.2.1 General Application Stakeholders . 200
9.2.2 Software Development Stakeholders 200
Turn-Key Software Development Stakeholders 201
Commercial Off-the-Shelf SW Development Stakeholders 201
9.2.3 Purpose of Listing Stakeholders . . 201
9.3 Stakeholder Perspectives 201
9.3.1 Perspectives of General Applications 202
9.3.2 Perspectives of Software Development 206
9.4 Discussion: Stakeholders and Their Perspectives 206
9.4.1 General 206
9.4.2 Principles, Techniques and Tools . 206
9.5 Exercises 207
9.5.1 Preamble 207
9.5.2 Assignments 207
9.5.3 Postlude 208

10 Domain Attributes ... 209
- 10.1 Introduction ... 209
- 10.2 Continuity, Discreteness and Chaos ... 210
  - 10.2.1 Time ... 210
  - 10.2.2 Continuity ... 210
  - 10.2.3 Discreteness ... 212
    - On a Notion of State ... 213
    - Methodological Consequences ... 213
    - Hybridicity ... 214
    - Methodological Consequences ... 218
  - 10.2.4 Chaos ... 218
    - Methodological Consequences ... 219
  - 10.2.5 Discussion ... 220
- 10.3 Statics and Dynamics ... 220
  - 10.3.1 Static Phenomena and Concepts ... 221
    - Static Attributes ... 221
  - 10.3.2 Dynamic Phenomena and Concepts ... 223
    - Inert Phenomena and Concepts ... 224
    - Active Dynamic Attribute ... 226
    - Reactive Dynamics Attribute ... 237
    - Discussion ... 239
- 10.4 Tangibility and Intangibility ... 239
  - 10.4.1 Humanly Tangible Phenomena ... 239
  - 10.4.2 Otherwise Physically Tangible Phenomena ... 242
  - 10.4.3 Intangible Phenomena ... 243
  - 10.4.4 Discussion ... 243
- 10.5 One, Two, ..., Dimensionality ... 244
  - 10.5.1 Zero Dimensionality ... 245
  - 10.5.2 One Dimensionality ... 245
  - 10.5.3 Multidimensionality ... 246
  - 10.5.4 Discussion ... 247
- 10.6 Discussion ... 247
- 10.7 Bibliographical Notes ... 247
- 10.8 Exercises ... 248
  - 10.8.1 A Preamble ... 248
  - 10.8.2 The Exercises ... 248

11 Domain Facets ... 249
- 11.1 Introduction ... 249
  - 11.1.1 Separation of Concerns ... 251
  - 11.1.2 Discussion of the Separation Principle ... 251
  - 11.1.3 Structure of Chapter ... 251
- 11.2 Domain Facilitators: Business Processes ... 251
  - 11.2.1 Business Processes ... 252
  - 11.2.2 Overall Principles ... 255
  - 11.2.3 Informal and Formal Examples ... 256
  - 11.2.4 Discussion ... 260
  - 11.2.5 Summary ... 261
  - 11.2.6 Reminder ... 261
- 11.3 Domain Intrinsics ... 262
  - 11.3.1 Overall Principles ... 262
  - 11.3.2 Conceptual Versus Actual Intrinsics ... 266
  - 11.3.3 Methodological Consequences ... 268
  - 11.3.4 Discussion ... 268
  - 11.3.5 Utter Barebone Intrinsics ... 268
  - 11.3.6 Reminder ... 269
- 11.4 Domain Support Technologies ... 269
  - 11.4.1 Overall Principles ... 269
  - 11.4.2 Methodological Consequences ... 272
  - 11.4.3 Discussion ... 273
  - 11.4.4 Reminder ... 273
- 11.5 Domain Management and Organisation ... 274
  - 11.5.1 Overall Principles ... 274
  - 11.5.2 A Conceptual Analysis, I ... 276
  - 11.5.3 Methodological Consequences, I+II ... 276
  - 11.5.4 Conceptual Analysis, II ... 276
  - 11.5.5 Methodological Consequences, III ... 278
  - 11.5.6 Discussion ... 279
  - 11.5.7 Reminder ... 279
- 11.6 Domain Rules and Regulations ... 280
  - 11.6.1 Overall Principles ... 280
  - 11.6.2 Methodological Consequences ... 281
  - 11.6.3 Rules and Regulation Languages ... 283
  - 11.6.4 Principles and Techniques ... 284
  - 11.6.5 Reminder ... 284
- 11.7 Domain Scripts ... 285
  - 11.7.1 The Description of Scripts ... 285
    - Routine Headers ... 296
    - Example Statements ... 297
    - Example Expressions ... 297
    - Abstract Syntax for Syntactic Types ... 298
    - Semantic Types Abstract Syntax ... 299
    - Semantic Functions ... 300
  - 11.7.2 Methodological Consequences ... 305
  - 11.7.3 Reminder + More ... 305
- 11.8 Domain Human Behaviour ... 306
  - 11.8.1 Overall Principles ... 306
  - 11.8.2 Methodological Consequences ... 310
  - 11.8.3 Human Behaviour and Knowledge Engineering ... 312
  - 11.8.4 Discussion ... 312
  - 11.8.5 Reminder ... 313
- 11.9 Other Domain Facets? ... 313
- 11.10 Composition of Models ... 313
  - 11.10.1 Collating Facet Descriptions ... 314
    - General ... 314
    - A Comprehensive Narrative ... 314
    - From Big Lies via Smaller Lies to the Truth ... 314
  - 11.10.2 Technical Issues ... 315
- 11.11 Exercises ... 315
  - 11.11.1 A Preamble ... 315
  - 11.11.2 The Exercises ... 315

12 Domain Acquisition ... 317
- 12.1 Introduction ... 317
  - 12.1.1 Domain Facts ... 318
  - 12.1.2 Elicitation of Domain Facts ... 318
  - 12.1.3 Recording Domain Facts ... 318
  - 12.1.4 Indexing Domain Description Sketches ... 319
- 12.2 The Acquisition Process ... 320
  - 12.2.1 Stakeholder Liaison ... 321
  - 12.2.2 Elicitation Studies ... 321
  - 12.2.3 Elicitation Interviews ... 323
  - 12.2.4 Elicitation Questionnaires ... 323
    - General Guidelines: Questionnaire Structure and Contents ... 323
    - Special Guidelines: Questionnaire Structure and Contents ... 324
  - 12.2.5 Elicitation Reports ... 326
- 12.3 Discussion ... 326
  - 12.3.1 Concept and Process Review ... 326
  - 12.3.2 Process Iteration ... 327
  - 12.3.3 Delineation: Acquisition and Analysis ... 327
  - 12.3.4 Principles, Techniques and Tools ... 327
- 12.4 Exercises ... 328
  - 12.4.1 A Preamble ... 328
  - 12.4.2 The Exercises ... 328

13 Domain Analysis and Concept Formation ... 329
- 13.1 Introduction ... 329
- 13.2 Concept Formation ... 329
  - 13.2.1 Simply Abstracted Concepts ... 330
  - 13.2.2 Breakthrough Abstracted Concepts ... 331
- 13.3 Consistencies, Conflicts and Completeness ... 332
  - 13.3.1 Inconsistencies ... 333
  - 13.3.2 Conflicts ... 333
  - 13.3.3 Incompleteness ... 333
  - 13.3.4 Looseness and Nondeterminism ... 334
- 13.4 From Analysis to Synthesis ... 334
- 13.5 Discussion ... 335
  - 13.5.1 General ... 335
  - 13.5.2 Principles, Techniques and Tools ... 335
- 13.6 Bibliographical Notes ... 336
- 13.7 Exercises ... 336
  - 13.7.1 A Preamble ... 336
  - 13.7.2 The Exercises ... 337

14 Domain Verification and Validation 339
14.1 Introduction 339
14.2 Domain Verification 340
14.2.1 Informal Reasoning 341
14.2.2 Testing 341
14.2.3 Formal Proofs 341
14.2.4 Model Checking 342
14.3 Domain Validation 342
14.3.1 The Domain Validation Documents 342
14.3.2 The Domain Validation Process 343
14.3.3 Domain Development Iterations 343
14.4 Discussion 344
14.4.1 General 344
14.4.2 Principles, Techniques and Tools 344
14.5 Exercises 344
14.5.1 A Preamble 344
14.5.2 The Exercises 345

15 Towards Domain Theories 347
15.1 Introduction 347
15.2 What Is a Domain Theory? 348
15.3 Example Statements of Domain Theories 348
15.4 Possible Domain Theories 350
15.5 How Do We Establish a Theory? 351
15.6 Purpose of a Domain Theory 352
15.7 Summary Principles, Techniques and Tools 352
15.8 Bibliographical Notes 352
15.9 Exercises 353
15.9.1 A Preamble 353
15.9.2 The Exercises 353

16 The Domain Engineering Process Model 355
16.1 Introduction 355
16.2 Review of Domain Development 355
16.3 Review of Domain Documents 357
16.4 Discussion 358

Part V REQUIREMENTS ENGINEERING

17 Overview of Requirements Engineering 361
17.1 Introduction 364
17.1.1 Further Characterisation of 'Requirement' 365
17.1.2 The "Machine" 365
17.2 Why Requirements, and for What? 366
17.2.1 Why Requirements? 366
17.2.2 Requirements for What? 366
17.2.3 What Does 'Implements' Mean? 366
17.3 Getting Started on Requirements Development 367
17.3.1 Initial Informative Documentation 367
17.3.2 Requirements Eurekas 368
Initial Eureka of Requirements 369
Ongoing Eureka of Requirements 369
A Systematic Source of Requirements Eurekas 369
Placement of Initial Requirements Eurekas 370
17.3.3 Pragmatic Prescriptive Documentation 370
17.3.4 Planning Requirements Development 371
17.4 On Domains, Requirements and the Machine 371
17.5 Overview: Requirements Engineering Stages 373
17.6 The Requirements Document 374
17.6.1 A Preview of Things to Come 374
17.6.2 Contents of a Requirements Document 374
17.6.3 Comments on Requirements Documents 375
17.7 The Structure of the Rest of the Part 375
17.8 Bibliographical Notes 375
17.9 Exercises 375
17.9.1 A Preamble 375
17.9.2 The Exercises 376

18 Requirements Stakeholders 377
18.1 Introduction 377
18.2 General Application Stakeholders 378
18.3 COTS Software House Stakeholders 378
18.3.1 General 378
18.3.2 "Corporate Knowledge" 379
18.3.3 Classes of Domain-Specific Requirements 379
18.3.4 Generic COTS Software Stakeholder Perspective 379
18.4 Discussion 379
18.4.1 General 379
18.4.2 Principles, Techniques and Tools 380
18.5 Exercises 380
18.5.1 Preamble 380
18.5.2 The Exercises 380

19 Requirements Facets 383
19.1 Introduction 384
19.2 Rough Sketching and Terminology 384
19.2.1 Initial Requirements Modelling 384
19.2.2 Rough-Sketch Requirements 385
Entities 385
Functions 387
Events 388
Behaviours 388
19.2.3 Requirements Terminology 392
19.2.4 Systematic Narration 397
19.3 Business Process Reengineering Requirements 398
19.3.1 Michael Hammer's Ideas on BPR 398
19.3.2 What Are *BPR Requirements?* 399
19.3.3 Overview of BPR Operations 400
19.3.4 BPR and the Requirements Document 400
Requirements for New Business Processes 400
Place in Narrative Document 400
Place in Formalisation Document 401
19.3.5 Intrinsics Review and Replacement 401
19.3.6 Support Technology Review and Replacement 401
19.3.7 Management and Organisation Reengineering 402
19.3.8 Rules and Regulation Reengineering 403
19.3.9 Human Behaviour Reengineering 403
19.3.10 Script Reengineering 404
19.3.11 Discussion: Business Process Reengineering 404
Who Should Do the Business Process Reengineering? 404
General 405
19.4 Domain Requirements 405
19.4.1 Domain-to-Requirements Operations 405
19.4.2 Domain Reqs. and the Reqs. Document 406
Requirements for Functionalities 406
Place in Narrative Document 406
Place in Formalisation Document 407
19.4.3 A Domain Example 407
19.4.4 Domain Projection 408
19.4.5 Domain Determination 409
19.4.6 Domain Instantiation 413
19.4.7 Domain Extension 414
19.4.8 Domain Requirements Fitting 415
19.4.9 Discussion: Domain Requirements 417
19.5 Interface Requirements 417
19.5.1 Shared Phenomena and Concept Identification 418
19.5.2 Interface Requirements Facets 418
19.5.3 Interface Reqs. and the Reqs. Document 419
Requirements for "Input/Output" 419
Place in Narrative Document 419
Place in Formalisation Document 420
19.5.4 Shared Data Initialisation 420
19.5.5 Shared Data Refreshment 421
19.5.6 Computational Interface Requirements 422

19.5.7 Man-Machine Dialogue 423
19.5.8 Man-Machine Physiological Interface 423
19.5.9 Machine-Machine Dialogue 431
19.5.10 Discussion: Interface Requirements 432
Dialogue Prescription Techniques and Tools 432
General 432
Special Principles and Techniques . 432
19.6 Machine Requirements 433
19.6.1 Machine Requirements Facets 434
19.6.2 Machine Reqs. and the Reqs. Document 434
Requirements for "The Machine Only" 434
Place in Narrative and Formalisation Document 434
19.6.3 Performance Requirements 435
19.6.4 Dependability Requirements 437
Accessibility 439
Availability 439
Integrity 440
Reliability 440
Safety 440
Security 440
Robustness 441
19.6.5 Maintenance Requirements 442
Adaptive Maintenance 442
Corrective Maintenance 442
Perfective Maintenance 443
Preventive Maintenance 443
Extensional Maintenance 443
19.6.6 Platform Requirements 444
Development Platform 444
Execution Platform 444
Maintenance Platform 444
Demonstration Platform 445
Discussion 445
19.6.7 Documentation Requirements 445
19.6.8 Discussion: Machine Requirements 445
19.7 Full Requirements Facets Document 446
19.8 Discussion: Requirements Facets 446
19.8.1 General 446
19.8.2 Principles, Techniques and Tools . 446
19.9 Bibliographical Notes 447
19.10 Exercises 447
19.10.1 A Preamble 447
19.10.2 The Exercises 447

20 Requirements Acquisition 451
20.1 Requirements Acquisition Versus Domain Models 451
20.2 Domain Model-Based Requirements Acquisition452
20.2.1 Domain Requirements Acquisition, A Preview 452
Assumptions 452
Domain Requirements Acquisition, Basic Steps 452
20.2.2 Remaining Requirements Acquisition, A Preview 453
Assumptions 453
Machine Requirements Acquisition, Basic Steps 453
Interface Requirements Acquisition, Basic Steps 453
20.2.3 Further Issues 454
20.3 Overview of Concepts 454
20.3.1 Requirements 455
20.3.2 Elicitation of Requirements 455
20.3.3 Recording Requirements 455
Requirements Index 455
Requirements Type 455
Requirements Units 456
20.3.4 Indexing Requirements Prescription Sketches 457
20.4 The Acquisition Process 457
20.4.1 Stakeholder Liaison 458
20.4.2 Elicitation Studies 459
20.4.3 Elicitation Interviews 459
20.4.4 Elicitation Questionnaires 460
General Guidelines: Questionnaire Structure and Contents 460
Special Guidelines: Questionnaire Structure and Contents 461
20.4.5 Elicitation Reports 463
20.5 Discussion 464
20.5.1 Concept and Process Review 464
20.5.2 Process Iteration 464
20.5.3 Delineation: Acquisition and Analysis 464
20.5.4 Principles, Techniques and Tools . 464
20.6 Exercises 465
20.6.1 A Preamble 465
20.6.2 The Exercises 465

21 Requirements Analysis and Concept Formation 467
21.1 Introduction 467
21.2 Concept Formation 469
21.3 Consistencies, Conflicts, and Completeness ... 469
21.3.1 Inconsistencies 469
21.3.2 Conflicts 470
21.3.3 Incompleteness 470
21.3.4 Looseness and Nondeterminism ... 471
21.4 From Analysis to Synthesis 471
21.5 Discussion 471
21.5.1 General 471
21.5.2 Principles, Techniques and Tools . 472
21.6 Bibliographical Notes 472
21.7 Exercises 473
21.7.1 A Preamble 473
21.7.2 The Exercises 473

22 Requirements Verification and Validation 475
22.1 Introduction 475
22.2 Requirements Verification 476
22.2.1 Informal Reasoning 477
22.2.2 Testing 477
22.2.3 Formal Proofs 478
22.2.4 Model Checking 478
22.3 Requirements Validation 478
22.3.1 The Requirements Validation Documents 479
22.3.2 The Requirements Validation Process 479
22.3.3 Requirements Development Iterations 479
22.4 Discussion 480
22.4.1 General 480
22.4.2 Principles, Techniques and Tools . 480
22.5 Bibliographical Notes 481
22.6 Exercises 481
22.6.1 Preamble 481
22.6.2 The Exercises 481

23 Requirements Satisfiability and Feasibility 483
23.1 Introduction 483
23.2 Satisfaction Study 484
23.2.1 Correct (Validated) Requirements Document 484
23.2.2 Unambiguous Requirements Document 484
23.2.3 Complete Requirements Document 484
23.2.4 Consistent Requirements Document484
23.2.5 Stable Requirements Document . . 484
23.2.6 Verifiable Requirements Document 485
23.2.7 Modifiable Requirements Document 485
23.2.8 Traceable Requirements Document 485
23.2.9 Faithful Requirements Document . 485
23.2.10 Discussion of Satisfiability 486
23.3 Technical Feasibility Study 486
23.3.1 Feasibility of Business Process Reengineering 486
23.3.2 Feasibility of Hardware 486
23.3.3 Feasibility of Software 486
23.3.4 Discussion of Technical Feasibility. 487
23.4 Economic Feasibility Study 487
23.4.1 Feasible Development Costs 487
23.4.2 Feasible Write-off Costs 487
23.4.3 Gains Outweigh Costs? 487
23.4.4 Discussion of Economic Feasibility 488
23.5 Compliance with Implicit/Derivative Goals . . . 488
23.5.1 Review of Implicit/Derivative Goals488
23.5.2 Discussion of Implicit/Derivative Goals 488
23.6 Discussion 488
23.6.1 General 488
23.6.2 Principles, Techniques and Tools . 489
23.7 Exercises 490
23.7.1 A Preamble 490
23.7.2 The Exercises 490

24 The Requirements Engineering Process Model 493
24.1 Introduction 493
24.2 Review of Requirements Development 493
24.3 Review of Requirements Documents 494
24.4 The Repeat Table of Contents Listing 494
24.5 Discussion 495

Part VI COMPUTING SYSTEMS DESIGN

25 Hardware/Software Codesign 499
25.1 Introduction — On Architecture 499
25.2 Hardware Components and Modules 500
25.3 Software Components and Modules 500
25.4 Hardware/Software Codesign 500
25.5 Stepwise Refinement of Architectures 501
25.6 Discussion 501
25.7 Principles, Techniques and Tools 501

26 Software Architecture Design 503
26.1 Introduction 503
26.2 Initial Domain Requirements Architecture 504
26.3 Initial Machine Requirements Architecture . . . 506
26.4 Analysis of Some Machine Requirements 508
26.4.1 Performance 508
26.4.2 Availability 508
26.4.3 Accessibility 509
26.4.4 Adaptive Maintainability 509
26.5 Prioritisation of Design Decisions 509
26.6 Corresponding Designs 509
26.6.1 Design Decision wrt. Performance . 510
26.6.2 Design Decision wrt. Availability . . 511
26.6.3 Design Decision wrt. Accessibility . 512
26.6.4 Design Decision wrt. Adaptability . 514
26.7 Discussion 515
26.7.1 General 515
26.7.2 Principles and Techniques 516
26.8 Bibliographical Notes 517
26.9 Exercises 517
26.9.1 A Preamble 517
26.9.2 The Exercises 517

27 A Case Study in Component Design 519
27.1 Overview Introduction 519
27.1.1 System Complexity 519
27.1.2 Proposed Remedies 520
27.1.3 Stepwise Development 520
27.1.4 Stagewise Iteration 521
27.2 Overview of Example 521
27.3 Methodology Overview 522
27.3.1 Principles 522
27.3.2 Techniques 523
27.4 Step 0: Files and Pages 524
27.4.1 A “Snapshot” 524
27.4.2 An Abstract Formal Model 524
27.4.3 Abstract Versus Concrete Basic Actions 525
27.4.4 Concrete Actions 527
27.5 Step 1: Catalogue, Disk and Storage 527
27.5.1 Catalogue Directories 527
Data Structure 528
Invariant 529
27.5.2 Abstraction 530
27.5.3 Actions 531
Action Signatures 531
Create and Erase File Actions 532
Put Page Action 532
Get and Delete Page Actions 533
27.5.4 Adequacy and Sufficiency 533
Adequacy 534
Sufficiency 534
27.5.5 Correctness 534
Comparable Results 534
The Correctness Statement 535
27.6 Step 2: Disks 536
27.6.1 Data Refinement 536
27.6.2 Disk Type 536
A “Snapshot” 536
27.6.3 FS0, FS1 and FS2 Types 536
Concrete Semantic Types 536
27.6.4 Disk Type Invariant 537
27.6.5 Disk Type Abstraction 538
27.6.6 Adequacy, Sufficiency, Operations and Correctness 538
27.7 Step 3: Caches 538
27.7.1 Technology Considerations 538
27.7.2 Cached Directory and Page Access 539
27.7.3 Invariance 540
27.7.4 Abstraction 541
27.7.5 Actions 541
Open and Close Actions 541
Create and Put Actions 542
Erase, Get, and Delete Actions . . . 542
27.7.6 Adequacy, Sufficiency and Correctness 543
27.8 Step 4: Storage Crashes 543
27.8.1 Storage and Disk 543
27.8.2 Concrete Semantic Types 544
27.8.3 Invariance 544
27.8.4 Consistent Storages and Disks 544
Consistent Storage 544
Consistent Disk 545
27.8.5 Abstractions 546
27.8.6 Garbage Collection 546
27.8.7 New Actions 547
Check and Crash Actions 547
27.8.8 Some Previous Commands 547
Open and Close Actions 547
Put Action 548
27.9 Step 5: Flattening Storage and Disks 548
27.9.1 “Flat” Storage and Disk 548
27.9.2 “The Rest” 549
27.10 Step 6: Disk Space Management 549
27.10.1 The Issue 549
27.10.2 “The Rest” 550
27.11 Discussion 551
27.11.1 General 551
27.11.2 Principles and Techniques 551
27.12 Bibliographical Notes 552
27.13 Exercises 552
27.13.1 A Preamble 552
27.13.2 The Exercises 552

28 Domain-Specific Architectures . . . 555
28.1 Introduction . . . 555
28.1.1 General . . . 555
28.1.2 Some Definitions . . . 556
28.1.3 On Architectures . . . 556
28.1.4 Problem Frames . . . 557
28.1.5 Chapter Structure . . . 558
28.2 Translator Architectures . . . 558
28.2.1 Translator Domain . . . 559
28.2.2 Translator Requirements . . . 560
28.2.3 Translator Design . . . 560
Translator Functions . . . 561
A Multipass Compiler . . . 561
28.2.4 Process Graph for Translator Development . . . 562
28.3 Information Repository Architectures . . . 568
28.3.1 Information Repository Domain . . . 569
28.3.2 Information Repository Requirements . . . 570
28.3.3 Information Repository Design . . . 571
The Role of A Database Management System 571
A Relational Database Management System Architecture 572
Other Database Management System Architectures and Discussion . . . 582
28.4 Client/Server Architectures . . . 582
28.4.1 Client/Server Domain/Requirements Models . . . 582
Single-Client, Single-Server Domain/Requirements Architecture . . . 583
Multiple-Client, Multiple-Server Domain/Requirements Model . . . 584
28.4.2 Some Meta-RSL/CSP Constructs . . 585
Motivation and Justification . . . 586
Three Process Interaction Macros . 586
'Meaning' of the Three Process Interaction Macros . 586
Pragmatics of Macros . . . 587
28.4.3 Single-Client, Single-Server Model 587
28.4.4 Multiple-Client, Single-Server Model . . . 591
28.4.5 Client/Server Event Manager Model . . . 592
28.4.6 Discussion . . . 600
28.5 Workpiece Architectures . . . 600
28.5.1 Workpiece Domain . . . 601
28.5.2 Workpiece Requirements . . . 601
28.5.3 Workpiece Systems Design . . . 604
28.6 Reactive System Architectures . . . 604
28.6.1 Reactive Systems Domain . . . 604
28.6.2 Reactive Systems Control Requirements . . . 606
28.6.3 Reactive Systems Control Design . 607
28.6.4 Discussion of Reactive Systems Design . . . 608
28.7 Connection Frame . . . 608
28.7.1 Connection Domain . . . 609
28.7.2 Connection Requirements . . . 610
28.7.3 Connection Systems Design . . . 612
28.8 Discussion . . . 612
28.8.1 General . . . 612
28.8.2 Principles, Techniques and Tools . 612
28.9 Exercises . . . 613
28.9.1 A Preamble . . . 613
28.9.2 The Exercises . . . 613

29 Etcetera: Coding and All That! . . . 617
29.1 From Formal Specification to Programming . . 617
29.1.1 From Specifications to Programs . 618
29.1.2 From Abstract Types to Data Structures . . . 618
29.1.3 From Applicative to Imperative Programs . . . 618
29.1.4 Translations into Concurrent Programs . . . 619
29.1.5 From RSL to SML, Java, C# and Other Languages . . . 619
29.2 The Beauty of Programming . . . 619
Art, Discipline, Craft, Science, Logic and Practice . . . 619
29.3 Programming Practices . . . 620
29.3.1 Structured Programming . . . 620
29.3.2 Extreme Programming . . . 620
29.3.3 Object-Oriented cum UML Programming . . . 621
29.3.4 Chief Programmer Programming . . 621
29.4 Confidence-Building Software Development . . . 622
29.4.1 When to Verify, Model Check and Test . . . 622
The V Diagram . . . 623
Order of Tests . . . 623
Discovery of Errors, Misinterpretations and Misunderstandings . 623
Cost of Corrective Maintenance . . 624
Impossibility of Certain Corrective Maintenance . . . 624
29.4.2 Demo $\rightarrow$ Skeleton $\rightarrow$ Prototype $\rightarrow$ System . . . 624
29.5 Verification, Model Checking and Testing . . . 627
29.5.1 Verification . . . 628
29.5.2 Model Checking . . . 630
29.5.3 Testing . . . 630
Test Objectives . . . 631
Test Strategy . . . 631
Test Coverage . . . 631
Test Cases and Test Suites . . . 632
Test Adequacy Criteria — Test Requirements . . . 632
Classical Testing Approaches . . . 632
Formal Specification-Based Testing Approaches . 633
29.5.4 Discussion . . . 633
29.6 Discussion . . . 633
29.7 Exercises . . . 634
29.7.1 A Preamble . . . 634
29.7.2 The Exercises . . . 634

30 The Computing Systems Design Process Model . . . 635
30.1 Introduction . . . 635
30.2 Review of Software Design . . . 635
30.2.1 A Process Model . . . 636
30.2.2 Discussion . . . 637
30.3 Review of Software Design Documents . . . 638
30.4 Discussion . . . 640

Part VII CLOSING

31 The TripTych Development Process Model . . . 643
31.1 Phase Process Models . . . 643
31.2 Phase Documentation Table of Contents . . . 647
31.3 Conclusion . . . 649

32 Finale 651
- 32.1 Informal and Formal Software Engineering . . . 651
  - 32.1.1 Informal Software Engineering 652
  - 32.1.2 Formal Software Engineering 652
  - 32.1.3 Conclusion 652
- 32.2 Myths and Commandments of Formal Methods652
  - 32.2.1 First Seven Myths 653
  - 32.2.2 Seven More Myths 654
  - 32.2.3 Ten Formal Methods Commandments 655
- 32.3 FAQ: Frequently Asked Questions 657
  - 32.3.1 General 657
  - 32.3.2 Domains 658
  - 32.3.3 Requirements 660
- 32.4 Research and Tool Development 660
  - 32.4.1 Evolving Principles, Techniques and Tools 660
  - 32.4.2 Grand Challenges 661
    - Three Dimensions of Grand Challenges 661
    - Integration of Formal Techniques . 661
    - Trustworthy Evolutionary Systems Development 661
    - Domain Theories 662
    - On the Nature of “Grand Challenges” 662
- 32.5 Application Areas 663
  - 32.5.1 Additional Areas 663
  - 32.5.2 The Examples 664
- 32.6 Closing Remarks 665

Part VIII APPENDIXES

A An RSL Primer 669
- A.1 Types 669
  - A.1.1 Type Expressions 669
  - A.1.2 Type Definitions 671
    - Concrete Types: 671
    - Subtypes 671
    - Sorts (Abstract Types) 672
- A.2 The RSL Predicate Calculus 672
  - A.2.1 Propositional Expressions 672
  - A.2.2 Simple Predicate Expressions 672
  - A.2.3 Quantified Expressions 673
- A.3 Concrete RSL Types 673
  - A.3.1 Set Enumerations 673
  - A.3.2 Cartesian Enumerations 674
  - A.3.3 List Enumerations 674
  - A.3.4 Map Enumerations 675
  - A.3.5 Set Operations 675
  - A.3.6 Cartesian Operations 677
  - A.3.7 List Operations 678
  - A.3.8 Map Operations 679
- A.4 λ-Calculus and Functions 681
  - A.4.1 The λ-Calculus Syntax 681
  - A.4.2 Free and Bound Variables 682
  - A.4.3 Substitution 682
  - A.4.4 α-Renaming and β-Reduction . . . 682
  - A.4.5 Function Signatures 683
  - A.4.6 Function Definitions 683
- A.5 Further Applicative Expressions 684
  - A.5.1 Let Expressions 684
  - A.5.2 Conditionals 685
  - A.5.3 Operator/Operand Expressions . . 686
- A.6 Imperative Constructs 686
  - A.6.1 Variables and Assignment 686
  - A.6.2 Statement Sequences and **skip** . . 686
  - A.6.3 Imperative Conditionals 687
  - A.6.4 Iterative Conditionals 687
  - A.6.5 Iterative Sequencing 687
- A.7 Process Constructs 687
  - A.7.1 Process Channels 687
  - A.7.2 Process Composition 688
  - A.7.3 Input/Output Events 688
  - A.7.4 Process Definitions 688
- A.8 Simple RSL Specifications 689

B Indexes 691
- B.1 Concepts Index 692
- B.2 Characterisations and Definitions Index 709
- B.3 Authors Index 713

References 717

References

1. Andrew W. Appel. *Modern Compiler Implementation in Java.* Cambridge University Press, January 1998. ISBN: 0521583888.
2. C.A. Gunther. *Semantics of Programming Languages.* The MIT Press, Cambridge, Mass., USA, 1992.
3. Peter D. Mosses. *Action Semantics.* Cambridge University Press: Tracts in Theoretical Computer Science, 1992. .
4. Wolfang Reisig. *A Primer in Petri Net Design.* Springer Verlag, March 1992. 120 pages.
5. Wolfgang Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs in Theoretical Computer Science.* Springer Verlag, May 1985.
6. Wolfgang Reisig. *Elements of Distributed Algorithms: Modelling and Analysis with Petri Nets.* Springer Verlag, December 1998. xi + 302 pages.
7. John C. Reynolds. *Theories of Programming Languages.* Cambridge University Press, Edinburgh Building, Shaftesbury Road, Cambridge CB2 2RU, England, 10 December 1998. ISBN 0521594146, 512 Pages
8. John C. Reynolds. *The Semantics of Programming Languages.* Cambridge University Press, 1999.
9. David A. Schmidt. *Denotational Semantics: a Methodology for Language Development.* Allyn & Bacon, 1986.
10. Robert Tennent. *The Semantics of Programming Languages.* Prentice–Hall Intl., 1997.
11. Reinhard Wilhelm. *Compiler Design.* Addison Wesley, 1995. ISBN: 0-201-42290-5. .
12. G. Winskel. *The Formal Semantics of Programming Languages.* The MIT Press, Cambridge, Mass., USA, 1993.

[illegible]

Dines Bjørner
National University of Singapore, May 2005