

Monographs in Theoretical Computer Science

An EATCS Series

Editors: J. Hromkovič M. Nielsen

Founding Editors: W. Brauer G. Rozenberg A. Salomaa

On behalf of the European Association
for Theoretical Computer Science (EATCS)

Dines Bjørner

Domain Science and Engineering

A Foundation for Software Development

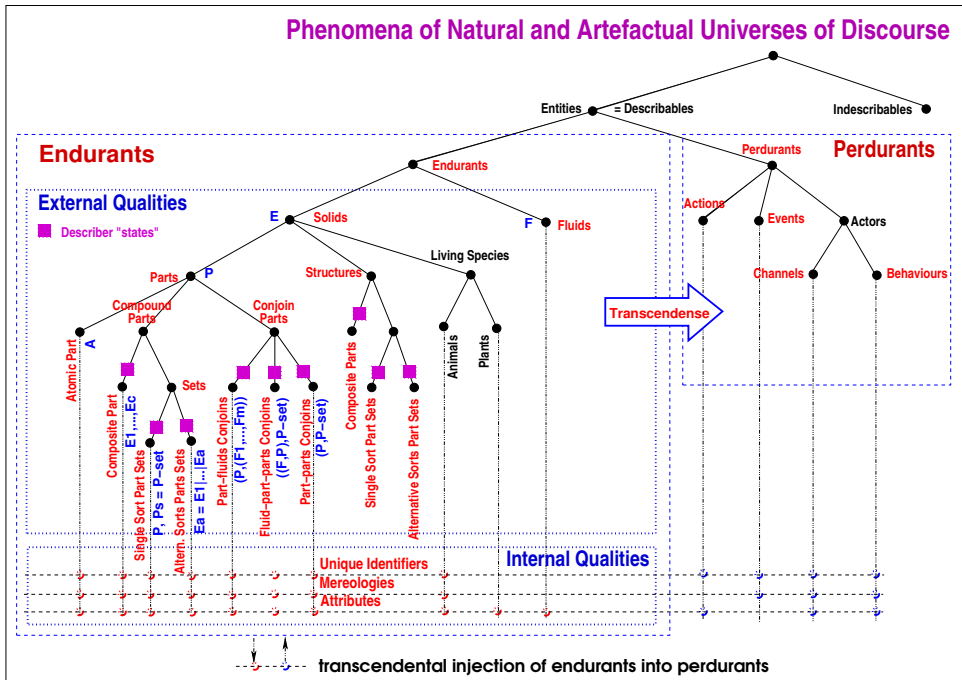
Dines Bjørner
Technical University of Denmark
Holte, Denmark

ISSN 1431-2654 ISSN 2193-2069 (electronic)
Monographs in Theoretical Computer Science. An EATCS Series
ISBN 978-3-030-73483-1 ISBN 978-3-030-73484-8 (eBook)
<https://doi.org/10.1007/978-3-030-73484-8>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2021
This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

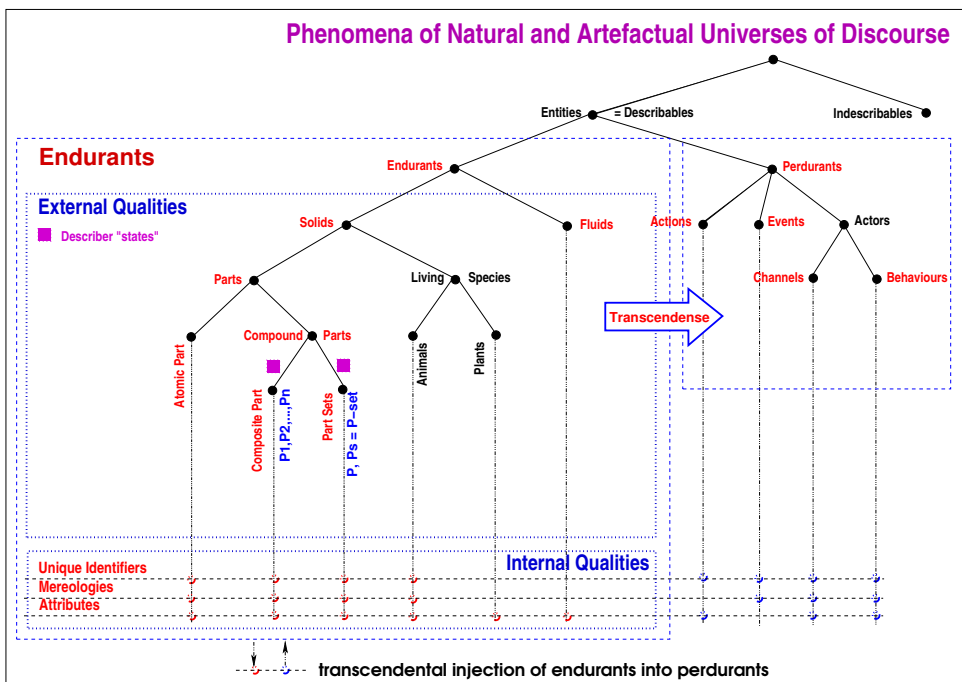
Kari
Charlotte & WeiWei, Nikolaj & Bodil
Camilla, Marianne, Katrine, Caroline and Jakob



Ontology Graph 1: The Ontology for Chapters 4–7s Domain Analysis

...

Ontology Graph 2: A “Minimal” Ontology for Domain Analysis



Preface

The Triptych Dogma

In order to *specify* **software**,
we must understand its requirements.

In order to *prescribe* **requirements**
we must understand the **domain**.

So we must **study**, **analyse** and **describe** domains.

Domains – What Are They ?

By a *domain* we shall understand a *rationally describable* segment of a *discrete dynamics* segment of a *human assisted reality*, i.e., of the world, its *solid or fluid entities*: *natural* [“God-given”] and *artefactual* [“man-made”], and its *living species entities*: *plants* and *animals* including, notably, *humans*. Examples of domains are: *rail, road, sea and air transport; water, oil and gas pipelines; industrial manufacturing; consumer, retail and wholesale markets; health care; et cetera*.

Aim and Objectives

- The **aim** of this monograph is to contribute to a methodology for analysing and describing domains.
- The **objectives** – in the sense of ‘how is the aim achieved’ – is reflected in the structure and contents and the didactic approach of this monograph. The main elements of my approach – along one concept-axis – can be itemized:
 - ∞ There is the founding of our analysis & description approach in providing a base **philosophy**, cf. Chapter 1.
 - ∞ There is the application of ideas of **taxonomy** [see Item 41 on page 8], as in Chapter 4, and **ontology** [see Item 25 on page 6], as in Chapter 5, to understand the possibly hierarchical structuring of domain phenomena respectively the understanding of properties of phenomena and relations between them.
 - ∞ There is the notions **endurants** and **perdurants** – with endurants being the phenomena that can be observed, or conceived and described, as a “complete thing” at no matter which given snapshot of time [281, Vol. I, pg. 656], and perdurants being the phenomena for which only a fragment exists if we look at or touch them at any given snapshot in time [281, Vol. II, pg. 1552].
 - ∞ There is the introduction of base elements of **calculi** for analysing and describing domains.
 - ∞ And finally there is the notion of **transcendental deduction**, cf. Chapter 6, for “morphing” certain kinds of endurants into certain kinds of perdurants, Chapter 7.

Along another conceptual-axis the below are further elements of my approach:

- ∞ We consider domain descriptions, requirements prescriptions and software design specifications to be **mathematical** quantities.
- ∞ And we consider them basically in the sense of **recursive function theory** [351, Hartley Rogers, 1952] and **type theory** [330, Benjamin Pierce, 1997].

- ∞ That is, we do not consider rather appealing diagrammatic description approaches such as for example Petri nets [343, 344, 345, 347, 348, Wolfgang Reisig]¹.

An Emphasis

When we say *domain analysis & description* we mean that the result of such a domain analysis & description is to be a model that describes a usually infinite set of domain instances. Domains exhibit endurants and perdurants. A domain model is therefore something that defines the *nouns* (roughly speaking the endurants) and *verbs* (roughly speaking the verbs) – and their combination – of a *language* spoken in and used in writing by the practitioners of the domain. Not an instantiation of nouns, verbs and their combination, but all possible and sensible instantiations.

Doing Science Versus Using Science

We are not doing mathematics, we use it !

We must distinguish between doing mathematics and using mathematics. This monograph uses mathematics to investigate the universe of domain science & engineering. My books [42, 43, 44] teach their readers how to use simple mathematics to develop software, and, in this monograph, to specifically develop domain models.

I use philosophy to underpin my approach. I do not philosophize. My source, Kai Sørlander does and must. I cannot prevent domain modelers, e.g., those who use the methodology of this monograph, from (“lightweight”) philosophizing, but I do not teach them to do it.

Far too many computing science courses and scientists of computers/computing confuse these two things, **doing science** versus **using science**. It is as if their teachers would rather do mathematics than computing.

Relations to Philosophy

Rather unusual, this monograph contains what the author considers an important chapter, Chapter 2, on **Philosophy**. Not a survey of philosophy topics that might be relevant to the domain science & engineering researchers and scientists, but the promulgation of a rather specific angle under which to pursue domain science & engineering studies. It is very much inspired by the work of the Danish philosopher Kai Sørlander [366, 367, 368, 369, 1994–2016].

General

The claim of this monograph is twofold:

- that domain engineering is a **viable**,
- yes, we would claim, **necessary** initial phase of software development; and
- that domain science &² engineering is a **worthwhile** topic of research.

¹ In a recent technical report, [82], we present, following the method of this monograph, a domain description of an example drawn from a Petri net like, i.e., HERAKLIT, description of a *retailer* system [162].

² We use the ampersand ‘&’ to emphasize that domain science & engineering is one topic, not two.

I mean this rather seriously:

- How can one think of implementing **software**,
- preferably satisfying some **requirements**,
- without demonstrating that one understands the **domain** ?

So in this monograph I shall

- explain what domain engineering is,
- some of the science that goes with it, and
- how one can “derive” requirements prescriptions
 - ∞ (for computing systems)
 - ∞ from domain descriptions.

But **there is an altogether different reason**, also, for presenting these papers in monograph form:

- Software houses may not take up the challenge to develop software
 - ∞ that satisfies customers’ expectations, that is, reflects the domain such as these customers know it,
 - ∞ and software that is correct with respect to requirements, with proofs of correctness often having to refer to the domain.
- But computing scientists are shown, in this monograph, that domain science & engineering is a field full of interesting problems to be researched.

Application Areas

Computers are man-made. They are artefacts. Physicists and engineers compute over domains of **physics** and **engineering designs**, and their computations range mostly over *phenomena of physics*. Manufacturing, logistics and transport firms as well as goods importers/exporters, wholesalers and retail firms use computers significantly. Their domain is mostly **operations research**. With **domain science & engineering** the domain (of possible software applications) is now definable in terms of what the method of this monograph is capable of handling. Briefly, but by far not exhaustively, that domain includes such which focus on man-made objects, i.e., on artefacts, and the interaction of humans with these. In that respect the domain science & engineering, when used for the purposes of software development, straddles the aforementioned application areas but now, we claim, with some firm direction.

Work in Progress

The state of this monograph reflects that it is ‘*work in progress*’. The first publications indicating what is presented here were [56, 60, Summer 2010]. Since then there has been a number of publications in peer-reviewed journals [73, 79, 77, 81, Years 2017–2019]. In the period of submission of the most recent of these [81, Spring 2018], and during the writing of this monograph, up to this very moment this *Preface* is being written, new research discoveries are made. The way that these new research ideas fit well within the framework, also in its detailed aspects, makes me think that the body of work presented here is stable and durable. I have therefore decided to release the monograph now in the hope that it might inspire others to continue the research.

The Monograph as a Textbook

Many universities appear to teach their science students, whether BSc or MSc, only such material for which there exists generally accepted and stable theories. I have over the years, since 1976, when I first joined a university staff — then as a full professor — mostly not adhered to this limitation, but taught, to BSc/MSc students, such material that yet had to reach the maturity of a *scientific theory*. So, go ahead, use this monograph in teaching !

Specific

This monograph is intended for the following mathematics-minded audiences:

- primarily researchers, lecturers and PhD students in the sciences of computers and computing – conventionally speaking: those who have few preconceived objections to the use of discrete mathematics;
- hopefully also their similarly oriented, curious and serious MSc students;
- and finally, recent, and not so recent, practicing software engineers and programmers – again open-minded with respect to new foundations for programming and formalisms.

At the end of most chapters’ ‘*Problem Exercise*’ sections, we suggest a number of anywhere from engineering to science challenges: project-oriented domain analysis & description class-project exercises as well as more individual research problems of more-or-less “standard” degree of difficulty to plain challenging studies. The class-project exercises amount to rather “full-scale” 4–6 student term projects.

Sources

This is a monograph of 11 chapters. Except for three (Chapters 1, 3 and 11), these chapters build on the following publications:

• Chapter 2: Philosophy [80]	11–18
• Chapter 4: External Qualities [81, Sects. 2–3] ³	47–106
• Chapter 5: Internal Qualities [81, Sect. 4]	107–153
• Chapter 6: Transcendental Deduction [81, Sects. 5–6] and [80]	155–157
• Chapter 7: Perdurants [81, Sect. 7]	159–204
• Chapter 8: Domain Facets [78, 55]	205–240
• Chapter 9: Requirements [69, 47]	243–298
• Chapter 10: Demos, Simulators, Monitors and Controllers [61]	301–311

Chapters 1–3 pave the way. They introduce the reader to a **vocabulary of concepts** specific to computing science; to some **fundamental ideas of philosophy** – a new for any treatise of our field; and to **prerequisite concepts of discrete mathematics**, of **space, time and matter**, and of **unique identification** and **mereology** – also new for any treatise of our field.

Chapters 4–7 **form the real core of this monograph**. It is here we develop what we shall, unashamedly, refer to as both a science and an engineering, i.e., a methodology for understanding the concept of ‘*domains*’ such as we shall define it. These chapters study and

³ [73] is a precursor for [81]

develop **calculi** for the **analysis** of domains and for their **description**. At the same time as presenting this study these chapters also **present a method** for actually developing domain descriptions. This duality, the beginnings of a *scientific, theoretical foundation* for domain analysis & describer, and the beginnings of a *method* for actual engineering development, may seem confusing if the twin aspects are not kept clear from one another. We have endeavoured to present the two aspects reasonably separated.

Chapters 8–10 are “bonus” chapters! They contain some quite original concepts: **domain facets** (Chapter 8) such as *intrinsic*s, *support technology*, *rules & regulations*, *scripts*, *license languages*, *management & organisation* and *human behaviour*; **requirements engineering** (Chapter 9) concepts such as the distinction into *domain requirements*, *interfaces requirements* and *machine requirements*, *projection*, *instantiation*, *determination*, *extension* and *fitting*, and more – not quite the way conventional requirements engineering textbooks treat the field; and **demos**, **simulators**, **monitors** and **controllers** (Chapter 10) are all concepts that, we claim, can be interestingly understood in light of *domain descriptions* being developed into *requirements prescriptions* and these into *software designs* and *software*. These chapters may, for better or worse, not be of interest to some computer scientists, but should be of interest to software engineering practitioners and people who do study the more mundane aspects of software engineering.

Some Caveats

This monograph uses the **RAISE Specification Language, RSL** [190, 187] for its formal presentations and for its mixed mathematical notation and RSL informal explanations. We refer to Appendix C for a résumé of RSL. [188, 179, 183] provide short, concise introductions to the RAISE Method and to RSL. We refer to <https://raisetools.github.io/> for Web-based information about RAISE and the RAISE Tool Sets.

Equally relevant other specification languages could be **VDM SL** [92, 93, 164], **Z** [397], the **B Method** notation [2], **Alloy** [262], and others. Also algebraic approaches are possible, for example: **CafeOBJ** [168], **CASL** [135] or **Maude** [298, 133]. Lecturer and students, readers in general, perhaps more familiar with some of the above languages than with **RSL**, should be able to follow our presentations, but perform their exercise/term project work in the language of their choice.

This monograph is the first in which domain science & engineering is presented in a coherent form, ready for scientific study as well as for university classes. But it is far from a polished textbook: Not all “corners” of describable, manifest and artefactual domains are here given “all the necessary” *principles, techniques and [language] tools* necessary for “run-of-the-mill” software development. We have given sufficiently many university courses, over previous texts, and these have shown, we claim, that most students can be expected, under guidance of professionals experienced in formal specifications, to contribute meaningfully to professional domain analysis & description projects.

We have left out of this monograph potential chapters on for example: possible **Semantic Models** of the domain analysis & description calculi [65]. We invite the reader to study this reference as well as to contribute to domain science. Examples of the latter could, for example, entail: **A Study of Analysis & Description Calculi**: *on the order of analysis & description prompts; on the top-down analysis & description, as suitable, for artefactual domains versus bottom-down analyses & descriptions, as perhaps more suitable, for natural and living specific domains, including humans; a deeper understanding of Intentional Pull*, et cetera.

Acknowledgments

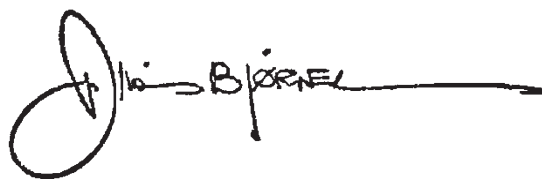
This is most likely the last book that I may be able to publish. Over the years I have co-edited, edited, co-authored or authored a number of published books. Some more noteworthy are: [92, 35, 96, 93, 36, 87, 21, 94, 42, 43, 44, 83, 48, 49, 50, 52, 53, 54].

Over all these years I have benefited in my research from a large number of wonderful people. I bear tribute, in approximate chronological order, to a few of these: (the late) Cai Kinberg, Gunnar Wedell, (the late) Jean Paul Jacob, Peter Johansen, Ivan Havel, Leif Saalbach Andersen, (the late) Gerald Weinberg, (the late) Lotfi Zadeh, (the late) E.F. Codd, (the late) John W. Backus, (the late) Peter Lucas, Cliff B. Jones, (the late) Hans Bekič, Kurt Walk, (the late) Christian Gram, Hans Bruun, (the late) Asger Kjerbye Nielsen, Hans Henrik Løvengreen, Andrzej Blikle, Dömölki Balint, Jozef Gruska, Erich Neuhold, (the late) Douglas T. Ross, Neil D. Jones, Ole N. Oest, (the late) Søren Prehn, Michael A. Jackson, Sir Tony Hoare, Hans Langmaack, (the late) Gilles Kahn, (the late) Kesav V. Nori, Larry E. Druffel, Enn S. Tyugu, Mathai Joseph, Olivier Danvy, Dominique Méry, Jorge R. Cuellar, Zhou Chao Chen, Kokichi Futatsugi, Kazuhiro Ogata, Chris George, Tetsuo Tamai, Klaus Havelund, Jin Song Dong, Arutyun Avetisyan, Marko Schütz-Schmuck, and Krzysztof M. Brzeziński.

I also wish to thank the many colleagues around the world who, in recent years, since my retirement, at age 70, in 2007, have let me try out the ideas of earlier versions of this monograph on their students in MSc/PhD courses: Egon Börger (Pisa), Dominique Méry (Nancy), Wolfgang J. Paul (Saarbrücken), Alan Bundy (Edinburgh), Tetsuo Tamai (Tokyo), Dömölki Balint (Budapest), Andreas Hamfeldt (Uppsala), Luis Barbosa (Braga), Jin Song Dong (Singapore), Jens Knoop (Vienna), Magne Haverlaen (Bergen), Zhu Huibiao (Shanghai) and Chin Wei Ngan (Singapore).

I thank my Springer editor *Ronan Nugent* for his indefatigable, courageous and time-consuming work in getting this monograph published. Ronan's support is deeply appreciated. Thanks Ronan!

I finally wish to thank *Kai Sørlander* for his Philosophy [366, 367, 368, 369]. As you shall find out, Sørlander's Philosophy has inspired me tremendously. Ideas that were previously vague are now, to me, clear. I am sure that You will be likewise enlightened.



Dines Bjørner. August 30, 2021: 10:46 am
Fredsvvej 11, DK-2840 Holte, Denmark

Contents

Preface	ix
The Triptych Dogma	ix
Domains – What Are They ?	ix
Aim and Objectives	ix
An Emphasis	x
Doing Science Versus Using Science	x
Relations to Philosophy	x
General	x
Application Areas	xi
Work in Progress	xi
The Monograph as a Textbook	xii
Specific	xii
Sources	xii
Some Caveats	xiii
Acknowledgements	xiv

Part I SETTING THE SCOPE

1 CONCEPTS	3
1.1 A General Vocabulary	3
1.2 More on Method	8
1.3 Some More Personal Observations	10
1.4 Informatics Thinking	10
2 DOMAIN PHILOSOPHY	11
2.1 Some Preliminaries	11
2.2 Overview of the Sørlander Philosophy	13
2.3 From Philosophy to Physics and Biology	17
2.4 Philosophy, Science and the Arts	18
3 SPACE, TIME and MATTER	19
3.1 Prologue	19
3.2 Logic	19
3.3 Mathematics	20
3.4 Space	26
3.5 Time	29
3.6 Spatial and Temporal Modelling	38
3.7 Matter	39
3.8 Identity and Mereology	39
3.9 A Foundation	41

Part II DOMAINS

Chapter 4–8 Overview	43
A Theory, not The Theory	43
On Learning a Theory and On Learning a Method	44

4	DOMAINS – A Taxonomy: External Qualities	47
4.1	Overview	47
4.2	Domains	47
4.3	Universe of Discourse	48
4.4	External Qualities	51
4.5	Entities	52
4.6	Endurants and Perdurants	56
4.7	Endurants: Solid and Fluid	58
4.8	Parts, Structures and Living Species	60
4.9	Natural Parts and Artefacts	62
4.10	Structures	64
4.11	Living Species – Plants and Animals	65
4.12	Fluids	67
4.13	Atomic, Compound and Conjoin Parts	68
4.14	On Discovering Endurant Sorts	76
4.15	A Review of the Ontology of Endurants	77
4.16	Endurant Observer Function Prompts	79
4.17	Calculating Sort Describers	82
4.18	On Endurant Sorts	90
4.19	States	91
4.20	A Domain Discovery Process, I	93
4.21	Formal Concept Analysis	97
4.22	Summary	98
4.23	Bibliographical Notes	102
4.24	Exercise Problems	103
5	DOMAINS – An Ontology: Internal Qualities	107
5.1	Overview of This Chapter	107
5.2	Unique Identifiers	108
5.3	Mereology	112
5.4	Attributes	119
5.5	Intentionality	139
5.6	Systems Modeling	145
5.7	Discussion of Endurants	146
5.8	A Domain Discovery Process, II	146
5.9	Domain Description Laws	147
5.10	Summary	148
5.11	Intentional Programming	150
5.12	Bibliographical Notes	150
5.13	Exercise Problems	150
6	TRANSCENDENTAL DEDUCTION	155
6.1	Some Definitions	155
6.2	Some Informal Examples	155
6.3	Bibliographical Note	156
6.4	Exercise Problem	157
7	DOMAINS – A Dynamics Ontology: Perdurants	159
7.1	Structure of This Chapter	159
7.2	States and Time	159
7.3	Actors, Actions, Events and Behaviours: A Preview	161
7.4	Modelling Concurrent Behaviours	163
7.5	Channels and Communication	169
7.6	Signatures – In General	172
7.7	Behaviour Signatures and Definitions	176
7.8	System Initialisation	188
7.9	Concurrency: Communication and Synchronisation	189
7.10	Discrete Actions	189
7.11	Discrete Events	192
7.12	A Domain Discovery Process, III	193
7.13	Summary	196
7.14	Exercise Problems	201

8	DOMAIN FACETS	205
8.1	Introduction	205
8.2	Intrinsics	206
8.3	Support Technologies	210
8.4	Rules & Regulations	214
8.5	Scripts	217
8.6	License Languages	220
8.7	Management & Organisation	229
8.8	Human Behaviour	234
8.9	Summary	236
8.10	Bibliographical Notes	239
8.11	Exercise Problems	239

Part III REQUIREMENTS

9	REQUIREMENTS	243
9.1	Introduction	244
9.2	An Example Domain: Transport	245
9.3	Requirements	255
9.4	Domain Requirements	262
9.5	Interface and Derived Requirements	282
9.6	Machine Requirements	291
9.7	Summary	292
9.8	Bibliographical Notes	296
9.9	Exercise Problems	297

Part IV CLOSING

10	DEMOS, SIMULATORS, MONITORS AND CONTROLLERS	301
10.1	Introduction	301
10.2	Interpretations	302
10.3	Summary	310
11	WINDING UP	313
11.1	Programming Languages and Domains	313
11.2	Summary of Chapters 4–7	313
11.3	A Final Summary of Triptych Concepts	314
11.4	Systems Development	316
11.5	On How to Conduct a Domain Analysis & Description Project	316
11.6	On Domain Specific Languages	318
11.7	Some Concluding Observations	318
11.8	A Reflection on Methodologies	318
11.9	Tony Hoare's Reaction to 'Domain Modelling'	319

References	321
-------------------	------------

A	A PIPELINES DOMAIN: ENDURANTS	339
A.1	Solids and Fluids	339
A.2	Unique Identifiers	341
A.3	Mereologies	341
A.4	Attributes	343
B	MEREOLGY, A MODEL	349
B.1	Examples of Illustrating Aspects of Mereology	349
B.2	An Axiom System for Mereology	354
B.3	An Abstract Model of Mereologies	356
B.4	Some Part Relations	361
B.5	Satisfaction	363

C	FOUR LANGUAGES	365
C.1	The Domain Analysis & Description Calculi	365
C.2	The Language of Explaining Domain Analysis & Description	367
C.3	The RSL: Raise Specification Language	367
C.4	The Language of Domains	368
D	AN RSL PRIMER	369
D.1	Types	369
D.2	The RSL Predicate Calculus	372
D.3	Concrete RSL Types: Values and Operations	372
D.4	λ -Calculus + Functions	380
D.5	Other Applicative Expressions	382
D.6	Imperative Constructs	384
D.7	Process Constructs	385
D.8	Simple RSL Specifications	386
D.9	RSL Module Specifications	386
E	INDEXES	389
E.1	Definitions	389
E.2	Concepts	392
E.3	Examples	395
E.4	Method Hints	396
E.5	Analysis Predicate Prompts	397
E.6	Analysis Function Prompts	397
E.7	Attribute Categories	397
E.8	Perdurant Calculations	398
E.9	Description Prompts	398
E.10	Endurant to Perdurant Translation Schemas	398
E.11	RSL Symbols	398
	List of Figures	401