

Strategies for Real-Time System Specification

*by Derek J. Hatley
and Imtiaz A. Pirbhai*

Foreword by Tom DeMarco 

FREE SAMPLE CHAPTER

SHARE WITH OTHERS



**STRATEGIES FOR
REAL-TIME SYSTEM
SPECIFICATION**

This page intentionally left blank

STRATEGIES FOR REAL-TIME SYSTEM SPECIFICATION

By

Derek J. Hatley Imtiaz A. Pirbhai

**Dorset House Publishing
353 West 12th Street
New York, New York 10014**

Library of Congress Cataloging-in-Publication Data

Hatley, Derek J., 1934-
Strategies for real-time system specification.

Bibliography: p.

Includes index.

1. Real-time data processing. 2. System design.

I. Pirbhai, Imtiaz A., 1953- . II. Title.

QA76.54.H38 1987 004'.33 87-50801

ISBN 0-932633-04-8

CREDITS

Cover Design: Jeff Faville, Faville Graphics

Back Cover: Hatley photo: Bultman Studios, Inc.; Pirbhai photo: Stewart Auer.

The text of the book was set by the authors using the LaTeX document preparation system in cooperation with Smiths Industries. The graphics were prepared by the authors using Macintosh Plus and MacDraft.

Copyright © 1987 by Derek J. Hatley and Imtiaz A. Pirbhai. Published by Dorset House Publishing Co., Inc., 353 West 12th Street, New York, NY 10014.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Distributed in the United Kingdom, Ireland, Europe, and Africa by John Wiley & Sons Ltd., Chichester, Sussex, England.

Printed in the United States of America

Library of Congress Catalog Card Number 87-50801

ISBN: 0-932633-04-8

To Sue

DJH

To my father,
Baba Anwar Shah Taji,
and in memory of my mother

IAP

Acknowledgments

First and foremost, we thank Wendy Eakin, our editor and publisher, who gently but firmly led these two neophyte authors through the rigors of writing a book. You too should thank her, dear Reader: For most of our careers we have worked on or with engineering specifications and standards, and if you are at all familiar with those kinds of documents, you will understand that she is all that stands between you and total boredom.

Many people have contributed in many ways to the work presented here, and while we can mention only a few, we thank you all. Bruce Chubb, Angelo Dimitriou, Rex Morin, and Rod Wierenga gave management support for development and introduction of the requirements method. Dave Bulman sat in on the method's original conception, and, with his great knowledge and experience, helped us sift out the good ideas from the bad. The original Methods Team consisted of Kathy Hornbach, Bill Kelly, Martie Lorenz, John McCreary, and George Wood. Don Morrow remains the preeminent practitioner of the method itself, while John TenCate is master of the art of transforming the resulting model into a real system. Linda Goggins succeeded in the enormous task of coordinating and compiling the first (manual) application of the method. Kent McPherson performed some miracles with L^AT_EX, the document preparation system for this book, to make it work the way we wanted.

Tim Petersen, Irv Reese, Dick Schoenmann, and Peter Sutcliffe provided the management support needed for the development of the architecture method, as well as support for the further expansion and acceptance of both the requirements and architecture methods in the user community.

Dan Bacon, Rodney Bell, James Bouhana, Bob Doering, Peter Hruschka, Lou Mazzucchelli, Don Morrow, Rick Swanborg, Tony Wasserman, and Bob Weisickle reviewed various drafts of the book and improved it immeasurably with their insights.

Finally, we give a very special acknowledgment to Paul Gartz. Paul has been an enthusiastic champion of these methods, and an inspiration to the rest of us working on them. His aggressive canvassing on behalf of the methods has been responsible for their acceptance by management and the industry.

Contents

List of Figures	xv
Foreword	xxiii
Preface	xxv
PART I: The Overall Strategy	1
1 Overview	3
1.1 The Birth of the Requirements Model	4
1.2 The Birth of the Architecture Model	7
1.3 Compatibility of the Models	7
1.4 Applicability of the Models	8
1.5 The System Life Cycle	8
2 The Role of the Methods	11
2.1 Structured Methods: What They Are	11
2.2 System Requirements Model	13
2.3 System Architecture Model	19
2.4 System Specification Model	25
2.5 The Development Life Cycle	27
2.6 Structured Methods: What They Are Not	31
2.7 Summary	32
PART II: The Requirements Model	33
3 Overview	35
3.1 The Structure of the Model	37

4	The Process Model	41
4.1	Data Context Diagrams	41
4.2	Data Flow Diagrams	44
4.3	Leveling and Balancing	47
4.4	The Numbering System	49
4.5	Data Flows	49
4.6	Data Stores	52
4.7	Process Specifications	53
4.8	Interpreting the Process Model	56
4.9	Summary	59
5	The Control Model	61
5.1	Control Context Diagrams	61
5.2	Control Flow Diagrams	64
5.3	Control Flows	67
5.4	Data Conditions	69
5.5	Control Stores	70
5.6	Control Specifications	70
5.7	Process Controls	73
5.8	Summary	76
6	Finite State Machines	77
6.1	Combinational Machines	79
6.2	Sequential Machines	83
6.3	Incorporating Finite State Machines into CSPECs	89
6.4	Summary	97
7	Timing Requirements	98
7.1	Repetition Rate	99
7.2	Input-to-Output Response Time	99
7.3	Summary	102
8	Requirements Dictionary	103
8.1	Primitive Attributes	104
8.2	Group Structure	105
8.3	Dictionary Data Bases	107
8.4	Summary	110
9	Requirements Model Interpretation and Summary	111
9.1	The Requirements Model Interpreted	111
9.2	Requirements Model Summary	113

PART III: Building the Requirements Model	117
10 Overview	119
10.1 Model Users and Builders	119
10.2 The Sources of Requirements	120
10.3 The Model Building Process	121
11 Getting Started	124
11.1 User Requirements Statements	124
11.2 Separating Data and Control	125
11.3 Establishing the System Context	129
11.4 Partitioning the Top Levels	132
11.5 Summary	135
12 Developing the Model's Structure	137
12.1 Abstraction and Decomposition	137
12.2 The Seven-Plus-or-Minus-Two Principle	138
12.3 Grouping and Decomposing Processes	139
12.4 Grouping and Decomposing Flows	140
12.5 Naming Processes and Flows	147
12.6 Use of Stores	149
12.7 Functionally Identical Processes	150
12.8 De-emphasizing the Control Model	151
12.9 Control Intensive Systems	153
12.10 The Dilemma of Detail: Requirements Versus Design	155
12.11 The Final Product	156
12.12 Summary	156
13 Preparing Process Specifications	158
13.1 The Role of Process Specifications	158
13.2 The Different Types of PSPECs	159
13.3 Some Important Signal Conventions	162
13.4 Structured English	165
13.5 Annotating with Comments	167
13.6 Summary	167
14 Preparing Control Specifications	169
14.1 Avoiding Control Specifications	169
14.2 Combinational Control	170
14.3 Sequential Control	176
14.4 Multi-Sheet CSPECs	182

14.5	Fitting CSPECs In	185
14.6	Summary	189
15	Defining Timing	190
15.1	Timing Overview	190
15.2	Response Time Specification	192
15.3	Summary	193
16	Managing the Dictionary	194
16.1	Flow Types	194
16.2	Dictionary Symbols	198
16.3	Summary	199
PART IV: The Architecture Model		201
17	Overview	203
17.1	Requirements-to-Architecture Template	204
17.2	Architecture Model Symbols	207
18	Architecture Diagrams	211
18.1	Architecture Context Diagrams	211
18.2	Flows and Interconnects	213
18.3	Architecture Flow Diagrams	213
18.4	Architecture Interconnect Diagrams	218
18.5	Summary	224
19	Architecture Dictionary and Module Specifications	225
19.1	Architecture Module Specification	226
19.2	Architecture Interconnect Specification	229
19.3	Timing Requirements	232
19.4	Architecture Dictionary	233
19.5	Summary	234
20	Completing the Architecture Model	235
20.1	Allocation to Hardware and Software	235
20.2	The Hardware and Software Architectures	236
20.3	The Complete Architecture Model	238

PART V: Building the Architecture Model	241
21 Overview	243
21.1 Architecture Development Process	244
21.2 Systems Come in Hierarchies	246
22 Enhancing the Requirements Model	248
22.1 Input and Output Processing	249
22.2 User Interface Processing	251
22.3 Maintenance and Self-Test Processing	254
22.4 The Complete Enhanced Requirements Model	256
22.5 Technology-Independent Versus Technology-Nonspecific	257
22.6 Organizational Implications	260
22.7 Summary	261
23 Creating the System Architecture Model	263
23.1 Architecture Context Diagram	263
23.2 Architecture Flow and Interconnect Diagrams	264
23.3 Example of AFD and AID Mapping	266
23.4 Model Consistency and Balancing	268
23.5 The Complete Architecture Model	271
23.6 Summary	272
24 Creating the Hardware and Software Architecture Models	273
24.1 Hardware and Software Partitioning	276
24.2 Applying the Template to Software Requirements	279
24.3 Developing the Software Architecture	282
24.4 The Hardware and Software Architecture Process	284
24.5 Summary	285
25 Architecture Development Summary	286
25.1 Partitioning the Modeling Process	286
PART VI: Examples	293
26 Automobile Management System	295
26.1 Problem Statement	295
26.2 Requirements and Architecture Development	297
26.3 Requirements Model	299
26.4 Architecture Model	319

27 Home Heating System	326
27.1 Problem Statement	326
27.2 Requirements Model	330
27.3 Architecture Model	340
28 Vending Machine	342
28.1 Customer Dialogue	342
28.2 Requirements Model	344
28.3 Architecture Model	355
APPENDICES	363
APPENDIX A: Standard Symbols and Definitions	363
A.1 Introduction	363
A.2 Standard Symbols	363
A.2.1 Data flow	363
A.2.2 Control flow	364
A.2.3 CSPEC bar	364
A.2.4 Process	365
A.2.5 Store	365
A.2.6 Terminator (source or sink, also external)	365
A.2.7 Architecture module	366
A.2.8 Information flow vector	366
A.2.9 Information flow channel	367
A.3 Requirements Model	367
A.3.1 Context diagrams	368
A.3.2 Flow diagrams	370
A.3.3 Process and control specifications	372
A.3.4 Timing specifications	376
A.3.5 Requirements dictionary	377
A.3.6 Requirements model balancing rules	379
A.4 Architecture Model	382
A.4.1 Architecture context diagram	382
A.4.2 Architecture flow and interconnect diagrams	384
A.4.3 Architecture module and interconnect specifications	386
A.4.4 Architecture dictionary	388
A.4.5 Balancing rules	390

APPENDIX B: Making the Models into Documents	392
B.1 Organizing the Models	392
B.2 Military Standards	396
APPENDIX C: Information Modeling: The Third Perspective	398
References	403
Index	405

This page intentionally left blank

List of Figures

A Map of the Book	xxvi
1.1 The universal hierarchy of systems	4
1.2 The total system life cycle	9
2.1 Vending machine DFD	14
2.2 Vending machine PSPEC	14
2.3 Composite data entries from vending machine dictionary . .	15
2.4 Vending machine CFD	16
2.5 Vending machine CSPEC	17
2.6 Primitive control entries from vending machine dictionary . .	18
2.7 The structure of the requirements model	18
2.8 Vending machine AFD	21
2.9 Vending machine AMS	21
2.10 Vending machine architecture dictionary entries	22
2.11 Vending machine AID	22
2.12 Vending machine AIS	23
2.13 Augmented vending machine architecture dictionary	23
2.14 The structure of the architecture model	24
2.15 The formation of the system specification	25
2.16 Requirements-to-architecture iterative loop	26
2.17 System, software, and hardware specification structure	28
2.18 Overlapping systems development phases	28
2.19 Development life cycle spiral	30
3.1 Composite chart of the requirements model	38
4.1 The process structure (highlighted)	42
4.2 Data context diagram	43
4.3 Data flow diagram	45
4.4 Transaction centers	46
4.5 Four levels of data flow diagrams superimposed	47

4.6	Leveling, numbering, and parent/child relationships	48
4.7	Data flows	51
4.8	Splitting flows between levels	52
4.9	Two examples of process specifications	54
4.10	The structures of structured English	57
4.11	The primitive network	58
5.1	The control structure (highlighted)	62
5.2	Control context diagram	63
5.3	Control flow diagram	64
5.4	Variations on CSPEC bars	66
5.5	Comparison between discrete and continuous signals	68
5.6	A data condition	69
5.7	The model as a feedback control loop	72
5.8	Process controls	74
5.9	Controlling a transaction center	75
6.1	Decision table for a heating system	81
6.2	Condensed decision tables for a heating system	82
6.3	Rotary combination lock: a sequential machine	84
6.4	State transition diagram	85
6.5	State transition table	87
6.6	State transition matrix	88
6.7	State transition matrix (alternate form)	89
6.8	Combinational CSPEC, with its DFD and CFD	91
6.9	Sequential CSPEC, with its DFD and CFD	92
6.10	Block diagram of composite CSPEC	93
6.11	Composite CSPEC	94
6.12	Typical multi-sheet control specification	96
7.1	Response time specifications	100
7.2	Timing diagram to supplement the response time specification	101
8.1	Typical primitive continuous signal definitions	104
8.2	Typical primitive discrete signal definitions	104
8.3	Requirements dictionary symbols	106
8.4	Dictionary listing	108
8.5	Indented explosion of a large group flow	109
9.1	Primitive network with control structure	112

11.1	Signal and process categories	127
11.2	Typical PSPEC: a primitive requirements statement	129
11.3	Assuming the worst: all conceivable functions included	130
11.4	Functions reduced after checking system scope	131
11.5	Diagram with a first-cut control flow	132
11.6	Composite top-level data and control flow diagram	133
11.7	DFD 0: Control & Monitor Auto	134
11.8	CFD 0: Control & Monitor Auto	135
12.1	Correction of uneven flow distribution	141
12.2	Correction of clustered processes	142
12.3	Splitting and merging data flows	143
12.4	Decomposing data flows	144
12.5	Splitting off unused flows	144
12.6	Flows with common members	145
12.7	Multiple flows	146
12.8	Two-way flows	146
12.9	The half-arrow convention	147
12.10	A flow tree	147
12.11	Functionally identical processes	151
12.12	Composite chart of the model for control intensive system	154
13.1	A PSPEC with equations	160
13.2	A tabular PSPEC	161
13.3	A flight profile used in a PSPEC	163
13.4	Geometrical relationships used in a PSPEC	164
13.5	Continuous, discrete-valued, and time-discrete signals	165
14.1	A full decision table	172
14.2	A reduced decision table	173
14.3	An ambiguous set of input combinations	174
14.4	Alternative representation of transaction center control	176
14.5	Data flow diagram showing throttle control	180
14.6	Control specification showing throttle control	182
14.7	A multi-sheet state transition matrix	183
14.8	Organization of a composite CSPEC	184
14.9	A floating CSPEC	187
15.1	Dictionary, response time, PSPEC, and CSPEC timing specifications	191

16.1	Flow types and their attributes	195
17.1	Architecture model components	204
17.2	Requirements template	205
17.3	Architecture template	205
17.4	Architecture template for the Automobile Management System	206
17.5	Architecture model layering	207
17.6	Architecture module symbol	208
17.7	Information flow vector symbol	209
17.8	Information flow channel symbols	209
18.1	ACD for a flight management system	212
18.2	ACD for the Automobile Management System	213
18.3	AFD for a flight management system	215
18.4	AFD for a data acquisition subsystem	216
18.5	Architecture template for a flight management system	217
18.6	Architecture template for a data acquisition subsystem	217
18.7	AFD 0: Automobile Management System	218
18.8	AFD layering	219
18.9	AFD 2: Shaft Interface Module	220
18.10	AFD 5: Throttle Interface Module	220
18.11	AID for a flight management system	221
18.12	Redundancy of architecture modules	221
18.13	Redundancy of architecture channels	222
18.14	AID 0: Automobile Management System	223
18.15	AID 2: Shaft Interface Module	223
19.1	Architecture model support components	225
19.2	Architecture module specification	227
19.3	A DFD, CFD, and their CSPEC to be allocated	228
19.4	Splitting CSPECs for allocation	228
19.5	Traceability matrix for the Automobile Management System	230
19.6	Interconnect channel and corresponding AIS	231
19.7	A typical AIS	231
19.8	Partial architecture dictionary for the Automobile Management System	233
20.1	Partitioning between hardware and software	236
20.2	System, hardware, and software requirements hierarchy	237
20.3	System, hardware, and software architecture hierarchy	237

20.4	A vertical slice through the architecture model	239
21.1	Hierarchical nature of natural systems	246
21.2	Hierarchical nature of systems development	247
22.1	Outside-in approach to architecture development	249
22.2	Input and output buffers for the requirements model	250
22.3	Input and output processes for the Automobile Management System	251
22.4	Enhancements to the Automobile Management System DFD for user interface processing	252
22.5	DFD/CFD for the driver interface process	253
22.6	User interface buffer to the requirements model	254
22.7	Enhancements for maintenance and self-test buffer	255
22.8	Maintenance processing added to the requirements model	256
22.9	Enhanced DFD for Automobile Management System	257
22.10	Enhanced CFD for Automobile Management System	258
22.11	Architectural enhancements to the requirements model	259
23.1	ACD for the Automobile Management System	264
23.2	Generic DFD/CFD/CSPEC	265
23.3	AFD 0: Automobile Management System	266
23.4	AID 0: Automobile Management System	267
23.5	Architecture module specification for the Automobile Management System	268
23.6	Architecture interconnect specification for the Automobile Management System	269
23.7	Architecture dictionary for the Automobile Management System	269
23.8	Architecture development process	272
24.1	Hierarchical nature of the Automobile Management System	273
24.2	DFD of requirements allocated to the automobile management computer module	274
24.3	CFD of requirements allocated to the automobile management computer module	275
24.4	AID for automobile management computer hardware configuration	277
24.5	Requirements allocated to hardware	277
24.6	Enhanced hardware requirements	278
24.7	AFD for shaft interface module	278

24.8	AID for shaft bus receiver	278
24.9	Enhanced DFD for the automobile management computer software	280
24.10	Enhanced CFD for the automobile management computer software	281
24.11	A structure chart	282
24.12	Software development structure	283
24.13	Hardware and software architecture development process	284
25.1	Iterative process for developing system requirements and architecture	287
25.2	System modeling process	288
25.3	Hardware and software partitioning process	289
25.4	Hardware and software modeling process	289
25.5	Application of requirements and architecture modeling to large systems	290
25.6	Integrated system, hardware, and software model	291
A.1	Data flow: A solid arc with a name	364
A.2	Control flow: A dashed arc with a name	364
A.3	CSPEC bar: A short unlabeled bar	364
A.4	Process: A circle with a name and a number	365
A.5	Store: A pair of parallel lines containing a name	365
A.6	Terminator: A rectangle containing a name	366
A.7	Architecture module: A rounded rectangle containing a name and number	366
A.8	Information flow vector: A solid or dashed line with a name	367
A.9	Architecture information flow channels	367
A.10	Components of the requirements model	368
A.11	Sample data context diagram	369
A.12	Sample control context diagram	370
A.13	Sample data flow diagram	371
A.14	Data flow diagram naming and numbering rules	372
A.15	Sample control flow diagram	373
A.16	DFD and CFD naming and numbering	373
A.17	Samples of process specifications	374
A.18	Process specification naming and numbering rules	375
A.19	Generic CSPEC for combinational system	376
A.20	Sample CSPEC for combinational system	377
A.21	Generic CSPEC for sequential system	378

A.22	Sample CSPEC for sequential system	379
A.23	Illustration of flow decomposition in the requirements dictionary	381
A.24	Components of the architecture model	382
A.25	Architecture template	383
A.26	Sample architecture context diagram	383
A.27	Sample architecture flow diagram	385
A.28	AFD naming and numbering rules	386
A.29	Sample architecture interconnect diagram	387
A.30	Sample architecture module specification	388
A.31	Sample architecture interconnect specification	389
A.32	Example of enhancements to dictionary	389
C.1	Requirements projections of a system	399
C.2	Architecture template with three requirements projections . .	400

This page intentionally left blank

Foreword

“Most systems people use the term *real-time* rather loosely,” the young manager said. We were seated over dinner with three members of her staff and some other managers who took part in the day’s seminar. “They say they’ve got a real-time constraint when they’re worried about impatient insurance brokers or bankers sitting in front of their terminals. A real-time system, in their minds, is just one that needs to be ‘quick as a bunny.’ If they fail to meet that constraint, their users might be inconvenienced or even annoyed. When we use the term, it means something rather different.”

Her co-workers began to smile, knowing what was coming. “We build systems that reside in a small telemetry computer, equipped with all kinds of sensors to measure electromagnetic fields and changes in temperature, sound, and physical disturbance. We analyze these signals and transmit the results back to a remote computer over a wide-band channel. Our computer is at one end of a one-meter long bar and at the other end is a nuclear device. We drop them together down a big hole in the ground and when the device detonates, our computer collects data on the leading edge of the blast. The first two-and-a-quarter milliseconds after detonation are the most interesting. Of course, long before millisecond three, things have gone down hill badly for our little computer. We think of *that* as a real-time constraint.”

I had been lecturing that day on the use of data flow modeling techniques for system specification. There had been the odd question about the use of such techniques for real-time systems, and my (typically glib) answer was that the techniques were applicable, though perhaps not totally sufficient. After all, my own earliest work with data flow modeling had been at Bell Labs and at La CEGOS Informatique, in both cases working on real-time systems. Or were those real-time systems? Maybe they were really just systems that needed to be ‘quick as a bunny’? The young manager’s graphic example had left me in some doubt.

It has always been clear that something more than the basic tools of structured analysis are needed to specify timing and synchronization requirements.

In the simplest case, that “something” could be as trivial as a set of textual annotations, perhaps directly on the data flow diagrams. But for even slightly more complicated cases, there is the possibility of linked timing and ordering constraints that may involve a dozen or more system components. There has been a need for a systematic way to deal with such constraints.

Over the last few years, I began to hear favorable reports of some real-time modeling techniques called the Hatley/Pirbhai Extensions. This multi-perspective approach combined data flow decomposition with model components constructed in control- and information-space. The result appealed to me as a specification technique that was not only applicable, but for most cases sufficient for real-time systems. I contacted the developers and started to try out their extensions.

The act of writing a Foreword is a kind of endorsement. It says, if nothing else, “Read this book, it couldn’t hurt.” In this case, I can be considerably more positive than that. I can tell you that I learned valuable new techniques from Hatley and Pirbhai, and that I apply them regularly on real-world real-time applications.

October 1987
Camden, Maine

Tom DeMarco
The Atlantic Systems Guild

Preface

This book describes two methods for specifying, respectively, the requirements for and the design structure of software-based systems. Although the methods grew up around real-time embedded systems, systems of all types and sizes have benefited from them, largely because of their flexibility and adaptability. The methods can be viewed as an integrated toolkit from which all the tools are compatible, but only those that are useful for the particular job need be used.

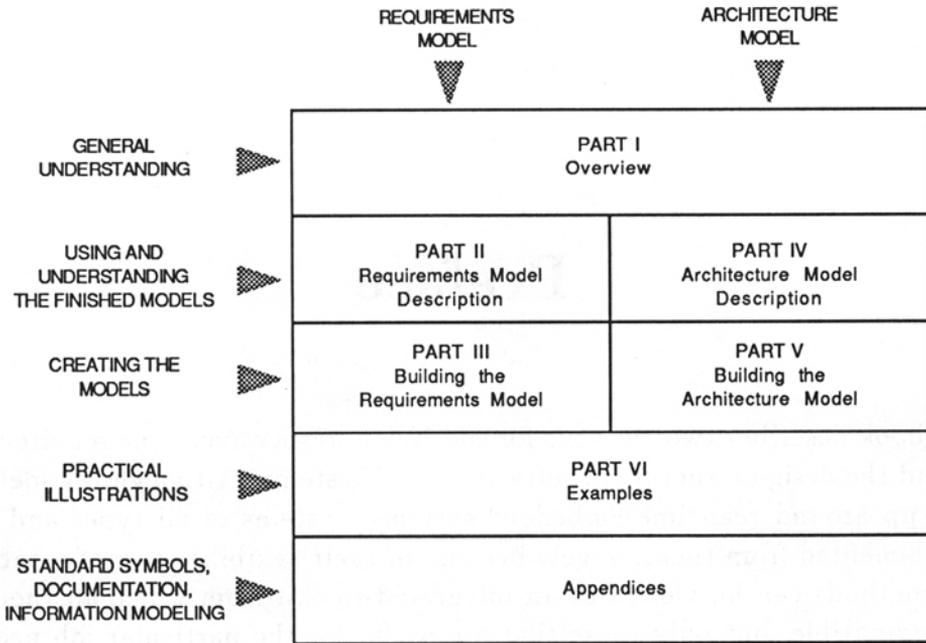
The subject of this book is neither computer science nor software. Although the vast majority of the systems to which the methods are applied will in fact be implemented using software in digital computers, the methods do not address how to write that software or how to design that hardware. What the methods do address is how to specify the *problems* that the hardware and software must solve.

Organization and audience of the book

The book is intended for a wide range of readers. We assume that you are involved, or at least interested, in software-based systems, but that your needs may range from just a general understanding to an in-depth working knowledge. We have tried to organize the book to make it easy for you to select those parts that are of specific interest to you and to avoid those that are not.

The figure on the following page shows the layout of the book. Part I provides an overview from which everyone will benefit. Parts II and IV describe *what* the methods are, and if you only need to understand the specifications resulting from their use, then these two parts are enough for you to read. If you need to *prepare* specifications, then you will also need to read Parts III and V, which describe *how* to use the methods. Part VI contains examples of the application of the methods, and we conclude the book with appendices and a brief bibliography.

Decide what your area of interest is—requirements analysis, design, or both—and what depth of understanding you need—general familiarity, use



A Map of the Book

of the completed specifications, or building the specifications—and it will be clear from the figure which parts of the book you should read.

Derek J. Hatley
Wyoming, Michigan
Imtiaz A. Pirbhai
Seattle, Washington

**STRATEGIES FOR
REAL-TIME SYSTEM
SPECIFICATION**

This page intentionally left blank

Chapter 3

Overview

Our strategy is to establish what the desired end product is, before describing how to produce it. Consequently, in Part II, we describe in detail the completed requirements model, as it would be received by the user, and in Part III, we describe *how* to construct the model. The principal tools of the requirements model are flow diagrams: graphical models of signal flows and processes acting on those flows. Flow diagrams consist of data flow diagrams and control flow diagrams. The former are essentially the same as the DFDs of DeMarco [4]; the latter are similar to DFDs but with some important and unique differences.

The other components of the requirements model are process specifications and the requirements dictionary (which are similar to DeMarco's mini-specifications and data dictionary), and control specifications (which are unique to this method).

You might wonder why we are describing here those parts of the method that have been described so often, and so well, before. The answer is, first, that we want the book to stand alone as a complete description of the whole method. Second, the earlier descriptions allowed a commendable degree of flexibility in the ways the method could be used. This has resulted in many different conventions being adopted in its use. It is therefore important to explain the particular conventions *we* have adopted in extending the method. If you have already been using "basic" structured analysis, you might very well find that some of our conventions are quite different from yours.

Key features of the method are

- It is a purely abstract model of the requirements, not a physical model of the design. It thus represents *what* has to be designed, not *how* to design it.

- It is diagrammatic, exploiting our ability to absorb visual information more effectively than textual or verbal information (“a picture is worth a thousand words”).
- It hides unnecessary detail at any given level, and provides for progressive decomposition from that level down—the information hiding, abstraction, and decomposition principles put forward by Parnas [13]—thus allowing presentation of the requirements from a top-level overview down to the most intricate detail.
- It meets the needs of large, complex systems, including real-time systems, for representation of a process structure and a control structure. It does this by integrating DeMarco’s structured analysis with finite state machine theory. Neither of the methods alone can meet these needs.
- It has built-in consistency checks within and between all its components. This feature of structured analysis has been extended to the whole structure, taking the guesswork out of the rigor and consistency of the model.
- It is self-indexing, another feature of structured analysis that has been successfully extended. It is easy to find your way around the resulting model without referring to page or paragraph numbers (although formal documentation requirements usually force us to keep these numbers).

The first of these features—the abstract nature of the model—deserves special emphasis, because it seems to be the one that newcomers to the method find hardest to grasp. Perhaps the best way to approach it is to understand the motivation for wanting such a feature, and this is best understood by reviewing the history of structured methods development.

Although structured methods generally represent their subject matter in a top-down fashion, the methods themselves have evolved in a bottom-up fashion. As software-based systems grew larger and more complex, the first obvious problem to be overcome was unmanageable and incomprehensible code. The need to solve this problem gave rise to structured programming in the form of constraints on module size, entry and exit points, branching, and so on. It also gave rise to high order languages, and, more recently, to program design languages.

Structured programming improved the quality of the code within individual modules, but still, as systems continued to grow in size and complexity, the interactions *between* the modules became as complex as the earlier code had been. Structured design methods, in which modules are constrained to communicate with each other in an organized way, were developed to deal

with this complexity. These methods improved the quality of the software structure external to the modules, such that if both structured design and structured programming methods were conscientiously applied, the resulting system would work exactly as the designer intended.

Unfortunately, it still often happens that what the designer intended is not what the user wanted! The reason for this is that as the technology allows us to make systems that have ever larger capacity and complexity, the requirements themselves for such systems become large, complex, and difficult to describe and understand. Thus, the need was established for a method to represent the functional requirements of a system in an organized and understandable way, and in a way that is entirely independent of the eventual design and implementation of the system. Given such a model, various designs can be developed and compared without changing the requirements representation.

DeMarco's structured analysis method was designed to meet this need for data processing systems, and the extended structured analysis method described here was designed to meet the need for more general systems.

The model does, unfortunately, have a machine-like appearance; moreover, we are used to models that *are* intended to represent the actual design structure or implementation. As should now be clear, this is *not* the intent here. Again, it is a purely abstract model, with highly idealized properties, whereby each of its processes is independently data triggered and infinitely fast—unlikely characteristics for a real system. These properties are deliberately chosen for the very reason that we *can* ignore the constraints of the physical world, and concentrate exclusively on the required functionality.

As we build a requirements model, we follow this separation principle by including those characteristics that are of concern to the users, and deliberately excluding those that are not. The latter are left to the designer, and the moment we allow them to migrate into the requirements model, then we are including items that are *not* requirements; in so doing, we are placing unnecessary and artificial constraints on the designer.

We will remind you of this property at appropriate places throughout Parts II and III to emphasize its importance in each aspect of the model.

3.1 The Structure of the Model

Figure 3.1 is a composite chart, which enlarges on the diagram of Figure 2.7, and illustrates how the major components of the requirements model relate to each other. We will describe the major roles and relationships of these components before going into their individual details in the following chapters.

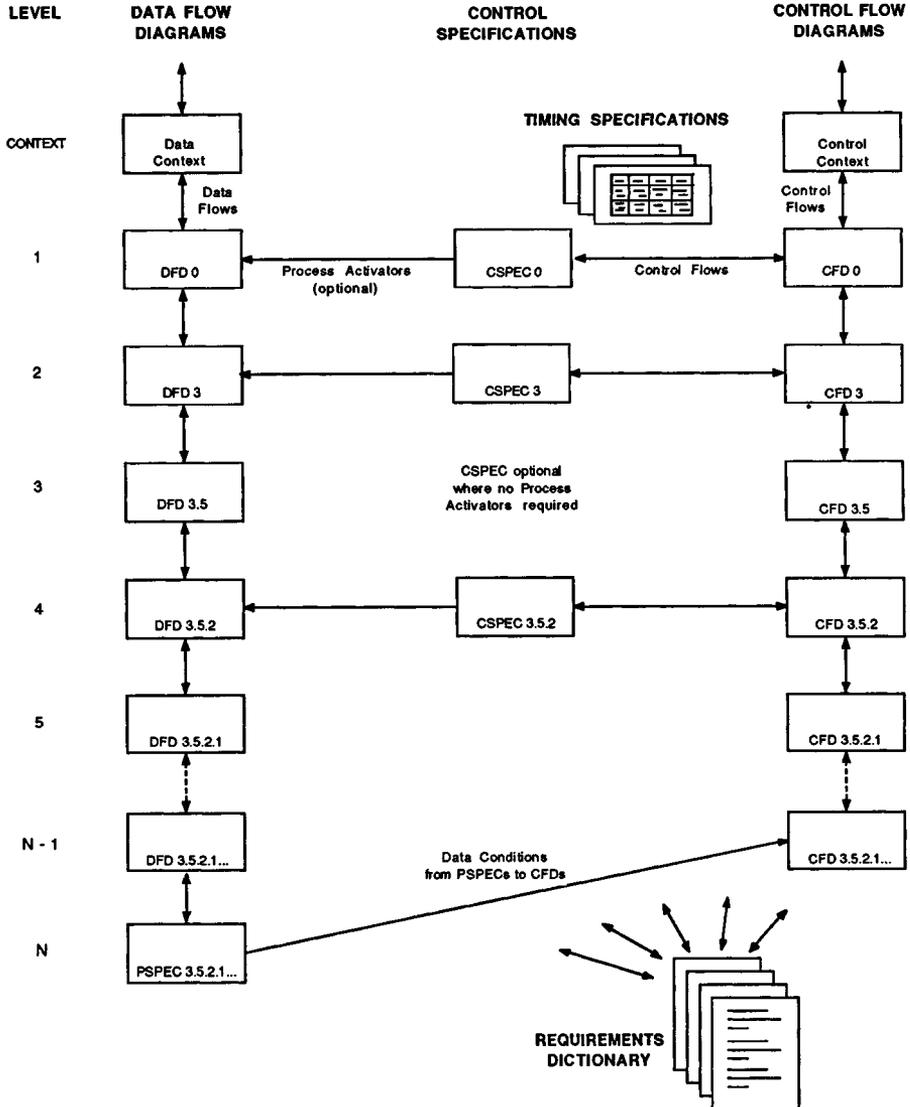


Figure 3.1. Composite chart of the requirements model.

At the top of Figure 3.1 are the data context diagram and the control context diagram. These show, respectively, the data and the control signals flowing between the system and the external entities with which the system must communicate. They also show the overall function of the system in the form of a single process. These diagrams represent the highest-level, or most abstract, view of the requirements, and show the central role the system is required to perform within its external environment.

Next (below the context diagrams in Figure 3.1) are two more flow diagrams—the level 1 DFD and CFD. They too represent the overall function of the system in graphical form, but in somewhat more detail than the context diagrams.

Below the level 1 diagrams are level 2 diagrams. For purposes of illustration, the figure shows just one DFD and one CFD at level 2, but, in fact, each of the several subprocesses on the level 1 diagrams may have its own level 2 DFD and CFD, so there will be several diagrams at this level. Just as the level 1 diagrams represent a more detailed statement of the system's purpose, so the level 2 diagrams represent more detailed statements of the purposes of the subprocesses on the level 1 diagram.

Similarly, level 3 diagrams decompose the processes and signal flows on the level 2 diagrams into more detailed statements, with a further multiplication of the number of diagrams. This decomposition continues level by level, each step revealing finer and finer details of the originally stated system's purpose.

The decomposition ends, for any given subprocess on the DFD side, with a concise statement of the purpose of a process using simple text, equations, tables, or diagrams. This statement constitutes a process specification, as illustrated at the bottom of the column of DFDs. A process that is described by a process specification is called a functional primitive, or just a primitive.

On the CFD side, the decomposition parallels that of the DFDs, but some of the control signals flow to and from control specifications, shown in the center column of Figure 3.1. The CSPECs describe the finite state machine functions of the system, which we will discuss in Chapter 6. Each CSPEC is associated with a specific CFD—the one adjacent to it in the diagram. It will usually control processes, but only those in the specific DFD adjacent to it in the figure. This close association of a DFD, CFD, and CSPEC is an important part of the structure and self-consistency properties of the requirements model.

The system input-to-output timing requirements are shown in the timing specification, shown between the context and level 1 diagrams (and detailed in Chapter 7). These specifications simply list events that are detected at the system inputs (the system stimuli), the corresponding events that are

required to occur at the system outputs (the system responses), and the timing constraints within which the system must generate these responses.

Finally, Figure 3.1 shows the requirements dictionary. Although it is shown outside the main structure, the RD serves a vital role throughout the model since it contains precise definitions of all the data and control flows in all their levels of decomposition.

The remaining chapters of Part II describe each component of the method in detail, but it is important not to lose track of how they fit into the overall structure of Figure 3.1 as you study them.

For the purposes of this description, we refer to the DFDs and PSPECs as the *process model*, the CFDs and CSPECs as the *control model*, and the two of them with the requirements dictionary and timing specifications as the *requirements model*. This is in keeping with the evolution of the methods: The process model corresponds to the classical structured analysis model; the control model is the part we added, derived from finite state machine theory; and the requirements model is the complete, integrated combination of the two.

Index

- Actions, 15, 83–88, 93, 97, 178–79
 - logic, 93, 94, 184
 - naming, 177, 189
- Activation table: *see* Process activation table
- Architecture communication channels:
 - AIS and, 229–32
 - redundancy of, in AID, 221–22
- Architecture context diagram (ACD), 203–204, 211–13, 263–64, 382–84
 - components in, 211
 - example, 212, 213, 383
 - for Automobile Management System, 264, 322
- Architecture dictionary (AD), 20, 25, 203–204, 225–26, 233–34, 388–89
 - of Automobile Management System, 233, 325
 - of vending machine example, 22, 23, 358
- Architecture flow diagram (AFD), 20, 25, 203–204, 213–20, 264–68
 - decomposition of, 215–17
 - examples, 215, 216
 - naming and numbering rules, 384, 386
 - of Automobile Management System, 218, 220, 266, 277–78, 322
 - of vending machine example, 21, 357
 - structure chart and, 282
- Architecture interconnect diagram (AID), 21–22, 25, 203–204, 213, 218–24, 264–68, 385–87
 - decomposition of, 215–16
 - layering rules, 222
 - naming and numbering rules, 222, 385, 386
 - of Automobile Management System, 277–78, 323
 - of vending machine example, 22, 357
 - redundancy in, 219, 221–24
- Architecture interconnect specification (AIS), 22, 23, 203–204, 229–32, 387–89
 - of Automobile Management System, 231, 324
 - of vending machine example, 23, 359
- Architecture model, 7, 12, 19–27, 99, 382–91
 - Automobile Management System, 319–25
 - balancing, 238, 268–71, 287–88, 368, 390–91
 - components, 203–204
 - decomposition of, 215
 - hierarchy of, 207
 - illustration, 24
 - implementation of, in hardware and software, 235–40
 - of Home Heating System example, 340–41
 - summary, 24–25, 203–10, 238–40
 - symbols of, 207–10
 - vending machine example, 355–59
- Architecture module, 20, 208
 - automobile management computer module example, 235–36, 274–77, 280–81
 - decomposition of, 215
 - defined, 366
 - implementation of, 235–36
 - in ACD, 211
 - in AID, 219, 221–24
 - naming rule, 208, 366
 - number in AFD, 214
 - partitioning of requirements, 273–85
 - redundancy of, 219, 221
 - symbols, 208, 366
 - timing constraints and, 232–33
- Architecture module specification (AMS), 20, 25, 215, 225–29
 - of Automobile Management System, 268, 324
 - of vending machine example, 21
 - traceability matrix and, 226, 229–30
- Architecture template, 204–207, 217, 248–62, 279–82, 383, 400

- benefits of, 260–62
- for Automobile Management System, 206
- software requirements and, 279–82
- Automobile management computer module, 235–36, 274–77, 280–81
- Automobile Management System, 295–325
 - ACD example, 213, 322
 - AD example, 234, 325
 - AFD example, 218, 220, 322
 - AID example, 223, 323
 - AIS for, 324
 - AMS example, 226–27, 324
 - architecture model of, 239, 319–50
 - architecture template for, 206
 - control diagrams for, 301
 - CSPEC for, 303, 308, 312, 314
 - data vs. control issues, 128–29
 - enhanced CFD for, 258, 321
 - enhanced DFD for, 257, 320
 - flow names in, 148
 - hierarchical nature of, 273
 - input and output processing for, 249–51
 - PSPEC of, 160, 305–306, 309, 312, 315
 - RD for, 316–18
 - response time specification, 100
 - safety requirements in, 245
 - sequential machine in, 179–82
 - STD in, 85, 87
 - STM in, 88, 89
 - system context, 129–32
 - timing specification, 304
 - top-level partitioning, 133–36
 - traceability matrix for, 230
 - transient signal in, 163
 - user interface processing, 251–54
- Automated tools, 6, 206, 404
 - used for balancing, 49, 140, 166
 - used for dictionary, 107, 140, 197, 199, 395
- Avionics system example, 4–7
- Backus-Naur form, 105–107, 196–99
- Balancing:
 - architecture model, 268–71, 390–91
 - requirements model, 47–49, 113, 123, 379–81
 - (*see also* specific entries)
- Balzer, R., 139, 404
- Bar symbol, 65–66, 71, 89, 93, 185, 364, 371
- Boehm, B., 29, 403
- Boolean equation form, 79–80, 97, 171, 198
- Booth, T., 86, 173, 403
- CASE tools, 6, 404
- Combinational control specification, 90–93, 169–78
- Combinational machine, 78–82, 114, 152, 157, 169–76
 - CSPEC for, 375–77
- Comment:
 - in PSPEC, 55, 167
 - in requirements dictionary, 106–107
- Conditional construct, 166
- Consistency checking, 5, 36
 - of architecture model, 268–71
 - of CSPECs, 71
 - of DFDs, 49, 50
 - of requirements model, 36, 48, 50, 52, 113, 115
- Constantine, L., 6, 282, 404
- Context diagram:
 - context process in, 137
 - control, 38–39, 41, 61–63, 76, 301, 369
 - data, 38, 39, 41–44, 59, 63, 368
 - level 1 and, 130–31
 - nonprimitive flows and, 105–107
 - primitive flows in, 67–68, 104–105
 - timing requirements and, 99
 - (*see also* Control context diagram, Data context diagram)
- Context process, 43–44, 63, 137
- Continuous (analog) machine, 77–78, 97
 - digital computer and, 78
 - PSPECs and, 77
- Continuous signals, 67–68, 77–78, 103, 125, 165
 - example of primitive in RD, 104, 196–97
- Control context diagram (CCD), 38–39, 41, 61–63, 76, 301, 369–70
 - of Automobile Management System, 301
 - of vending machine example, 344
- Control flow, 13, 41, 61, 64–68, 103
 - defined in requirements dictionary, 67, 104, 115
 - determined in requirements model, 125–29
 - graphic convention, 61, 63, 67, 364
 - naming, 364
 - types and attributes, 195–97
- Control flow diagram (CFD), 13, 15, 19, 35, 64–68, 76, 113, 371–72
 - allocation in architecture model, 227–29
 - balancing, 64, 113
 - composite example of, 33
 - control store on, 70, 76, 150, 365
 - control flow on, 15–16, 67–68
 - control signals and, 15–16, 67

- Control flow diagram (CFD), 13, 15, 19, 35, 64–68, 76, 113, 371–72
 - allocation in architecture model, 227–29
 - balancing, 64
 - control store on, 70, 150
 - decomposition of, 39, 65
 - interface with CSPEC (bar symbol), 65–66, 71, 89, 90, 185, 364, 371–72
 - leveled, 39, 64, 113
 - link to DFD, 69, 71
 - naming and numbering, 64, 113–14, 372–73
 - of automobile management computer module, 275, 281
 - of Automobile Management System, 258, 302ff.
 - of vending machine example, 16, 356 sample, 373
- Control model, 13, 15–19, 40, 61–76
 - control-intensive system and, 153–54
 - implementation details and, 151–52
 - purpose of, 152
 - vending machine example, 13–18
- Control outputs, 73
 - (*see also* Process controls)
- Control signal, 15–16, 67
 - RD and, 16
 - symbol for, on CFD, 15
 - (*see also* Control flow)
- Control specification (CSPEC), 15, 16, 19, 39, 64–67, 70–73
 - balancing, 95, 368, 379–81
 - bar symbol on, 65–66, 71, 89, 93, 185, 364, 371–72
 - combinational vs. sequential control, 90–93, 169–78, 376–77
 - composite, 93–94, 184
 - decision table and, 171–74
 - finite state machines and, 39, 73, 89–97, 188
 - floating, 187
 - interface with CFD, 65–66, 71, 89, 90, 185, 364
 - multi-sheet conventions for, 95–97, 114, 182–85
 - naming and numbering, 71, 114
 - of Automobile Management System, 303, 308, 312, 314
 - of combinational machine, 79–82, 170–76, 375–77
 - of vending machine example, 17, 20
 - PAT in, 174–75
 - placement of, in model, 185–86
 - primitive, 153–54
 - process controls and, 73–76, 90
 - purpose of, 73, 114, 169
 - rule for allocation in architecture model, 227–29
 - state transition diagram and, 65, 90, 92, 93
 - timing requirements in, 188
 - transaction center and, 75, 175–76
 - users' guide for, 95, 114, 153, 185
- Control store, 70, 76, 149–50
 - (*see also* Data store)
 - naming, 70, 150, 365
- Control structure, 5
 - balancing, 76
 - determining in requirements model, 111–13, 125–29
 - illustrated, 62, 112
- Cruise control system example: *see* Automobile Management System
- Data abstraction, 137, 209
 - (*see also* Information abstraction)
- Data condition, 19, 69–70, 76, 114
 - example, 69
- Data context diagram (DCD), 38, 39, 41–44, 368–69
 - external primitive flows in, 191
 - of Automobile Management System, 301
 - of vending machine example, 344
- Data flow, 41, 43, 44, 49–52, 59
 - arcs, 50–51
 - balancing, 47–49, 59, 143–44
 - consistency checking of, 49, 50, 52
 - data stores and, 52, 150
 - decomposition of, 50, 143
 - defined, 363
 - determined from user requirements statement, 125–29
 - graphic conventions, 43, 50–51, 143–47, 364
 - in requirements dictionary, 13, 103, 115, 144–45
 - naming rules, 46, 50, 143–44, 147–49, 363
 - primitive, 49
 - splitting and merging, 50, 143
 - types and attributes, 194–97
- Data flow diagram (DFD), 13, 19, 35, 44–53, 114, 370–71
 - allocation in architecture model, 227–29
 - composite example of, 133
 - data signal and, 67
 - decomposition of, 39, 45, 47

- leveling and balancing, 39, 45, 47–49, 59, 113
- link to CFD, 69, 71
- naming and numbering, 49, 370–74
- of automobile management computer module, 274, 280
- of Automobile Management System, 302ff.
- of vending machine example, 14
- parent/child relationship, 45, 47–49
- sample, 45, 371
- Data modeling: *see* Information modeling
- Data process, determining, 128
- Data store, 52–53, 59, 149–50
 - defined, 44, 365
 - flows and, 150
 - naming rules, 52–53, 150, 365
 - symbol, 44
- Data structure:
 - determined in requirements model, 125–29
- Date, C., 194, 403
- Decision table (DT), 73, 79–82, 97, 114, 166, 375
 - Home Heating System example, 81–82
 - in CSPEC, 171–74
 - luggage lock example, 79
 - PAT, 90–91, 174–75
 - STM and, 93–94
 - transaction center and, 175–76
- Decomposition, 36, 137–38
 - of flows, 140–47
 - of processes, 45, 139–40
 - 7 ± 2 principle, 138–39
- DeMarco, T., 5, 35, 36, 37, 56, 403
 - structured analysis method of, 5, 95
- Design, 243–44
 - constraints, 203
 - internal timing requirement and, 98, 102, 114, 190
 - phase, 155
- Discrete signal, 67–68, 77–78, 103–104, 125, 165
 - example of primitive in RD, 104, 196–97
- Don't care condition, 80, 172, 173, 189, 194
- Entity, in ACD, 211
- Equation, in PSPEC, 159–60
- Event, 15, 83–88, 93, 97, 178–79
 - asynchronous, 150
 - logic, 93, 94, 184
 - naming, 189
 - response time, 99–102
- Feedback control loop, 71–73, 114
- Finite state machine, 77–97, 125, 153
 - combinational, 78–82, 152, 157, 169–76, 375–77
 - control model and, 40, 70–73, 76
 - control structures and, 5, 113
 - CSPEC and, 15, 39, 73, 89–97
 - model, 77
 - sequential, 78, 83–89, 176–82, 375, 378–79
 - theory, 5, 36, 89
- Flavin, M., 403
- Flight management system example, 78
 - response time specification, 100
- Flow arcs, 43–44, 50–51
 - symbol on CFD, 65
 - symbol on DFD, 16
- Flow diagram: *see* Control flow diagram, Data flow diagram
- Flows:
 - allocation in AMS, 226–29
 - balancing, 143–44
 - (*see also* Control flow, Data flow)
- Functional primitive: *see* Primitive process
- Functional requirements, 3–10, 13, 19
 - (*see also* Control model, Process model, Timing requirements)
- Fundamental entity, 208
- Hardware design, 276–78
 - requirements allocated to hardware, 277
- Hardware/software implementation of architecture model, 235–36, 276–78
 - example, 236
- Hardware/software models, 288–91
- Hierarchical representation, 4, 9–10, 28–29, 36, 39
 - of Automobile Management System, 273
- Hofstadter, D., 216, 403
- Home Heating System example, 326–41
- Hopcroft, J., 86, 403
- Implementation, system, 10, 12, 56, 114
 - architecture, 229
 - of process model, 56
 - timing constraints and, 232–33
- Information abstraction, 5, 36, 113, 137–38
- Information flow channel, 213
 - in AFD and AIS, 25
 - in AID, 219
 - naming rule, 209, 367
 - symbol, 209, 367

- Information flow vector, 208–209
 - defined, 366
 - in ACD, 211
 - naming rule, 209, 367
 - symbol, 209, 367
- Information hiding, 36, 137, 185
- Information modeling, 13, 398–401, 404
- Input and output processing, 26, 204–207
- Input-to-output response time, 99–102, 113
- Interface definition, 7, 26, 203–207, 260
- Iteration:
 - of architecture model, 203, 286–91
 - of development process, 10, 26–30
 - of requirements model, 123, 139
- Leveling:
 - CFDs, 39, 64, 113
 - DFDs, 39, 45, 47–49, 59, 113
 - PSPECs, 53, 70
- Logical function, 43
- Maintenance processing, in architecture model, 26–27, 204–205, 254–56
- Martin, J., 194, 403
- McMenamin, S., 12, 150, 403
- Mealy sequential machine model, 86, 314
- Mellor, S., 86, 404
- Methods Team, 6
- Military standards (documentation), 396–97
- Miller, G., 138–39, 403
- Modeling process, 286
- Module, 20, 24
 - automobile management computer example, 274
 - software, 282–85
 - (*see also* Architecture flow diagram)
- Module specification: *see* Architecture module specification
- Moore sequential machine model, 86, 314
- Myers, G., 6, 283, 403
- Naming rules:
 - abbreviations used, 148–49
 - in AFD, 384, 386
 - of processes and flows, 46, 50, 138, 140, 147–49, 363–67
- Numbering system:
 - of flow diagrams, 49, 370–75
 - of multi-sheet CSPECs, 95–97, 114
- Objects (flows), naming, 46, 148
- Operational requirements specification, 192
- Page-Jones, M., 128, 283, 404
- Palmer, J., 12, 150, 403
- Parnas, D., 36, 137, 139, 283, 404
- Partitioning, 30
 - in Automobile Management System, 133–36
 - modeling process, 286–91
 - of architecture model for hw/sw implementation, 235–36, 276–78, 289
 - of requirements model, 113
- Physical configuration, 24, 25
- Physical modules, 203–10
- Physical requirements, 3, 7, 43, 203–210
- Primitive flow, 67–68, 103–105
 - attributes of, 104–105, 196–97
 - decomposing groups of, 140
 - defined, 103, 194
 - examples of, 104
 - grouping, 140, 194
 - symbol in dictionary, 106
- Primitive network, 58, 112
- Primitive process, 13, 39, 48, 53, 58, 59, 111–15
 - CFD and, 65
 - network of, 58, 112
 - PSPEC and, 13, 65
- Process, 45–46
 - allocation in AMS, 226–29
 - controls, 73–76, 111
 - data, 128
 - defined, 365
 - functionally identical, 150–51
 - naming rule, 46, 147–49, 365
 - numbering system, 49
 - on CCD, 61
 - on CFD, 64–65
 - parallel, 56
 - physical, 208
 - primitive, 13, 39, 49, 53, 58–59, 65, 111–15
 - sequential, 55
 - symbol for, 41
 - timing requirements in, 191
- Process activation table (PAT), 15, 90, 91, 174–75, 375
 - example, 91
 - of vending machine example, 17, 20
- Process activators, 73
 - CSPEC and, 186–87
- Process controls, 73–76, 90
 - in leveled DFDs, 74
- Processing, input and output:
 - Automobile Management System, 249–51

- in architecture model, 249–51
- Processing, user interface, 26–27, 251–54
- Process model, 13, 40, 41–60
 - interpreting, 56–59
- Process specification (PSPEC), 13, 19, 35, 39, 53–55, 59, 372, 374
 - balancing, 53, 70, 159, 166
 - comment in, 55
 - conditional statement in, 125–26
 - continuous machines and, 77
 - data condition and, 69–70
 - data signal and, 67
 - examples, 54, 129, 374
 - functional primitives, 53
 - leveling and balancing, 53, 70
 - naming and numbering rules, 374–75
 - of Automobile Management System, 160, 305–306, 309, 312, 315
 - of vending machine example, 14
 - role of, 158–59
 - structured English in, 53, 55–56, 165–66
 - transient signal in, 162–63
 - types of, 159–62
 - user requirements and, 55, 158
- Process structure:
 - categories, 127
 - in requirements model, 113
- Project dictionary:
 - abbreviations and, 148–49
 - list of word types, 166
- Prototyping, 30
- Putnam, L., 404
- Putnam model, 6, 404
- Real-time systems, 36
 - characteristics of, 3, 77, 98
 - development methods for, 11ff.
- Redundancy requirement, 26, 27
 - data store and, 53
 - in AFD, 218
 - in AID, 219, 221–24
 - in Automobile Management System, 277
- Reliability requirement, 222–23, 243
- Requirements:
 - data vs. control, 125–29, 152
 - defined, 10, 12
 - functional, 3, 99
 - physical, 3, 24
 - processing, in PSPECs, 158–68
 - timing, 39–40, 98–102, 113–15, 190–93, 232–33, 244
- Requirements dictionary (RD), 13, 19, 35, 40, 103–10, 377–79
 - attributes of, 194–97
 - cockpit display example, 105
 - data flow types in, 194–96
 - flow decomposition in, 381
 - flows in, 67, 103, 115
 - in a computerized data base, 107–108, 194
 - indented explosion of, 107, 109
 - naming rules, 379
 - of Automobile Management System, 316–18
 - of vending machine example, 15, 18, 354
 - primitives in, 103–107
 - symbols in, 105–107, 198–99, 378
 - timing signals in, 190–93
- Requirements model, 4, 7, 12, 13–19, 25–27, 35–40, 111–15
 - adding physical constraints, 248–62
 - allocation of components in Automobile Management System, 226–29
 - builders, 120–23
 - components of, 37–40, 113–15, 367–81
 - consistency checking of, 36, 49, 50, 52, 113, 115
 - decomposition and, 45, 122–23, 137–38
 - enhanced, 248–62
 - implementation details and, 152
 - leveling and balancing, 47–49, 113, 379–81
 - naming and numbering in, 47ff., 113
 - of Home Heating System, 330–39
 - of vending machine example, 344–54
 - overview of, 18–19, 35–40, 113–15
 - primitives in, 113
 - redundancy and, 26–27, 53
 - role of automata theory, 89
 - self-indexing, 5, 36, 115
 - signals in, 162–63
 - timing in, 190–93
 - tools of, 35
 - users, 119–21
- Response time specification, 13, 19, 99–102, 115
 - (*see also* Timing specification)
- Schuldt, G., 13, 404
- Self-test processing, 26–27, 254–55
- Sequential control specification, 90, 92, 93
- Sequential machine, 78, 83–89, 95, 97, 114, 152, 157, 176–82, 375
 - combination lock example, 83–84
 - CSPEC for, 378, 379
 - math representation, 178
 - Mealy and Moore models, 86, 178–79

- Seven-plus-or-minus-two principle, 138–39, 177, 214, 370, 403
- Signals:
 - attributes, 196–97
 - continuous vs. transient, 162–63, 165
 - control vs. data, 67–68, 70, 76, 111–12
 - discrete vs. continuous, 67–68, 77–78, 103–104, 165
 - external primitive, 99, 104
 - types, 127, 196
- Sink, 43, 44
- Software architecture, 277, 282–85
- Source, 43, 44
- Specification model, system, 24, 25–27
- STARS Methodology Conference, 295
- State, 85–89, 97, 177
 - naming, 189
- State transition diagram (STD), 15, 65, 73, 83–86, 90–97, 177–78, 375, 379
 - of Automobile Management System, 182, 303
 - of combination lock example, 84
 - of vending machine example, 17, 20
- State transition matrix (STM), 73, 87–89, 93, 97, 114, 177–78
 - examples, 88, 89
 - multi-sheet CSPECs and, 182–85
- State transition table (STT), 73, 87, 93, 97, 114, 177, 184, 375
 - example, 87
- Store: *see* Control store, Data store
- Structure chart, 282
- Structured analysis, 5, 35–37, 43, 52, 53, 95
 - control requirements and, 188
 - leveling and balancing principles of, 93
 - primitive network and, 113
 - process model and, 40
- Structured design, 6, 36, 282–84
- Structured documentation, 392–97
- Structured English, 53, 55–56, 59
 - examples, 57
 - in PSPECs, 159, 165–66
 - structure of, 57
- Structured methods:
 - defined, 11ff.
 - history of, 4–8, 36–37
 - iteration and, 27
- Structured programming, 36–37
- Swartout, W., 139, 404
- System analysts, role of, 120–23
- System context, 44, 122, 129–32
- System design:
 - building the architecture model, 243–45
 - factors, 98–99, 243–45
- System life cycle, 8–10
 - cost, 31
 - design, 56
 - implementation phase, 8, 32, 56, 261
 - requirements definition phase, 56, 120
- System maintenance, 26, 29
 - cost, 10
 - phase, 261
- System model, 287–88
- Systems:
 - control-intensive, 153–54
 - hierarchy of, 4, 10, 28–29, 246–47
- System scope, 44, 122, 129–32
- Systems development process:
 - illustration of, 26
 - iteration and, 10, 26, 30, 139
 - life cycle of, 27–31
- System specification:
 - hierarchical nature, 3, 28–29
 - leveled, 4, 28–29, 39
 - methods of, 3ff.
 - model, 24, 25–27
 - (*see also* Architecture model, Requirements model)
- Tabular PSPEC, 15, 161
 - (*see also* Process activation table, State transition diagram)
- Technology-dependent model, 12, 26, 244, 257–60
 - (*see also* Architecture model)
- Technology-nonspecific model, 248–62
 - (*see also* Requirements model)
- Terminator, 368
 - defined, 365
 - naming, 366
 - on ACD, 211
 - on CCD, 61, 63
 - on DCD, 41, 43–44
 - symbol, 41
- Testing requirements, 9, 26, 29, 31
 - phase, 261
- Timing diagram, 101, 115
- Timing requirements, 39–40, 98–102, 113–15
 - 190–93
 - in architecture model, 229, 232–33, 244
 - in CSPEC, 164, 188, 191
 - in PSPEC, 164
 - in requirements model, 39–40, 98–102, 113–15
 - repetition rate, 99, 102, 114–15, 188, 190–93

- response time, 99–102, 114–15, 190–93
- Timing specification, 32, 39–40, 99–102, 115, 190–93, 229, 232–33, 376
 - balancing, 193
 - of Automobile Management System, 304
 - of vending machine example, 347
(*see also* Response time specification)
- Traceability matrix, 226, 229–30, 388
 - of Automobile Management System, 230
 - of vending machine example, 358
- Transaction center, 46
 - control, 75, 175–76
- Transient signal, 162–63
- Transition arcs, 85–86, 178
- Truth table: *see* Decision table

- Ullman, J., 86, 403
- Understandability, 5, 123, 149, 156, 159, 177, 214
- User interface:
 - in Automobile Management System, 251–54
 - processing, 26–27, 251–54
 - specification of, 260
- User requirements, 37, 98, 119–23, 124–25
 - PSPEC and, 55, 158
- Users' guide, to CSPECs, 95, 114, 153, 185

- Vending machine example, 12–23, 342–59

- Ward and Mellor convention, 86, 404
- Ward, P., 86, 404

- Yourdon, E., 6, 283, 404