

Lecture Notes in Computer Science 2231
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Tokyo

Alessandro Pasetti

Software Frameworks and Embedded Control Systems



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Author

Alessandro Pasetti
ETH-Zentrum, Institut für Automatik
Physikstraße 3, 8092 Zürich, Switzerland
E-mail: pasetti@pnp-software.com

Dissertation der Universität Konstanz
Tag der mündlichen Prüfung: 8. Juni 2001

Referent: Prof. Dr. Wolfgang Pree
Referent: Prof. Dr. Kai Koskimies

Cataloging-in-Publication Data applied for

Pasetti, Alessandro:
Software frameworks and embedded control systems / Alessandro Pasetti. -
Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ;
Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; Vol. 2231)
Zugl.: Konstanz, Univ., Diss., 2001
ISBN 3-540-43189-6

CR Subject Classification (1998): C.3, D.2.11, D.2.13, J.2

ISSN 0302-9743
ISBN 3-540-43189-6 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2002
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10845761 06/3142 5 4 3 2 1 0

Preface

Professionally, the defining experience of my life was a period of nine years as control systems engineer with the European Space Agency (ESA). Given this background, it was perhaps inevitable that when, in 1999, I decided to start a new career as a software engineer I should choose as my area of work software architectures for embedded control systems. As it turned out, this was a lucky choice. After decades of academic neglect, embedded software is now beginning to receive the attention to which its ubiquity in everyday devices and its presence at the heart of many safety-critical systems entitles it. The novelty of the field means that much of the work that needs to be done consists in transferring to embedded systems the technologies and development practices that have become so prevalent in other fields. Following this line of research, I have concentrated on applying object-oriented software frameworks to embedded control systems. Framework technology has enjoyed wide currency for at least five years. Although it has proven its worth as a software reuse technique in many domains, it has been shunned in the embedded world mainly because it tends to be associated with lavish use of CPU and memory, both of which have traditionally been in short supply in embedded systems. Times are changing, though. Hardware advances are expanding the resources available to embedded systems and their adoption of framework technology no longer looks far-fetched or unrealistic.

The first objective of this book is to show how object-oriented software frameworks can be applied to embedded control systems. Software design may not be an art but it is certainly more of a craft than a science and teaching by example is in my view the best way to transfer design experience. I have accordingly chosen a case study as the means to make my point that framework technology can aid the development of embedded control software. The target application of the case study is the attitude and orbit control system of satellites. This domain is broader and less structured than the domains to which frameworks are normally applied. This led me to develop a concept of frameworks that was rather different from that proposed by other authors in that it gives a more prominent, or at least a more explicit, role to domain-specific design patterns. The second objective of the book is to discuss this new view of software frameworks and to present some methodological concepts that I believe can facilitate their development.

This book is therefore written with two audiences in mind: developers of embedded control software will hopefully be inspired by the case study in the second part of the book to apply framework technology to their systems, while researchers on software architectures might be moved to rethink their conceptualization of frameworks by the material presented in first part of the book. The link between

first and second part lies in the use of the concepts introduced in the methodological sections of the book to describe the case study framework.

One of the functions of a preface is to offer a path through the book to prospective readers. The key chapters are 3 and 4, introducing the concepts of framelet and implementation case, and chapter 8, giving an overview of the design principles behind the satellite framework. Hurried readers could limit themselves to these three fairly self-contained chapters, which would suffice to give them a rough idea of both the methodological and technological contributions made by the book. The framework concept and the methodological guidelines for framework design are presented in chapters 3 to 7. The case study is covered from chapter 8 to the end of the book. Readers who are not familiar with satellite control systems should read chapter 2 in order to acquire the domain background necessary for an understanding of the case study.

The satellite framework is presented as a set of design patterns that have been specifically tailored to the needs of satellite control systems. Each chapter in the case study part of the book covers one or a small number of related design patterns. These chapters are to a large extent independent of each other and could be read in any order or even in isolation of each other. One way to see this book (or at least its second part) is as a repository of design patterns for embedded control software development.

The appendix at the end of the book contains synoptic tables listing the architectural constructs offered by the satellite framework. The lists are cross-referenced and can serve as an aid to navigate the framework design presented in the book.

This book describes the satellite framework at the “design pattern level”. The satellite framework also exists as a publicly available prototype and its full design documentation and code can be downloaded from the project web site¹. Readers are invited to access this material but they should bear in mind that the code is offered without any guarantee of correctness and that its quality and completeness are those required for a proof-of-concept prototype. They should also be aware that the satellite framework project is a “living project” and its web site is constantly being updated as the framework design and implementation are reviewed and modified. Hence, as time progresses, inconsistencies will inevitably arise with the content of this book.

Having explained what the book offers to its readers, I should write a few words about what it expects of them. The focus is on *object-oriented* frameworks but no effort is made to explain the principles of object-oriented design which are assumed known to the readers. Similarly, familiarity with the design pattern concept is both assumed and necessary. Some background knowledge of framework technology would be useful but is not essential. No prior knowledge of satellite control systems is assumed, all the necessary background being offered in chapter 2. I have made every effort to make the case study as self-contained as possible

¹ Its address is subject to change. The site can be found with any internet search engine by searching for “AOCS Framework”.

but I suspect that full appreciation of the solutions it offers requires at least a passing acquaintance with embedded control systems. Class diagrams are in standard UML and pseudo-code examples are written using C++ syntax with the addition of the interface construct from Java which I find simply too useful to ignore.

Finally, I wish to thank Wolfgang Pree and Kai Koskimies who reviewed the first draft of this book and Jean-Loup Terraillon, Ton van Overbeek, Richard Creasey, and David Soo who, in various capacities, supported the satellite framework project.

August 2001

Alessandro Pasetti

Contents

1	Introduction and Context	1
1.1	The Embedded Software Problem	1
1.2	Empowering Application Specialists	6
1.3	The Component Software Challenge	9
1.4	Objectives and Contributions	13
2	Attitude and Orbit Control Systems (AOCS)	17
2.1	AOCS Systems	17
2.1.1	AOCS Functions	19
2.1.2	AOCS Operational Modes	23
2.1.3	AOCS Units	25
2.1.4	The AOCS Software	26
2.2	The AOCS and Other Control Systems	27
3	Software Frameworks	29
3.1	Frameworks, Components, and Inheritance	33
3.2	A More Complete View of Software Frameworks	34
3.3	Frameworks and Autocoding Tools	40
3.4	The Methodological Problem	41
3.4.1	Design Patterns and Abstract Interfaces Vs. Concrete Objects	42
3.4.2	Support for Design Patterns and Hot-Spots	43
3.4.3	Iterative System Specification	44
3.4.4	Design and Architecture	44
3.4.5	A Methodology for Frameworks	46
4	Framelets and Implementation Cases	49
4.1	The Framelet Concept	49
4.1.1	Implications for the Design Process	52
4.1.2	Framelets, Design, and Architecture	52
4.1.3	Framelets and Aspect-Oriented Programming	54
4.1.4	Framelet Features	56
4.1.5	Framelet Constructs	57
4.1.6	Framelet Heuristics	57
4.1.7	Framelets in the AOCS Framework	59
4.1.8	Related Approaches	60
4.2	The Implementation Case Concept	62
4.2.1	The Three Roles of Implementation Cases	63

4.2.2	Implementation Case Scenarios and Extensions.....	64
4.2.3	Description of Implementation Cases.....	65
5	Framework Specification.....	69
5.1	How Important Is Specification?	69
5.2	An Alternative Specification Approach.....	71
5.3	An Example from the AOCS Case Study	75
6	Framework Design	79
6.1	Overall Approach.....	79
6.2	Alternative Approaches	82
6.3	The Framework Concept Definition Phase.....	83
6.3.1	Definition of General Design Principles.....	84
6.3.2	Identification of Domain Abstractions	85
6.3.3	Construction of the Framework Domain Model.....	85
6.3.4	Identification of Framework Hot-Spots.....	86
6.3.5	Identification of Framework Design Patterns.....	86
6.3.6	Framelet Identification	87
6.3.7	Identification of Implementation Cases.....	87
6.3.8	Identification of Alternative Solutions	87
6.4	Framelet Concept Definition	87
6.4.1	Identification of Exported Interfaces and Implementations.....	88
6.4.2	Identification of Framelet Hot-Spots	88
6.4.3	Definition of Applicable Design Patterns.....	89
6.4.4	Definition of Framelet Contribution to the Framework.....	89
6.4.5	Definition of Framelet Contribution to Reusability.....	89
6.5	Framelet Architectural Definition.....	90
6.5.1	Definition of Framelet Constructs	90
6.5.2	Definition of Framelet Hot-Spots	91
6.5.3	Definition of Framelet Functionalities.....	91
6.6	Framework Design Description	92
6.6.1	Framework Concept Definition	92
6.6.2	Framelet Concept Definition	92
6.6.3	Framelet Architectural Definition.....	94
6.6.4	Overview of Design Description Techniques.....	94
6.6.5	Framelet Interactions	95
6.6.6	Examples from AOCS Case Study	96
7	The User's Perspective	99
7.1	A Reuse-Driven Development Process.....	99
7.2	The Functionality Concept.....	101
7.2.1	Functionality Types	102
7.2.2	Mapping Functionalities to Architectural Constructs.....	104
7.2.3	Completeness of Description.....	105
7.2.4	Mapping Requirements to Functionalities.....	106

7.2.5	Functionalities in the AOCS Framework.....	108
7.2.6	Alternative Approaches	108
8	General Structure of the AOCS Framework.....	111
8.1	The RTOS Example.....	111
8.2	The Lesson for the AOCS.....	113
8.3	Telemetry Management in the AOCS Framework	115
8.4	Controller Management in the AOCS Framework	117
8.5	The Manager Meta-pattern	120
8.6	Overall Structure.....	121
8.7	Architectural Infrastructure.....	123
8.8	Hierarchies of Design Patterns.....	123
8.9	The Framework Design Process	125
8.10	From Design to Architecture	126
8.11	Related Work.....	128
9	General Design Principles.....	131
9.1	Boundary Conditions	131
9.2	An Object-Oriented Framework	131
9.3	A Component-Based Framework.....	132
9.4	Delegation of Responsibility.....	133
9.5	Multiple Implementation Inheritance.....	133
9.6	External Interfaces	133
9.7	Basic Classes	133
9.8	Time Management	134
9.9	Language Selection.....	135
9.10	Execution Time Predictability	135
9.11	Scheduling	136
9.12	A Framelet-Based Framework.....	138
10	The System Management Framelet	141
10.1	The System Management Design Pattern.....	141
10.2	The System Reset Function	142
10.3	The System Configuration Check Function.....	143
10.4	Storage of Configuration Data	144
10.5	Reusability.....	144
11	The Object Monitoring Framelet.....	147
11.1	Properties and Property Objects	147
11.2	Change Objects.....	148
11.3	The Monitoring Design Patterns.....	150
11.3.1	The Direct Monitoring Design Pattern	150
11.3.2	The Monitoring through Change Notification Design Pattern.....	151
11.4	Implementation Case Example – 1	153
11.5	Implementation Case Example – 2	154
11.6	Alternative Solutions	155
11.7	Reusability.....	156

12	The Operational Mode Management Framelet	159
12.1	The Mode Management Design Pattern	160
12.2	Mode Change Actions	163
12.3	Coordination of Operational Mode Changes	163
12.4	AOCS Mission Mode Manager	164
12.5	Reusability	165
13	The Intercomponent Communication Framelet.....	167
13.1	The Shared Event Design Pattern	167
13.2	The Shared Data Design Pattern	169
13.3	AOCS Data	170
13.4	Data Pools	173
13.5	Implementation Case Example – 1	174
13.6	Implementation Case Example – 2	175
13.7	Implementation Case Example – 3	177
13.8	Alternative Implementations	178
13.9	Reusability	180
14	The Sequential Data Processing Framelet.....	183
14.1	Control Channels	184
14.2	The Control Channel Design Pattern	186
14.3	Implementation Case Example	189
14.4	Alternative Solutions	190
14.5	Reusability	192
15	The AOCS Unit Framelet	193
15.1	Abstract Unit Model	194
15.2	The AOCSUnit Class	197
15.2.1	The AOCS Unit Housekeeping and Functional Interfaces	198
15.3	Unit Triggers	200
15.4	Hardware Unit Components	202
15.5	Fictitious AOCS Units	203
15.5.1	The Fictitious Unit Design Pattern	204
15.6	Implementation Case Example	205
15.7	Reusability	207
16	The Reconfiguration Management Framelet.....	209
16.1	Some Definitions	209
16.2	The Reconfiguration Management Design Pattern	210
16.2.1	Intersection and Nesting of Reconfiguration Groups	213
16.2.2	Direct Access to Redundant Components	214
16.2.3	Preservation of Configuration Data	215
16.3	Reusability	215

17	The Manoeuvre Management Framelet	217
17.1	Manoeuvre Components	217
17.2	The Manoeuvre Design Pattern	218
17.3	Manoeuvre Initiation	220
17.4	Alternative Solution.....	220
17.5	Reusability.....	221
18	The Failure Detection Management Framelet	223
18.1	Overall Approach	223
18.2	Failure Detection Checks.....	224
18.2.1	Consistency Checks	224
18.2.2	Property Monitoring	225
18.3	The Failure Detection Design Pattern.....	227
18.4	Alternative Approaches	228
18.5	Failure Isolation.....	228
18.6	Reusability.....	230
19	The Failure Recovery Management Framelet	233
19.1	Failure Recovery Actions	233
19.2	Failure Recovery Strategy	234
19.3	Failure Recovery Design Pattern	235
19.4	Implementation Case Example – 1	237
19.5	Implementation Case Example – 2	239
19.6	Alternative Implementation.....	241
19.7	Reusability.....	241
20	The Telecommand Management Framelet	243
20.1	The Telecommand Management Design Pattern.....	243
20.2	The Telecommand Transaction Design Pattern.....	246
20.3	Telecommand Loading	247
20.3.1	Implementation Considerations	249
20.4	Implementation Case Example.....	250
20.5	Reusability.....	251
21	The Telemetry Management Framelet	253
21.1	The Telemetry Management Design Pattern	253
21.2	Implementation Case Example.....	256
21.3	Functionality List Example.....	257
21.4	Alternative Implementation.....	260
21.5	Reusability.....	261
22	The Controller Management Framelet.....	263
22.1	The Controller Design Pattern	263
22.2	The Controller Abstraction.....	264
22.3	Implementation Case Example.....	266
22.4	Reusability.....	268

23	The Framework Instantiation Process	271
23.1	Step-by-Step Instantiation	271
23.2	Framework Overheads.....	276
Appendix		279
References		285
Index		291