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 site[1]. 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