Preface

In a distributed computing system, *middleware* is defined as the software layer that lies between the operating system and the applications on each site of the system. Its role is to make application development easier, by providing common programming abstractions, by masking the heterogeneity and the distribution of the underlying hardware and operating systems, and by hiding low-level programming details.

Stimulated by the growth of network-based applications, middleware technologies are taking an increasing importance. They cover a wide range of software systems, including distributed objects and components, message-oriented communication, and mobile application support. Although new acronyms seem to appear almost every month in the area of middleware, these software systems are based on a few principles and paradigms that have been identified over the years, refined through experience, and embodied in working code. The aim of this book is to contribute to the elaboration and transmission of this body of knowledge for the benefit of designers and developers of future middleware systems. We believe that design patterns and software frameworks are adequate vehicles for achieving this goal.

This book approaches middleware systems from an architectural point of view. Architecture is the art of organizing parts into a whole to fulfill a function, in the presence of constraints. Architects and city planners [Alexander et al. 1977] have introduced the notion of a *design pattern* to describe, in an articulated way, the structures, the representations and the techniques used to respond to specific requirements in a given context. This notion has been adopted by software architects [Gamma et al. 1994], and is now central in software design. As experience builds up, new patterns are elaborated, and an expanding body of literature is devoted to the presentation and discussion of patterns for a variety of situations (see e.g., [Buschmann et al. 1995], [Schmidt et al. 2000] for a review of patterns applicable to distributed systems).

Design patterns allow software architects to reuse proven designs. Likewise, *software frameworks* [Johnson 1997] allow software developers to reuse working code. A software framework is a program skeleton that may be directly reused, or adapted according to well-defined rules, to solve a family of related problems. A framework usually implements a design pattern, and often uses several patterns in combination. Although the notion of a software framework is language-independent, it has mostly been used with object-oriented languages. In this context, a framework is a set of classes that may be adapted for specific environments and constraints, using well-defined rules of usage (e.g., overloading specified methods, etc.). As emphasized in [Schmidt and Buschmann 2003], patterns, frameworks, and middleware play a complementary role for improving the process of designing, building and documenting the increasingly complex applications of today.

The organization of this book is directed by architectural considerations, not by categories of middleware. For each function that a middleware system should fulfill, we summarize the main design issues and we present possible solutions in terms of design patterns. Some of these patterns are well established, and widely documented in the literature; others are more recent and are only described in research papers.

While the emphasis of this book is on design principles and paradigms of middleware systems, we think it important that these principles and paradigms be illustrated by real life examples of working code. We believe in the tenet of literate programming [Knuth 1992]: programs should be intended for reading as well as for execution. Therefore each chapter contains a discussion of a software framework related to the problem under study. These frameworks are mostly borrowed from software developed by ObjectWeb, a consortium of academic and industrial partners dedicated to the development of open source middleware. The actual source code, together with its documentation and prepared examples, is available from the ObjectWeb[1] site.

Although this book touches on various aspects of distributed systems, it is not intended as an introduction to distributed systems principles. These are covered in several textbooks such as [Coulouris et al. 2005, Tanenbaum and van Steen 2006, Veríssimo and Rodrigues 2001].

This book addresses two audiences: students of last-year undergraduate and introductory postgraduate courses, and designers and developers of middleware systems. For the students, this book may complement a course on distributed systems or a course on software engineering. The prerequisites are a basic knowledge of networking and distributed systems principles, and some practice in programming distributed applications. Familiarity with the Java language is required to follow the examples.

Acknowledgments I am indebted to my colleagues, former colleagues, and students of Project Sardes[2] for many fruitful exchanges on the subject matter of this book. Interaction with colleagues and participants of the ICAR[3] series of summer schools also provided motivation and feedback.

References

[Alexander et al. 1977] Alexander, C., Ishikawa, S., and Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press. 1216 pp.

[Buschmann et al. 1995] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1995). *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons. 467 pp.

[Coulouris et al. 2005] Coulouris, G., Dollimore, J., and Kindberg, T. (2005). *Distributed Systems - Concepts and Design*. Addison-Wesley, 4th edition. 928 pp.

[Gamma et al. 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley. 416 pp.

[1] http://www.objectweb.org/
[2] http://sardes.inrialpes.fr/
[3] http://sardes.inrialpes.fr/past-events/summer-schools/past-summer-schools.html#schools

[Johnson 1997] Johnson, R. E. (1997). Frameworks=(Components+Patterns): How frameworks compare to other object-oriented reuse techniques. *Communications of the ACM*, 40(10):39–42.

[Knuth 1992] Knuth, D. E. (1992). *Literate Programming*. Center for the Study of Language and Information, Stanford University - Lecture Notes, No 27. 368 pp.

[Schmidt and Buschmann 2003] Schmidt, D. C. and Buschmann, F. (2003). Patterns, frameworks, and middleware: Their synergistic relationships. In *25th International Conference on Software Engineering*, pages 694–704, Portland, Oregon.

[Schmidt et al. 2000] Schmidt, D. C., Stal, M., Rohnert, H., and Buschmann, F. (2000). *Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects*. John Wiley & Sons. 666 pp.

[Tanenbaum and van Steen 2006] Tanenbaum, A. S. and van Steen, M. (2006). *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2nd edition. 686 pp.

[Veríssimo and Rodrigues 2001] Veríssimo, P. and Rodrigues, L. (2001). *Distributed Systems for Systems Architects*. Kluwer Academic Publishers. 623 pp.