PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A Pattern Language for Distributed Computing

Volume 4

Frank Buschmann,
Siemens, Munich, Germany

Kevlin Henney,
Curbralan, Bristol, UK

Douglas C. Schmidt,
Vanderbilt University, Tennessee, USA

John Wiley & Sons, Ltd

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A Pattern Language for Distributed Computing

Volume 4

Frank Buschmann,
Siemens, Munich, Germany

Kevlin Henney,
Curbralan, Bristol, UK

Douglas C. Schmidt,
Vanderbilt University, Tennessee, USA

John Wiley & Sons, Ltd

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk
Visit our Home Page on www.wileyeurope.com or www.wiley.com

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, Ontario, L5R 4J3, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN-13: 978-0-470-05902-9 (hbk)

Typeset in 10/13 Bookman-Light by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Bell & Bain, Glasgow
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

For Anna, Bebé[†]*, and Martina*

Frank Buschmann

For Carolyn, Stefan, and Yannick

Kevlin Henney

For Lori, Bronson, Mom, and Dad

Douglas C. Schmidt

† Bebé, July 3, 1999

Table of Contents

Foreword

The patterns movement has been around for over a decade now, and has gone through the usual cycle of inflated expectations, backlash, and quiet acceptance. Frank, Doug, and Kevlin have been there the whole time, lauded and scoffed at, but above all quietly collecting good ideas from the field and describing them. The POSA series of books is rightly regarded as one of the most solid elements in the patterns literature, and every volume has a space in my library.

Earlier POSA volumes were traditional patterns books, describing patterns in a range of specific areas, mostly with patterns that hadn't been written up before. This book is different. Distributed Computing is a very wide topic and even the patterns we've captured so far is far more than would fit in a single volume. Indeed they are spread over multiple books, both within and outside the POSA series. This book's mission is to pull these patterns together. As a result you've got many more patterns here than you'd usually find, and consequentially a much terser description. Some of the patterns described here aren't primarily about distribution, but have some relevance for distributed system work. As a result the descriptions in this book highlight that usage, summarizing a pattern in a distributed systems context.

This book is also about more than the individual patterns—it's also about how they relate. Any system contains multiple patterns used together, but I for one find it harder to talk about inter-relationships than the individual patterns. A book like this cannot dodge this question, so here you'll find a lot of advice on how to combine patterns with distribution.

Distribution is a hard problem and often causes trouble. Indeed I'm often quoted for my tongue-in-cheek First Law of Distributed Object Design: 'Don't distribute your objects.' I wrote my first law for a good reason—distribution makes software harder, and as a result I always recommend avoiding it when you can. But however great my desire to question every distribution boundary, the reality is that distribution is an essential part of many software systems. And since distribution is hard, it's particularly important to take care over its design—which is why this book is also an important addition to a developer's library.

Martin Fowler

About This Book

Distributed computing is connecting the world and leveling playing fields [Fri06]. The ubiquity of the Web and e-commerce today exemplify a common motivation for distributed computing: the ability to connect to and access vast quantities of geographically dispersed information and services. The popularity of instant messaging and chat rooms on the Internet underscores another common motivation: staying connected to family, friends, colleagues, and customers. Other motivators for distributed computing include enhancing performance, scalability, and fault tolerance, as well as reducing costs by sharing expensive hardware and peripherals.

Given the importance of distributed computing in our professional and personal lives, many patterns in the software literature focus on this topic [POSA1] [POSA2] [POSA3] [Lea99] [VSW02] [VKZ04] [HoWo03] [PLoPD1] [PLoPD2] [PLoPD3] [PLoPD4] [PLoPD5]. Unfortunately, many of these patterns are described in relative isolation, referencing few other patterns, most of which are in the same publication. Despite the utility of each individual pattern, there is no holistic view of distributed computing that emphasizes how relevant patterns complete and complement each other. Building complex distributed systems therefore remains largely a dark art mastered only by a few wizards and gurus.

To provide a more holistic view, this book—the fourth volume of the *Pattern-Oriented Software Architecture* (POSA) series—describes a single pattern language that links many patterns relevant to distributed computing. Each pattern in this language either deals directly with distributed computing, or plays an important supporting role in that context. Our pattern language thus provides a guide to—and a communication vehicle for—the best practices in key areas of distributed computing.

Intended Audience

Our focus is on the design and implementation of software for distributed computing systems. The main audience for this book is therefore professional software architects or advanced students who are involved in developing software for distributed computing systems, both designing new applications and improving and refactoring existing ones. Our pattern language presents a rich set of patterns aimed at helping architects to create sustainable designs for distributed systems, and which address their requirements thoughtfully and professionally.

A secondary audience for this book is application developers who use component and communication middleware in their professional work. Our pattern language provides developers with an overview of the current state-of-the-practice in designing distributed systems, so that they can better understand how to use middleware effectively. A third group who can benefit from our pattern language is project and product managers. The language can give managers a deeper understanding of the essential capabilities of systems whose development they are leading, and provide a useful vocabulary for communicating with software architects and developers.

We do not however intend end-users or customers to use our pattern language directly. While judicious use of real-world metaphors might make the material accessible to this audience, it would require an alternative presentation of the language. Moreover, the book is not intended as a general tutorial on distributed computing. Although we discuss many aspects of this subject, and include an extensive glossary, readers need prior familiarity with core distributed computing concepts and mechanisms such as deadlock, transactions, synchronization, scheduling, and consensus. Additional information on topics related to distributed computing, such as the design of networking protocols and operating systems, can be found in the references.

Structure and Content

This book is arranged in three parts: some concepts, a story, and the pattern language itself.

Part I, *Some Concepts*, introduces the context of the book: the core pattern concepts necessary for an understanding of the book, an overview of the benefits and challenges of distributed computing, a summary of technologies for supporting distribution, and an introduction to our pattern language.

Part II, *A Story*, describes how a real-world process control system for warehouse management was designed using our pattern language for distributed computing. The story focuses on three areas of this software system: its baseline architecture, its communication middleware, and its warehouse topology representation.

Part III, *The Language*, forms the main part of the book. It contains a pattern language for distributed computing that addresses the following technical topics relevant to the construction of distributed systems:

- Specifying an initial software baseline architecture
- Understanding communication middleware
- Event demultiplexing and dispatching
- Interface partitioning
- Component partitioning
- Application control
- Concurrency
- Synchronization
- Object interaction
- Adaptation and extension
- Modal behavior
- Resource management
- Database access.

Each chapter introduces the topic area it addresses, summarizes key challenges, and then presents a set of patterns that help master these challenges. In total, our pattern language for distributed computing contains 114 patterns and connects to more than 150 patterns presented in other publications. It is thus one of the largest—if not *the* largest—software pattern language documented to date.

Although distributed computing is the language's focus, many parts of it have broader applicability. For example, most applications must be adaptable and extensible in some way, and each software system needs well-designed interfaces and components. For selected technical areas, our pattern language can therefore serve as a general guide to the best practices in modern software development, and is therefore not limited to distributed computing.

The book ends with a short reflection on our pattern language for distributed computing, a glossary of frequently used terms, an extensive list of references to work in the field, a pattern index, a general subject index, and an index that lists everyone who helped us shape the book.

There are undoubtedly properties and patterns of distributed systems that we have omitted, or which will emerge over time through the application and extension of the pattern language in practice. If you have comments, constructive criticism, or suggestions for improving the style and content of this book, please send them to us via electronic mail to `siemens-patterns@cs.uiuc.edu`. Guidelines for subscription can be found on the patterns home page at `http://hillside.net/patterns/`. This link also provides an important source of information on many aspects of patterns, such as available and forthcoming books, conferences on patterns, papers on patterns, and so on.

Acknowledgments

It is a pleasure for us to thank the many people who supported us in creating this book, either by sharing their knowledge with us or by reviewing earlier drafts of its various parts.

Champion review honors go to Michael Kircher, our shepherd, who reviewed all our material in depth, focusing on its correctness, completeness, consistency, and quality. Michael's feedback significantly increased the quality of the material in this book.

In addition, we presented parts of the language at three EuroPLoP pattern conferences, and also to several distribution and pattern experts. Ademar Aguimar, Steve Berczuk, Alan O'Callaghan, Ekatarina Chtcherbina, Jens Coldewey, Richard Gabriel, Ian Graham, Prashant Jain, Nora Koch, Doug Lea, Klaus Marquardt, Andrey Nechypurenko, Kristian Sørensen, James Siddle, Michael Stal, Steve Vinoski, Markus Völter, Oliver Vogel, and Uwe Zdun provided us with extensive feedback, which led to many minor—and also some major—revisions of the language and its presentation.

Many thanks go to Mai Skou Nielsen, who took the photos of Kevlin and Frank when they met at the JAOO 2006 conference in Aarhus, Denmark. Anton Brøgger helped locate details about the photo we present in the chapter on interface partitioning patterns. Publicis Kommunikationsagentur GmbH and Lutz Buschmann permitted us to use photos from their collections in this book.

Special thanks go to Lothar Borrmann and Reinhold Achatz for their managerial support and backing at the software engineering labs of Corporate Technology of Siemens AG, Munich, Germany.

Very special thanks go to our editor, Sally Tickner, our former editor Gaynor Redvers-Mutton, and everyone else at John Wiley & Sons who made it possible to publish this book. It was Gaynor who convinced us to write this POSA volume despite heavy loads in our daily work as software professionals. Sally, in turn, had an enormous amount of patience with us during the years we spent completing the manuscript. Very special thanks also go to Steve Rickaby, of

WordMongers Ltd, our copy editor, for enhancing our written material. This is the fourth POSA book fostered by Steve, and we look forward to working with him on forthcoming volumes.

Last but not least, we thank our families for their patience and support during the writing of this book!

About The Authors

Frank Buschmann

Frank Buschmann is Senior Principal Engineer at Siemens Corporate Technology in Munich, Germany. His research interests include object technology, software architecture, product lines, model-driven software development, and patterns. He has published widely in all these areas, most visibly in his co-authorship of the first two POSA volumes [POSA1] [POSA2], and the last two POSA volumes, this book and [POSA5]. Frank was a member of the ANSI C++ standardization committee X3J16 from 1992 to 1996, initiated the first EuroPLoP conference in 1996, co-edited several books on patterns [PLoPD3] [SFHBS06], and serves as an editor of the Wiley Series in Software Design Patterns. In his development work at Siemens, Frank has led architecture and implementation efforts for several large-scale industrial software projects, including business information, industrial automation, and telecommunication systems.

When not at work Frank spends most of his time enjoying life with his wife Martina and daughter Anna, having fun riding his horse Eddi, watching the time go by in Munich beer gardens, getting excited when watching his favorite soccer team Borussia Dortmund, dreaming when listening to a performance at the Munich opera, and relaxing with rare Scotch single malts before bedtime.

Kevlin Henney

Kevlin Henney is an independent consultant based in Bristol, UK. His work involves teaching, mentoring, and practicing across his areas of interest, which include programming languages and techniques, software architecture, patterns, and agile development. His clients

range from global firms to smaller start-ups that are involved in the worlds of systems software, telecommunications, embedded systems, middleware development, business information, and finance.

Kevlin is a regular speaker at software conferences, and has also been involved with the organization of many conferences, including EuroPLoP. He has been involved with the C++ standard through the BSI and ISO, as well other language standardization efforts. Kevlin is also known for his writing, which has included conference papers and regular (and irregular) columns for many publications, including *C++ Report*, *C/C++ Users Journal*, *Java Report*, *JavaSpektrum*, *Application Development Advisor*, *The Register*, *EXE*, and *Overload*.

In what passes for spare time, Kevlin enjoys spending time with Carolyn, his wife, and Stefan and Yannick, their two sons. This takes in Lego, toy fixing, reading, and the odd beer or glass of wine.

Douglas C. Schmidt

Doug Schmidt is a Professor of Computer Science and Associate Chair of the Computer Science and Engineering program at Vanderbilt University, Nashville, Tennessee, USA. His research focuses on patterns and pattern languages, optimization principles, and empirical analysis of techniques that facilitate the development of quality of service (QoS)-enabled component middleware and model-driven engineering tools that support distributed real-time and embedded systems.

Doug is an internationally recognized expert on patterns, object-oriented frameworks, real-time middleware, modeling tools, and open-source development. He has published over 300 papers in top technical journals and conferences, has co-authored books on patterns [POSA2] and C++ network programming [SH02] [SH03], and has also co-edited several popular books on patterns [PLoPD1] and frameworks [FJS99a] [FJS99b]. In addition to his academic research, Doug has led the development of ACE, TAO, CIAO, and CoSMIC, which are widely used open-source middleware frameworks and model-driven

engineering tools that contain a rich set of reusable components, implemented using the patterns presented in this book.

In his rare spare time, Doug enjoys spending time with his wife Lori and their son Bronson, as well as weight-lifting, guitar playing, debating world history and politics, and driving Chevy Corvettes.

Guide To The Reader

You can have it all. You just can't have it all at once.

Oprah Winfrey

This book is structured so that you can read it in various ways. The most straightforward way is to read it from cover-to-cover. If you know where you want to go, however, you can choose your own route through the book. In this case, the following hints can help you decide which topics to focus on and the order in which to read them.

Introduction to Patterns and Pattern Languages

This book presents a distributed computing *pattern language*, which is a family of interrelated patterns that define a process for systematically resolving problems that arise when developing software for distributed systems. We designed the book to help you use these patterns in your daily software development activities, to create working, sustainable software architectures for distributed systems. It is not a comprehensive tutorial *about* patterns and pattern languages in general, however, since we assume that you are familiar with both concepts.

If this book is your initial exposure to patterns, we suggest you first read the introduction to patterns in *A System of Patterns* [POSA1] and *Design Patterns* [GoF95]. Both books explore the fundamental concepts and terminology related to patterns for software architectures and designs. If you are familiar with patterns, but not with pattern languages, we recommend you read Chapter 1, *On Patterns and Pattern Languages*, and the white paper on *Software Patterns* by James O. Coplien [Cope96], which outline the concept of pattern languages in enough detail to allow you to benefit from the distributed computing pattern language this book. Both the above briefly also explore advanced aspects of the pattern concept that go beyond the fundamental ideas presented in [POSA1] and [GoF95].

Introduction to Distributed Computing

This book assumes that you are familiar with the key concepts and mechanisms of distributed computing. Chapter 2, *On Distributed Systems*, describes briefly the benefits and challenges of distributed computing and summarizes technologies for supporting distribution, but does not discuss distributed computing and distributed systems in detail. The chapter is intended to provide the overall theme of the book: to achieve the benefits of distributed computing, you must explicitly and thoughtfully address the challenges associated with it, guided by patterns in our language.

If you need more background information on distributed computing, we recommend *Distributed Systems: Principles and Paradigms* by Andrew S. Tanenbaum and Maarten van Steen [TaSte02] and *Reliable Distributed Systems* by Ken Birman [Bir05].

Introduction to the Pattern Language for Distributed Computing

Before you start reading all or selected patterns in our pattern language, we suggest you read Chapter 3, *On the Pattern Language*. This chapter introduces you to our language as a whole, focusing on:

- Its intent, scope, and audience.
- The general structure of the language, the key topics and challenges in distributed computing it addresses, and the concrete patterns it contains.
- The pattern form and notation we use to describe and illustrate the patterns in the language.

The chapter also serves as a general map to the pattern language, so that you will know where you are when reading a particular pattern or set of patterns. This map helps to keep you from losing the forest for the trees when reading specific details on each pattern.

The Pattern Language in Action

Part II of this book, *A Story*, presents a concrete example of how our pattern language for distributed computing can be applied in practice on a warehouse management process control system. Through the story of the construction of a real-world system we illustrate how our pattern language for distributed computing can inform the architectures and developers of high-quality software systems. If you learn best by example, we recommend you to read the story before you read the pattern language in depth, although the story also works

well when read after the language. The story demonstrates how our pattern language can support the creation and understanding of:

- *Baseline architectures* for distributed systems that effectively partition their functional and infrastructure responsibilities, and enable the systems to meet their quality of service requirements.
- *Communication middleware* that allows the components of a distributed system to interact with one another efficiently, robustly, and portably.
- The detailed design of concrete components in a distributed system that support their assigned responsibilities and meet their requirements.

Although the story is self-contained, the best way to digest it is to read a specific section until the fundamental solution statement for the problem addressed in that section is described. At this point, we recommend you read the pattern synopsis in Part III if you are not familiar with it. Once you are comfortable with your understanding of the pattern, continue reading the story to see how the chosen pattern is applied in the warehouse management system, and consider which alternative patterns were not selected, and why.

The Pattern Language in Detail

Part III of the book, *The Language*, contains the pattern language for distributed computing that addresses key technical topics relevant to the construction of distributed systems. Here are some ways to read this material:

- *Start-to-finish*. The technical topics covered by the language and the patterns that address them are (roughly) presented in their order of relevance and application when building distributed systems.
- *Topic-wise*. If you are interested in a specific technical topic, such as the partitioning of components, you can read the corresponding chapter of the language. An introduction lists and discusses the challenges associated with the technical concerns, introduces the patterns that help master these challenges, and contrasts and

compares them regarding their commonalities and differences. The condensed summaries of the patterns themselves, the main part of the chapter, follows this introduction.

- *Pattern-wise*. Finally, if you are interested in a specific pattern, you can use the inner front cover or the index to locate it in the language and read it directly.

The pattern summaries do not address detailed implementation issues, such as how a pattern is realized in a specific programming language or on a specific middleware platform. Each pattern summary presents and discusses the essential problem and the forces the pattern addresses, the key solution it embodies, and the consequences it introduces. In addition, each pattern is linked with all other patterns of the language that help implement it, as well as with other patterns whose implementation it can support. Throughout this book, where a pattern from the language is mentioned, it is followed by its page reference in parentheses. If you are interested in specific realization details, we recommend that you consult the pattern's original source(s), which we reference for each pattern. Throughout this book, where a pattern from the language is mentioned, it is followed by its page reference in parentheses.

Patterns described in this book

From Mud To Structure: DOMAIN MODEL (182), LAYERS (185), MODEL-VIEW-CONTROLLER (188), PRESENTATION-ABSTRACTION-CONTROL (191), MICROKERNEL (194), REFLECTION (197), PIPES AND FILTERS (200), SHARED REPOSITORY (202), BLACKBOARD (205), and DOMAIN OBJECT (208).

Distribution Infrastructure: MESSAGING (221), MESSAGE CHANNEL (224), MESSAGE ENDPOINT (227), MESSAGE TRANSLATOR (229), MESSAGE ROUTER (231), BROKER (237), CLIENT PROXY (240), REQUESTOR (242), INVOKER (244), CLIENT REQUEST HANDLER (246), SERVER REQUEST HANDLER (249), and PUBLISHER-SUBSCRIBER (234).

Event Demultiplexing and Dispatching: REACTOR (259), PROACTOR (262), ACCEPTOR-CONNECTOR (265), and ASYNCHRONOUS COMPLETION TOKEN (268).

Interface Partitioning: EXPLICIT INTERFACE (281), EXTENSION INTERFACE (284), INTROSPECTIVE INTERFACE (286), DYNAMIC INVOCATION INTERFACE (288), PROXY (290), BUSINESS DELEGATE (292), FACADE (294), COMBINED METHOD (296), ITERATOR (298), ENUMERATION METHOD (300), and BATCH METHOD (302).

Component Partitioning: ENCAPSULATED IMPLEMENTATION (313), WHOLE-PART (317), COMPOSITE (319), MASTER-SLAVE (321), HALF-OBJECT PLUS PROTOCOL (324), and REPLICATED COMPONENT GROUP (326).

Application Control: PAGE CONTROLLER (337), FRONT CONTROLLER (339), APPLICATION CONTROLLER (341), COMMAND PROCESSOR (343), TEMPLATE VIEW (345), TRANSFORM VIEW (347), FIREWALL PROXY (349), and AUTHORIZATION (351).

Concurrency: HALF-SYNC/HALF-ASYNC (359), LEADER/FOLLOWERS (362), ACTIVE OBJECT (365), MONITOR OBJECT (368).

Synchronization: GUARDED SUSPENSION (380), FUTURE (382), THREAD-SAFE INTERFACE (384), DOUBLE-CHECKED LOCKING (386), STRATEGIZED LOCKING (388), SCOPED LOCKING (390), THREAD-SPECIFIC STORAGE (392), COPIED VALUE (394), and IMMUTABLE VALUE (396).

Object Interaction: OBSERVER (405), DOUBLE DISPATCH (408), MEDIATOR (410), MEMENTO (414), CONTEXT OBJECT (416), DATA TRANSFER OBJECT (418), COMMAND (412), and MESSAGE (420).

Adaptation and Extension: BRIDGE (436), OBJECT ADAPTER (438), INTERCEPTOR (444), CHAIN OF RESPONSIBILITY (440), INTERPRETER (442), VISITOR (447), DECORATOR (449), TEMPLATE METHOD (453), STRATEGY (455), NULL OBJECT (457), WRAPPER FACADE (459), EXECUTE-AROUND OBJECT (451), and DECLARATIVE COMPONENT CONFIGURATION (461).

Object Behavior: OBJECTS FOR STATES (467), METHODS FOR STATES (469), and COLLECTIONS FOR STATES (471).

Resource Management: OBJECT MANAGER (492), CONTAINER (488), COMPONENT CONFIGURATOR (490), LOOKUP (495), VIRTUAL PROXY (497), LIFECYCLE CALLBACK (499), TASK COORDINATOR (501), RESOURCE POOL (503), RESOURCE CACHE (505), LAZY ACQUISITION (507), EAGER ACQUISITION (509), PARTIAL ACQUISITION (511), ACTIVATOR (513), EVICTOR (515), LEASING (517), AUTOMATED GARBAGE COLLECTION (519), COUNTING HANDLE (522), ABSTRACT FACTORY (525), BUILDER (527), FACTORY METHOD (529), and DISPOSAL METHOD (531).

Database Access: DATABASE ACCESS LAYER (538), DATA MAPPER (540), ROW DATA GATEWAY (542), TABLE DATA GATEWAY (544), and ACTIVE RECORD (546).

I Some Concepts

Language is a city to the building of which
every human being brought a stone.

Ralph Waldo Emerson

The first part of this book provides the context for our pattern language for distributed computing. We outline the concepts of patterns and pattern languages briefly, introduce the benefits and challenges of distributed computing, and provide an overview of, and introduction to, the pattern language itself.

This book focuses on patterns and a pattern language for distributed computing. To understand these patterns and the language, and to apply it successfully when building production distributed systems, knowledge of the relevant concepts in patterns and distributed computing, as well as of available distribution technologies, is both helpful and necessary. In addition, using the pattern language effectively in development projects requires you to understand its general scope, structure, content, and presentation.

The first part of the book therefore provides an overview of these concepts, and also provides an overview of our pattern language for distributed computing.

- Chapter 1, *On Patterns and Pattern Languages*, outlines all aspects of the pattern and pattern language concepts that are relevant for understanding our pattern language for distributed computing. We introduce the fundamental concept of patterns, discuss core properties of this concept, and show how patterns can be connected to form *pattern languages*, networks of patterns that work together systematically to address a set of related and interdependent software development concerns.
- Chapter 2, *On Distributed Systems*, provides an overview of the key benefits and challenges of building distributed systems, and outlines which of those challenges are addressed by various generations of distribution technologies, how the technologies address the challenges, and which remain unresolved and must be addressed by the architectures of applications in distributed systems.
- Chapter 3, *On the Pattern Language*, introduces our pattern language for distributed computing. We address the language's intent and scope to define its general applicability. An overview of the thirteen problem areas and 114 patterns illustrates the concrete structure and scope of the language. Information is given about the language's presentation, such as the pattern form and notations used, along with hints about its use to support applications in production development projects.

The three chapters in this part set the context for the entire book: the pattern story about a process control system for warehouse management that we tell in Part II, *A Story*, and the pattern language itself, in Part III, *The Language*.

1 On Patterns and Pattern Languages

Neither can embellishment of language be found
without arrangement and expression of thoughts,
nor can thoughts be made to shine
without the light of language.

Marcus Tullius Cicero, Roman stateman, orator,
and philosopher, 106–43 BC

In this chapter we introduce patterns briefly, including their history, along with a number of pattern concepts. We examine the anatomy of a pattern, what it offers, and what drives it. We explore the relationships we often find between patterns. We conclude with a discussion of pattern languages, what they are, and how they can be presented and used.

1.1 Patterns Introduced

From a design perspective, software is often thought of in terms of its parts: functions, source files, modules, objects, methods, classes, packages, libraries, components, services, subsystems, and so on. These all represent valid views of the different kinds and scales of units of composition with which developers work directly. These views focus on the parts, however, and de-emphasize the broader relationships and the reasoning that make a design what it is. In contrast, patterns have become a popular and complementary way of describing and evolving software designs, capturing and naming proven and common techniques. They emphasize the *why*, *where*, and *how* of designs, not just the *what*.

A pattern documents a recurring problem–solution pairing within a given context. A pattern, however, is more than either just the problem or just the solution structure: it includes both the problem and the solution, along with the rationale that binds them together. A problem is considered with respect to conflicting forces, detailing why the problem *is* a problem. A proposed solution is described in terms of its structure, and includes a clear presentation of the consequences—both benefits and liabilities—of applying the solution.

The recurrence of patterns is important—hence the term *pattern*—as is the empirical support for their designation as patterns. Capturing the commonality that exists in designs found in different applications allows developers to take advantage of knowledge they already possess, applying familiar techniques in unfamiliar applications. Of course, by any other name this is simply 'experience.' What takes patterns beyond personal experience is that patterns are named and documented, intended for distilling, communicating, and sharing architectural knowledge.

From Building Architecture to Software Architecture

Although patterns are now popular and relatively widespread in the world of software development, they originated in the physical world of building rather than the virtual world of software. Throughout the 1960s and 1970s the architect Christopher Alexander and his colleagues identified the concept of patterns for capturing