PART I

INTRODUCTION

In Part I we review design patterns and lay the groundwork for the Pattern-Oriented Analysis and Design (POAD) methodology. Chapter 1 introduces the purpose of the POAD methodology: why it is needed and how it is useful. Chapter 2 discusses the role that design patterns play in software engineering in general and object-oriented design in particular. Most of the experiences in using patterns follow an ad hoc approach for pattern instantiation and composition. Systematic development using patterns utilizes a composition mechanism to glue patterns together at the design level. Generally, we categorize composition mechanisms as behavioral and structural composition. We discuss some well-known behavior-based and structure-based pattern composition approaches in Chapter 3.

1
Pattern-Oriented Analysis and Design

THE ROLE OF PATTERNS IN SOFTWARE DEVELOPMENT

As the complexity of software systems increases, we look for approaches to facilitate the development of software applications. Design patterns [Gamma et al. 1995; Buschmann et al. 1996] and design frameworks [Johnson & Foote 1988; Fayad & Schmidt 1997; Pree 1996; Fayad & Schmidt 1999] are among these promising approaches. Design patterns promise reuse benefits early in the development lifecycle. To reap the benefits of deploying these proven design solutions, we need to define design composition techniques to construct applications using patterns. Versatile design models should be developed to support these techniques.

Reusing software in practical applications is a difficult task, yet it is essential to reduce the development effort and assure higher software quality. Design patterns help in leveraging reuse to the design phase because they provide a common vocabulary for design, they provide a means of understanding designs, and they are proven building blocks from which more complex applications are built. The collection of widely available pattern catalogues stimulates further thinking on how to use these trusted solutions to develop applications. Experienced designers and researchers have expended much effort in documenting good quality practices in software design as design patterns. Whereas a lot of attention is given to finding and documenting design patterns, techniques to deploy and glue these proven design solutions are still lacking systematic support.

Designing applications by systematically deploying design patterns is not a trivial process. Although approaches for design using pattern composition techniques have been proposed, they fall short of the goal of having a systematic process. The purpose of this book is exactly to achieve this goal.

PURPOSE OF POAD

As the demand on software systems increases, researchers as well as practitioners look for development methodologies and techniques to automate the production of software and facilitate its maintainability. These techniques have recently embodied design patterns and frameworks. In particular, we recognize the need for a development methodology to

develop large-scale complex systems and, at the same time, learn from the experiences of other systems designers in solving recurring design problems.

The documentation of design patterns, as it stands, describes details about a pattern, its usage, structure, behavior of participants, forces and consequences, and guidelines for implementation. What we perceive missing is how to compose these patterns together to develop applications. A complete system cannot, nor will ever, be built from a single pattern. It is the integration and composition of patterns that makes a whole system.

We can compose patterns together at the class level or the object level. Class models expose the implementation and maintenance aspects of a pattern, while object models expose the runtime, behavioral, and role aspects. Several researchers and practitioners [Reenskaug 1996; Riehle 1997] address the problem of gluing patterns using role and responsibility modeling (behavioral composition). Less attention is given to the problem of composing patterns as a composition of classes (structural composition). This book is an advancement of the techniques in the structural composition category. We believe that patterns are not *only* about behaviors or roles; their structure aspects are the ones that we implement using traditional object-oriented programming languages. Though it is easier to understand the relationship between objects using behavior models, it is easier to implement that behavior if we have a structure model that captures the classes and their relationships. It is also more likely that developers and designers will continue to use OO programming languages such as Java and C++, and it is less likely that they will use other nonstandard programming construct. Therefore, a pattern composition approach that uses structure models, which have one-to-one mappings to program constructs such as class models, is needed.

The purpose of Pattern-Oriented Analysis and Design (POAD) is to

- *Promote pattern-based development*. We are looking for ways to get more designers to use patterns. We want to attract novice designers and help them use patterns by providing simplified approaches and methods that they can follow to use patterns in their design process. To promote pattern-based development, we need to define composition approaches that are easy to use.
- *Develop systematic approaches to glue patterns*. There is an increasing need to develop systematic composition approaches that facilitate the process of gluing patterns. Models that facilitate the integration of design patterns at the design level should be developed to support these approaches.
- *Develop design frameworks*. We can facilitate the development of design frameworks by using patterns as design building blocks.
- *Improve design quality.* Design patterns are good quality designs. Reusing patterns in a design is anticipated to improve the design quality of software applications built using patterns as their basic building blocks.

PATTERN-ORIENTED DESIGN ISSUES

In promoting pattern-based development and creating new approaches to compose patterns, we are confronted with many challenges:

- *What qualifies a pattern as a design component?* To use patterns as building blocks, we need to find the characteristics that qualify a pattern as a design component.[1] How can we define pattern interfaces for the purpose of integration with other patterns?
- *Can we compose applications solely from design patterns?* Many applications use one or more patterns in their design. The challenge is whether applications can be built by gluing design patterns? How can these patterns interface? What are pattern interfaces, and what interface mismatch issues could arise? Given the current literature of design patterns, is this repository sufficient to design applications using design patterns? What type of patterns can be used?
- *How can we systematically develop applications using design patterns?* Is there a well-defined design process that can be followed to develop applications using design patterns as their building blocks?

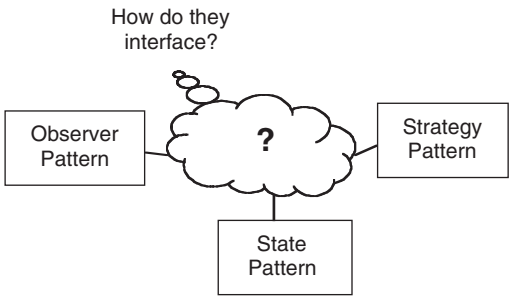

FIGURE 1–1 Can we compose application designs using design patterns?

POAD IS A SOLUTION

We developed POAD to address the above problems. In POAD, patterns are stringed at the high-level design, and their constituents are merged in the subsequent design stages to obtain a dense and profound design. The POAD approach provides the following solutions:

1. *Modeling patterns as design components.* Since patterns will be the building blocks of pattern-oriented designs, they should first possess the characteristic of a component, mainly the ability to be composed at the design level. This necessitates the definition of pattern interfaces and the essential pattern properties to enable a pattern to become a design component. POAD defines a particular type of patterns called constructional design patterns and defines various levels of abstraction and logical views in designing with constructional patterns. POAD also defines how these levels are traceable to lower design levels in terms of classes.
2. *A design methodology.* We discuss a methodology to construct pattern-oriented designs. POAD is an explicit effort to study patterns as the core building blocks of object-oriented (OO) designs. In POAD, we learn from the experiences of OO analysis and design methods and define a new pattern-oriented method that builds

[1] More elaborate discussion about the differences between design components and deployable components will follow in Part II.

upon the Unified Modeling Language (UML) syntax and semantics. We call the designs developed using this methodology: *pattern-oriented designs*.

3. *Real-world applications.* To study the applicability of the POAD technique, we apply the design methodology to four applications and construct a pattern-oriented design for each.

The main objective of POAD is to provide a solution to improve the practice of systematically deploying design patterns in application development. We believe that the problem should be tackled at an early phase of the development lifecycle, namely at the analysis and design levels. The approach we take is to develop the POAD methodology for building OO designs using design patterns as their building blocks.

In a software engineering context, a design methodology has three main dimensions: the *technology*, the *process*, and the *organization* aspects.

Technology aspects. This dimension defines the fundamentals of the design methodology, which includes the concepts, notations, and visual and formal models. We define visual models that are used to structurally glue patterns to develop pattern-oriented designs. For this purpose, we introduce the concept of pattern interfaces and discuss the relation to software components and architecture. We discuss the syntax and semantics support of UML for the POAD methodology.

Process aspects. This dimension defines the tasks and steps essential to develop pattern-oriented designs. Based on the visual models, analysis and design steps are defined. The output and deliverables of each step are also defined. Tool support for POAD is also discussed in the book.

Organizational aspects. This dimension defines how the enterprise is organized to put the methodology in effect.

Part II of the book addresses the technology aspects. In Part III we discuss the process aspects. Organizational aspects are not yet addressed in the POAD methodology.

WHAT IS COVERED IN THIS BOOK?

This book is mainly about the POAD methodology: its models and processes. It is also about the application of the POAD methodology to several case studies. We summarize what is covered in this book as follows:

- A *Pattern-Oriented Analysis and Design* (POAD) software development approach that is based on structural composition of design patterns.
- The concept of *pattern-oriented frameworks* and *pattern-oriented designs*.
- Application of the notion of *design components* and *interfaces* to design patterns and introduction of the concept of *constructional design patterns*.
- Application of the UML extension mechanisms to model composition of design patterns. These extensions facilitate the integration of the POAD approach to existing tools that support UML models.
- Application of the POAD design methodology to four case studies to develop pattern-oriented designs and pattern-oriented frameworks.