Design Patterns for Object-Oriented Software Development

Wolfgang Pree
Johannes Kepler University Linz
Altenbergerstr. 69
A-4040 LINZ, Austria
+43 70 2468 9432
pree@swe.uni-linz.ac.at

Hermann Sikora
RACON Software, Inc.
Goethestr. 80
A-4020 LINZ, Austria
+43 70 6929 213
lrzgsik@grz-linz.raiffeisen.at

ABSTRACT
The tutorial gives an overview of state-of-the-art design patterns approaches, including pattern catalogs and framework patterns. A selection of useful patterns is discussed in detail. The tutorial also introduces socalled hot spot cards, which have proved to be a useful communications vehicle between domain experts and software engineers in order to exploit the potential of design patterns. In the early development phases hot spot cards help to capture those system aspects that have to be kept flexible. Case studies based on Java illustrate how to apply design patterns.

Keywords
Design patterns, software architecture, frameworks, object-oriented design

INTRODUCTION
The terms *object-orientation* and *component technology* have become buzzwords associated with the solution of many problems in the realm of software engineering. Above all, software components are supposed to be better reusable and more flexible compared to conventionally developed software. Unfortunately, the benefits associated with object technology have their price.

Only frameworks, that is, semifinished software architectures, enable the exploitation of the full potential of object-oriented software construction. Frameworks represent the highest level of reusability known today: not only source code and single components, but also architectural design—which we consider the most important characteristic of frameworks—is reused in applications built on top of a framework. Overall, well-designed and thoroughly implemented frameworks enable a degree of software reusability that can significantly improve software quality.

ICSE 97 Boston MA USA

The tutorial first focuses on the concepts underlying frameworks, and goes on to present advanced design heuristics for developing such artefacts: though design patterns are currently heralded as a means to support framework development, we still lack a link between design patterns that capture and communicate proven object-oriented design and the framework development process. As a pragmatic solution to this problem, we introduce a *hot-spot-driven design approach* using *hot spot cards* to bridge this gap. These cards proved to be a valuable communication vehicle between domain experts, whose knowledge is a critical success factor for framework design, and software engineers. They help to reduce the number of framework design iterations. In other words, hot spot cards fill the gap between the early activities in the development process and the design/implementation activities. They let you get the most out of patterns.

Case studies and examples illustrate framework technology and hot-spot-driven design. We use the Unified Booch & Rumbaugh Notation and the Java programming language to present the examples.

FRAMEWORKS & PATTERNS
How can we teach framework development? Usually it is recommended that programmers who want to learn about framework construction should take a close look at various existing frameworks. Unfortunately, only very few frameworks with an excellent design are accessible. Furthermore, exploring a framework is a tedious task. Poor documentation of frameworks makes studying design and implementation details painful and time consuming. Abstracting *design patterns* which have been obscured by many implementation details requires an in-depth look at a framework. Sometimes it becomes even impossible

to understand particular design decisions without any hints.

This is why the roots of the design pattern movement go back to people who tried to describe framework construction on a level higher than the underlying programming language. Erich Gamma pioneered a catalog-like presentation of design patterns in his Ph.D. thesis [3]. There he describes core design aspects of the GUI framework ET++ [6] in an informal way, combining informal text with class and object diagrams. His thesis formed the basis of an enhanced catalog-like presentation of more than 20 design patterns published in the Gang-of-Four (GoF) book [5].

Gamma *et al.* state the difference between frameworks and catalog patterns: ".... frameworks are implemented in a programming language. .... In this sense frameworks are more concrete than design patterns. .... Mature frameworks usually reuse several design patterns." [4]

Meanwhile other authors published patterns in a catalog style, for example, in the realm of the PLoP (Pattern Languages of Programming) conference proceedings [2] or as separate book [1].

These pattern catalogs contain numerous similar entries. They mainly differ in the semantics (method/class names) of the aspects that are kept flexible. Pree [7] coins the term metapattern and discusses the commonalities between catalog entries. The tutorial presents the essential framework construction principles by means of a case study and relates them to catalog entries in the GoF book.

Finally, are patterns just a hype or not? Lewis *et al.* [6] view the pattern movement from the perspective of frameworks as part of the evolution of this technology: "Patterns ... is one of the most recent fads to hit the framework camp. ... Expect more buzzwords to appear on the horizon." Because patterns have become the vogue in the software engineering community, the term now is used wherever possible, adorning even project management or organizational work. So the genericity of the term pattern might be the reason that patterns are found everywhere, a fact which is regarded as a clear indication of a hype.

HOT SPOT CARDS

How can one assess the quality of a framework? Besides the usual software quality attributes such as correctness, ease of use, efficiency, and portability, frameworks have to offer more. Though this quality attribute is related to flexibility, it does not mean that frameworks are better if mere flexibility is maximized. So striving for flexibility for the flexibility's sake, achieved by incorporating as many design patterns as possible, does not result in a good framework. On the contrary, unnecessary flexibility leads to significantly more complexity. Frameworks must be adaptable in an adequate way depending on domain-specific requirements.

Overall, flexibility has to be given into a framework with the right doses. As the quality of a framework depends directly on the appropriateness of hot spots, hot spot identification has to become an explicit activity in the framework development process. Means for documenting and communicating hot spots between domain experts and software engineers become crucial.

The tutorial illustrates how hot spot identification can be communicated by what we call *hot spot cards*. Hot spot cards form the basis for transforming an object model into a domain-specific framework.

Based on flexibility requirements specified as a stack of hot spot cards, software engineers have to transform the object model. In this step the essential framework-centered construction patterns assist the software engineer. The tutorial discusses the relationship between the information captured on hot spot cards and framework construction patterns. Pree [8] describes in detail how to accomplish the transformation.

REFERENCES

1. Buschmann F., Meunier R., Rohnert H., Sommerlad P. and Stal M. (1996) *Pattern-Oriented Software Architecture—A System of Patterns.* Wiley and Sons
2. Coplien J. and Schmidt D. (eds.) (1995). *Pattern Languages of Program Design.* Conference Proceedings. Reading, Massachusetts: Addison-Wesley
3. Gamma E. (1992). Objektorientierte Software-Entwicklung am Beispiel von ET++: Design-Muster, Klassenbibliothek, Werkzeuge. *Doctoral Thesis*, University of Zürich, 1991; published by Springer Verlag, 1992
4. Gamma E., Helm R., Johnson R. and Vlissides J. (1993). Design patterns: abstraction and reuse of object-oriented design. In *Proceedings of the ECOOP'93 Conference*, Kaiserslautern, Germany; published by Springer Verlag
5. Gamma E., Helm R., Johnson R. and Vlissides J. (1995). *Design Patterns—Elements of Reusable Object-Oriented Software.* Reading, Massachusetts: Addison-Wesley
6. Lewis T., Rosenstein L., Pree W., Weinand A., Gamma E., Calder P., Andert G., Vlissides J., Schmucker K. (1996) *Object-Oriented Application Frameworks.* Manning Publications/Prentice Hall
7. Pree W. (1995). *Design Patterns for Object-Oriented Software Development.* Wokingham: Addison-Wesley/ACM Press
8. Pree W. (1996). *Framework Patterns.* New York City: SIGS Books