Lecture Notes in Computer Science 6510

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

James Noble Ralph Johnson
Paris Avgeriou Neil B. Harrison
Uwe Zdun (Eds.)

Transactions on Pattern Languages of Programming II

Special Issue on Applying Patterns

Springer

ISSN 0302-9743 (LNCS) e-ISSN 1611-3349 (LNCS)
ISSN 1869-6015 (TPLOP)
ISBN 978-3-642-19431-3 e-ISBN 978-3-642-19432-0
DOI 10.1007/978-3-642-19432-0
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011921520

CR Subject Classification (1998): D.2, D.3, D.1, K.6, K.6.3

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Even though patterns are a software engineering success story and have become mainstream in recent years, the application of the different types of patterns in practice has not yet fully met the expectations of the patterns community. Among other reasons, this is due to the inefficient advertisement of patterns to the practitioners' community, as well as the lack of techniques and tools for using patterns, and for introducing them into organizations. There are many pattern enthusiasts in the industry but the large body of patterns knowledge is not fully exploited in practice. Many practitioners are aware of the GoF patterns and use them regularly in design and coding, but there is substantially less evidence of the systematic use of patterns in other areas of software, such as software architecture, analysis, or processes. While there are many anecdotes of the benefits of using patterns, the impact of using patterns is not fully understood yet. We note that there is some increase in teaching patterns in higher education and professional training programs, but the results are only slowly becoming visible in industry.

We are interested in studying the application of patterns in practice both from the practitioners' and the researchers' point of view. Practitioners such as experienced architects and designers have applied pattern-based approaches in industrial projects and have shown promising results even if they lack scientific rigor. Researchers have proposed scientific approaches that aim at methods in software engineering or other domains in order to support the systematic application of patterns but may still be at the research stage.

We initiated an effort to gather together practitioners and researchers working on this topic by organizing a special track on the European Conference on Pattern Languages of Programs in the years 2008, 2009, and 2010. The track received ample interest from the patterns community and received a number of high-quality submissions, containing a mix of practical and research approaches. It provided encouraging results and a strong indication that the application of patterns is an important topic for the community at large. The other positive development is that some of the papers discussed in this track have included empirical validation in an attempt to provide evidence about the effectiveness of the proposed approaches. In this special issue we have included five papers which are characteristic of this theme in applying patterns both from researchers and practitioners.

In less than two decades since software patterns were introduced, their effect on software development has been remarkable, particularly with respect to object-oriented design patterns. Object-oriented design patterns have become mainstream: they are widely used, college courses on patterns are common, and design pattern expertise even appears as a desired skill in job position descriptions.

Outside the realm of the OO design patterns, the picture is somewhat cloudier: patterns other than OO design patterns are not as well known. This is partly due to the profound success of the design patterns book. Nonetheless, there are numerous other categories of patterns, and some of them enjoy widespread use in individual organizations or domains (e.g., distributed computing and enterprise applications).

But how much use do patterns have? And how beneficial are they? Unfortunately, the evidence of the utility of patterns is still largely anecdotal. This is even true of the vaunted design patterns: everyone agrees that they are the greatest thing since curly braces, but we aren't sure exactly how much they improve design. This is not exclusively a problem within software patterns; the quality of software designs is notoriously difficult to measure. Furthermore software designers' intuition is indeed worth something: if many designers swear by patterns, they are probably right. However if we aim at convincing organizations about the benefits of applying patterns, we have to follow rigorous approaches and provide hard evidence rather than relying on anecdotes and gut feeling. This is in line with the software engineering community's attempt to turn towards evidence-based software engineering in order to have a more profound impact on the state of practice.

Therefore, there is a need to examine the use of patterns in industrial settings, collect data, and provide evidence to answer questions on how to use patterns as well as their benefits and liabilities. Thus one purpose of this special issue is to explore the usage of patterns in the real world. A related purpose is to continue to spread knowledge of particular patterns so that best practices also become ubiquitous practices.

The articles in this issue all demonstrate techniques for applying patterns in an industrial or research setting. Some have confronted the topic within software engineering; others offer approaches in other pattern domains, which is an indication of the diverse fields where patterns are applied.

In "Lessons Learned from Using Design Patterns in Industry Projects," Dirk Riehle elaborates on his long-term experience in applying design patterns in industrial projects and subsequently affecting the organizational culture. Riehle explains the main benefits that design patterns bring in software development: communication, implementation, and documentation. He then suggests the introduction of a design language that is specific to an organization, which codifies the collective reusable knowledge about patterns extended from design to implementation. Finally he presents specific lessons learnedand guidance for creating and applying such a design language using study groups and writers' workshops. The evidence collected from the author's experience demonstrates benefits for the organizations involved and justifies the investment required.

In "Choose Your Own Architecture – Interactive Pattern Storytelling", Jim Siddle proposes the concept of interactive pattern stories as a way to use patterns in the education of software engineers and especially designers. The idea behind interactive pattern stories is to enrich traditional pattern stories with some interactivity by having the readers make decisions and follow different

paths through the story. The author presents an example interactive pattern story, which uses patterns from the distributed systems domain. The approach enables readers to explore various alternative solutions to design problems and reflect on the solutions' benefits and liabilities. The interactive format used in this approach is engaging and game-like, though it requires substantial effort to write the stories' format. The applicability of the approach is mostly within higher software engineering education or professional training.

In "Experiences in Using Patterns to Support Process Experts in Process Description and Wizard Creation", Birgit Zimmermann, Christoph Rensing, and Ralf Steinmetz propose the use of patterns as a means to codify expert knowledge in adaptation processes and semi-automate the knowledge transfer. Their approach starts with the creation of process patterns for a specific application domain: adapting e-learning material. The knowledge stored in the process patterns through a specific formalism is then used to semi-automatically generate wizards that can eventually guide the end users step-by-step to perform the adaptations. The approach has been empirically validated in an industrial setting with promising results. This is one of the few approaches that consumes the knowledge (about processes) within patterns and semi-automatically creates tooling to support processes.

In "Modifiers: Increasing Richness and Nuance of Design Pattern Languages", Gwendolyn Kolfschoten, Robert O. Briggs, and Stephan Lukosch tackle one of the major challenges in using patterns: the tradeoff between a rich pattern language that provides a plethora of inter-dependent solutions and the cognitive over-load that large pattern languages may cause. Especially novices learning a pattern language may get overwhelmed by the number of choices and the rich dependencies between them. The authors propose to resolve this with the concept of the modifier: a variation that can be applied to a set of other patterns. Modifiers result in similar modifications to the solutions of all the patterns within the set of patterns to which they are applied. The concept of a modifier is exemplified with two case studies: two different pattern languages for facilitation and computer-mediated interaction. The approach has been validated through expert judgment that yielded positive results.

In "Patterns for Effectively Documenting Frameworks", Ademar Aguiar and Gabriel David present best practices for documenting object-oriented frameworks in pattern form. They present six documentation patterns as well as many details on how to apply these patterns in practice. In particular, a pattern-based documentation process covering the major roles and activities of object-oriented framework documentation is presented. In addition to this process, guidelines are given concerning why, by whom, to whom, and when framework documentation should be performed and a detailed pattern sequence is given clearly outlining the relationships of the patterns. Finally, various issues of framework documentation and the types of information to be documented are systematically discussed.

Patterns have always been about capturing best practices in software so that others may benefit from them. Yet for this very reason, the practical use of any pattern predates its discovery. This must be so, because patterns are proven

solutions – they have already been used multiple times before they can be considered patterns. In this sense, people use patterns in two ways: intentionally, based on their knowledge of the patterns, and unknowingly. The ongoing challenge, therefore, is to bring the hitherto unknown patterns to light. And it is a challenge: there are so many patterns that have already been written, along with others to be written, that finding the patterns that are most important for a designer at a particular moment is daunting. We hope that searching the vast landscape of patterns will become more viable through organization of patterns, searching tools, and the continued emergence of domain-specific patterns. We have no illusions that this will be easy. And it is made more difficult by the continued emergence of new patterns, as more and better ways of developing software are devised. But the prospect of these new patterns is indeed a happy thought.

We advocate patterns as a cost-effective way to transfer knowledge, establish communities of practice that share this knowledge, and solve problems based on sound reasoning instead of intuition and trial-and-error attempts. However, the second and even greater challenge to pattern adoption is to accompany the patterns and pattern languages with the appropriate evidence about their effectiveness and the support they can provide to practitioners. Introducing patterns into an organization or into a project is an investment; therefore it has to be justified with the appropriate evidence. Consequently we make a plea to practitioners and researchers alike, to gather such evidence and disseminate it across the community. There are numerous stories about using patterns in the field; it is our responsibility to share them and reinforce our collective knowledge.

December 2010

Paris Avgeriou
Neil B. Harrison
Uwe Zdun

Table of Contents