Christoph Fehling · Frank Leymann
Ralph Retter · Walter Schupeck
Peter Arbitter

Cloud Computing Patterns

Fundamentals to Design, Build, and Manage Cloud Applications

EXTRA MATERIALS
extras.springer.com

Springer

Christoph Fehling · Frank Leymann
Ralph Retter · Walter Schupeck
Peter Arbitter

Cloud Computing Patterns

Fundamentals to Design, Build, and Manage Cloud Applications

Springer

Cloud Computing Patterns

Christoph Fehling • Frank Leymann •
Ralph Retter • Walter Schupeck •
Peter Arbitter

Cloud Computing Patterns

Fundamentals to Design, Build, and Manage Cloud Applications

Springer

Christoph Fehling
University of Stuttgart
Stuttgart
Germany

Frank Leymann
University of Stuttgart
Stuttgart
Germany

Ralph Retter
T-Systems International GmbH
Frankfurt
Germany

Walter Schupeck
Daimler AG
Stuttgart
Germany

Peter Arbitter
Microsoft Deutschland GmbH
Unterschleißheim
Germany

Additional material to this book can be downloaded from http://extra.springer.com.

ISBN 978-3-7091-1567-1 ISBN 978-3-7091-1568-8 (eBook)
DOI 10.1007/978-3-7091-1568-8
Springer Wien Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013955000

Printed on acid-free paper

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

To Carina and my parents, Elisabeth and Horst. Thank you for your love and support.
– Christoph

To the woman of my dreams!
– Frank

To Anni and Emil.
– Ralph

To my grandsons Leonardo and Max Viktor.
– Walter

To Annika, Paulina, Nils, and my lovely wife Martina.
– Peter

Foreword by Gregor Hohpe

Over the recent years, you may have observed a strategy, dare I say "pattern," for software engineering books: take a popular buzzword and append the word "patterns" to suggest that this particular title contains a substantial treatment of the subject matter and is organized as a collection of easy to digest chapters, which promise to provide solutions to recurring problems. For the pattern community, this is a definite sign of success, and not one that is due to large marketing budgets but rather many successful titles that deliver on this very promise. A recent search on the "world's largest bookstore" in the category "Books – Software Development" yielded no less than 499 titles containing the word "patterns." How does one pick out the gems from such a vast collection of solutions to recurring problems?

First, good patterns books present a coherent pattern language, not just a collection of patterns, in line with Christopher Alexander's seminal title *A Pattern Language*. Structuring a pile of patterns into a sequence of chapters is laudable but does not make a language. Rather, a pattern language covers a specific domain and usually offers multiple patterns for common design challenges and offers a clear road map through the patterns.

Many influential software patterns books have also adopted a strong visual language. I may be biased because *Enterprise Integration Patterns* was one of the first titles to expand the notion of a pattern *sketch* to a visually coherent iconic language, but because software architecture and specifically software patterns aim to communicate complex design ideas in an easy-to-grasp way, a visual language is a strong asset.

Almost all significant software patterns books are not simply the result of the authors' deep insight or determination but have been a true community effort. For example, the Pattern Languages of Programming (PLoP) conference, which has been running since 1994, has been the birthplace of many great patterns books. While it is certainly not a requirement for pattern authors to workshop their papers at PLoP, the claim that patterns are harvested rather than invented points to the importance of community participation. This also means that good patterns books don't usually pop out of nowhere but are the pinnacle of a lengthy process of stepwise refinement based on community feedback.

Last but not least, most successful patterns books strike a delicate balance between academic rigor and real-world applications. The academic world brings depth of thinking and a clear structure, while the industry contributes the required validation and real-world examples.

If you are wondering why the foreword to *Cloud Computing Patterns* muses about the pattern movement as a whole, I invite you to revisit after reading this title. I hope you will agree that I was in fact describing the essential properties of *cloud computing patterns*! I have not much left to say but congratulate the authors on the successful delivery of a relevant, well-structured, and community-validated software patterns title. Because I am sure I won't have to convince you that cloud computing is one of the most relevant software engineering topics of the last decade.

– Gregor Hohpe, coauthor of *Enterprise Integration Patterns* (Hohpe, G., Woolf B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley. http://www.eaipatterns.com/ (2003)).

Gregor Hohpe

Foreword by Robert Hanmer

The forecast for computing is full of clouds. The computer press and the public media are abuzz with "cloud-this" and "cloud-that," but what does it all mean? In many industries, service providers are looking to the cloud to help simplify their hardware management and increase their flexibility and adaptability to change and to provide new and innovative services, but much of what's written about clouds is about the nuts and bolts and offerings of specific cloud providers. The big picture of how to build useful and robust applications to support business needs on top of clouds is missing. Specific case studies are useful, but they can be limiting because they also are tied to specifics. What's needed is a resource to understand the ins and outs of clouds and how they can be used from a top-down perspective. This book is that resource.

I first met the authors at the 2011 Pattern Languages of Programming Conference in Portland, OR, USA. They submitted a paper to the conference that was about early work that ultimately led to this book. Because a paper about clouds was timely, it was one of the most popular papers at the conference. Everyone wanted to learn more about clouds, and all were intrigued by the range of cloud solutions studied to understand the patterns.

Patterns in the style of Christopher Alexander and the *Design Patterns* book [2] (aka the Gang of Four - GOF patterns) are based on real-life experiences and proven practices, not unproven theories. The successes, failures, and "aha's" of real-life solutions were captured in these patterns which were mined from projects across several industries. The authors aren't trying to sell clouds and they aren't just patting themselves on the back because they've built something nice with clouds. They explain clouds based on what they've seen work in practice. The topics range from the basics of on-demand self-service, broad network access, pay per use, and rapid elasticity to complex topics like structuring multi-tier applications and how to best handle issues like fault tolerance and performance.

Patterns explain not just what a cloud is or what you must do to effectively use a cloud but also explain the trade-offs involved in using the cloud. The patterns tie the cloud definitions back to the NIST Definitions of Cloud Computing [3] to thoroughly cover the topic. I agree with the authors that the pattern form is a good way to explain clouds, because patterns explain the design decisions and trade-offs you must be aware of to build an effective cloud solution. The authors' experiences help you to know why things are the way they are as well as highlight how to work around or benefit from that rationale.

You'll find seven chapters in this book. Each chapter contains many patterns, which describe a solution to a problem with an explanation of why that solution is the right one. The chapters build from the basic fundamentals of the NIST definition of a cloud all the way to cloud application shapes and sizes. In between, the authors cover the basics of the different kinds of cloud offerings that you'll find and the basics of designing your solution's cloud architecture and how to manage your cloud. The last chapter talks about problems and pitfalls to avoid, for example, the mismatch between the requirements imposed from the bottom up by the cloud providers and the requirements pressing downward from the business analysts.

I especially like the author's treatment of non-functional requirements. They don't ignore the issue of fault tolerance and robustness which would give you a view of only the sunniest days ahead. This book points out that the rapid elasticity of the cloud is not a guarantee of high reliability by itself but that further design principles like elasticity and resiliency management are needed to ensure that the application achieves the required level of reliability and availability.

I think that this book will help us all better utilize the cloud. And it's a good example of using patterns to explain a complex topic. The scope of computing-related patterns is expanded from the range of technical domains like fault-tolerant software and small memory systems to enterprise applications and then all the way towards structuring complete business processes and environments like we see here with cloud computing.

Cloud-this and cloud-that are explained in this book, which will help you prepare for whatever the cloudy future of computing delivers.

– Robert S. Hanmer, author of *Patterns for Fault Tolerant Software* (Hanmer, R.: Patterns for Fault Tolerant Software. Wiley, Chichester (2007)).

Robert Hanmer

Preface

Which is the right cloud computing provider for my company?
How do I use the cloud computing products of a certain provider and how do they compare to the products I already have in place? We use an infrastructure as a service platform already. What is different now?
How does cloud computing impact existing applications if they are moved to the cloud?
How can I use cloud computing to reduce provisioning and setup times, so my users can access servers more quickly?

During the past years of our collaborative research and consulting on cloud computing, we often encountered these questions or very similar ones. Often, these questions were driven by the need to rapidly produce results in big enterprise cloud initiatives that were founded with the ambitious goals to reduce the cost of IT tremendously and make provisioning of IT resources faster. We also encountered a general trend that cloud computing initiatives are often driven *bottom-up*, starting with infrastructure automation activities. These initiatives, therefore, tend to focus on the IT infrastructure and how to change its setup, deployment, and management and leave application architecture standards for the enterprise untouched. Results are company-internal infrastructure-as-a-service offerings that can provision virtual servers in a fast and efficient manner via a self-service portal. In this book, we describe how cloud computing changed IT infrastructure provisioning and management as well, but mainly we focus on the following questions:

How does cloud computing change the architecture of applications using it efficiently? How can I use cloud infrastructure and platform offerings to efficiently and rapidly design, build, and manage applications to support the changing needs of my business?

Many products found in the cloud computing market display similar functionality that customers may use to build their cloud applications. The discussions of IT architects and developers are often hindered by different product names, provider-specific terminologies, the different distributions of common functionality among products, as well as the fact that some products are merely *cloud-washed*, and thus their name is extended with the term “cloud” to make them more attractive. In our research, we, therefore, analyzed cloud products as well as applications built using different cloud vendors and cloud computing technologies. The goal of this analysis has been to extract common behavior and common components of cloud products as well as the common architecture principles involved to build cloud applications. The result is the book you hold in your hands.

The reason for abstracting from existing cloud providers and cloud applications is to capture provider-independent and sustainable knowledge abstracted from concrete products. This abstract description covers how to build cloud applications, how different cloud products behave, and when to choose a particular form of a cloud offering given a concrete usage scenario.

What You Will Learn and What This Book Is About and Not About

In this book, we build upon our experiences and approach cloud computing principles with a *top-down* mind-set. The goal of this book is not to show you the benefits of cloud infrastructure automation technology over traditional data centers but to show you how cloud principles can be supported on the business and application layers. This book should give you nuggets of advice in the form of patterns that allow you to better understand how you can support cloud properties on the application level and how to select appropriate cloud infrastructure and platform offerings to make your life easier. Furthermore, this book gives IT architects and developers a common vocabulary when discussing about cloud products and architectures of custom cloud applications without focusing on a concrete cloud provider.

During our consulting and research experiences, we often missed the abstraction of cloud definitions from concrete provider offerings and vendor products. Thus, we chose to use a pattern-based approach to describe cloud offerings vendor neutrally in our technical report (Fehling, C., Leymann, F., Mietzner, R., Schupeck, W.: A collection of patterns for cloud types, cloud service models, and cloud-based application architectures. Technical report, University of Stuttgart) in 2011. As this approach proved very popular wherever we presented and discussed it, we decided to revise and extend this approach and the resulting pattern catalogue into what you will find in this book. In order to not lose track with reality, we describe the essential properties of a pattern and then add concrete providers and vendor offerings where we are aware of them in “known uses” sections of the pattern. These sections are by no means exhaustive; they should give you an initial idea on providers and vendors offering a product that exhibits the properties described in the pattern.

If you take this book as a catalogue of patterns that will help you to design, build, and manage *cloud-native* applications as well as select suitable cloud infrastructure and platform offerings, it will provide you with the most value. We wish you as much motivation and satisfaction reading about and applying the patterns in the book as we had collecting and presenting them at various conferences and events. For further information, please also refer to:

http://www.cloudcomputingpatterns.org
Germany

Christoph Fehling
Frank Leymann
Ralph Retter
Walter Schupeck
Peter Arbitter

Free. Android. Cloud Computing Patterns is mobile application provides CIOs, software architects, project managers, developers with a set of architectural patterns that offer nuggets of advice on how to achieve common cloud computing-related goals. Cloud Computing Design Patterns and Mechanisms. This resource catalog is published by Arcitura Education in support of the Cloud Certified Professional (CCP) program. These patterns and their associated mechanism definitions were developed for official CCP courses. (Note that this site is still undergoing improvements. Please provide feedback or report issues to info@arcitura.com.) Find cloud computing patterns stock images in HD and millions of other royalty-free stock photos, illustrations and vectors in the Shutterstock collection. Thousands of new, high-quality pictures added every day.Â 72,127 cloud computing patterns stock photos, vectors, and illustrations are available royalty-free. See cloud computing patterns stock video clips. of 722. cloud icon white background.