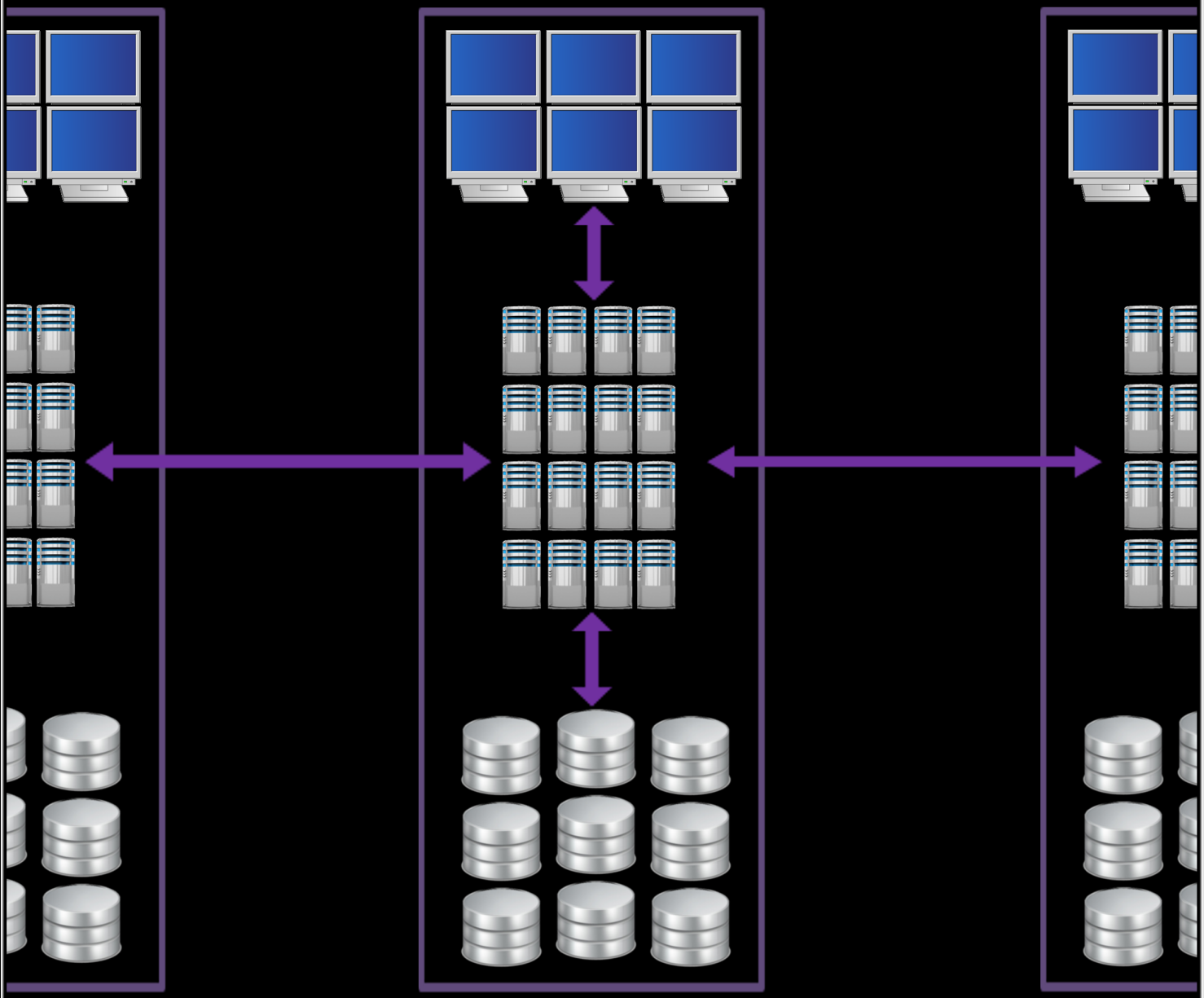
Applied Modularity

Do we really need OSGi,
Micro Services and the Java Module System?



by Reik Oberrath

Applied Modularity

Do we really need OSGi, Micro Services and the Java Module System?

Reik Oberrath

This book is for sale at http://leanpub.com/applied-modularity

This version was published on 2018-12-15

I am very grateful to the IKS GmbH for supporting my efforts with this book. Special thanks goes out to my colleagues Jörg Vollmer (co-founder of the Clean Coding Cosmos), Christoph Schmidt-Casdorff (software architect and co-author of "OSGi Einstieg und Überblick") and Hartwig Tödter (all-rounder with long time IT experience – both in practise and education) for in-depth discussions on the subject and reviewing several drafts of this book. In addtion, I appreciate the testing effort of Fabian Prinz, which contributed much to the quality of the GitHub code that belongs to this book. Finally, I would like to thank Valerie Tenberg whose input made this book far more understandable and readable.

Contents

Preface

In the wide field of software engineering modularity enjoys a good reputation while its counterpart, the monolith (frequently called a “Big Ball of Mud”) conjures a bad one. How far this opinion is justified will be reviewed at the end of this book. However, modularity is an abstract topic and a lot of experience is needed to recognise and understand its benefits as well as its risks and costs. Given the abstract nature of this topic, typical approaches to explain modularity and its application appear to be very academic. This is why I would like to take a more pragmatic view, i.e. comparing different approaches to modularise software.

Building modular software is possible in many different ways, e.g. Service Component Architecture (SCA), the Common Object Request Broker Architecture (CORBA), Open Service Gateway Initiative (OSGi), Service Oriented Architecture (SOA), Micro Services and the Java Module System. I am a software developer who has been active in recent years regarding some of these ways. Thanks to this experience, I have gained some insight as to how differently modularity can be realised in software engineering. With this book I would like to illustrate those ways of modularity where I have gained experience. Since I am a Java developer, this is done with the background of the world of Java. By this, I believe, that terms like classpath, Maven, JDK, runtime, etc. are familiar to you.

The three approaches to modularity mentioned in the title of this book are different in many aspects, but similar in a specific one: they lead to modular software. Since I have worked intensely with the corresponding technologies (except maybe the still new Java Module System which I have, to date, never seen applied in an enterprise application), I will compare them to each other to give you an impression and idea when you apply it. This book gives a short introduction to each of these topics on a broad scale that allows a big picture of modularity to be recognised. This book presents only those details on the three topics that help to see how similarly or differently modularity can be treated. It will not give you an overview about all the features these three approaches provide. Finally, a fourth approach to modularity called a Modular Monolith is explained in this book.

Before looking at a single specific approach we must, at first, find a common understanding about what modularity actually is and what it is good for. This is why the book starts with an academic chapter that lays some theoretical foundations. The second chapter introduces an arbitrarily chosen but typical business domain to illustrate the approaches to modularity. Each of the following chapters focuses on a single approach to modularity and introduces one or two different implementations of the business domain introduced in chapter 2. After presenting a number of implementation in the orchestration design, chapter 7 deals with three choreography implementations. The second last chapter of the book compares all of the implementations introduced in this book on a technical level. The final chapter compares the different modularity approaches on a more abstract level, strikes common balances, draws general conclusions and leads to related topics such as Domain Driven Design (DDD) or transactionality.

For the implementations mentioned in the book, code is available at GitHub. This enables you to import all implementations in your IDE and give you the chance to apply modularity yourself by viewing, studying, running, modifying and rerunning the code. The combination of the GitHub code and the explanations as well as project reports in this book should give you a good idea what it really means to take those ways to modularity. Now I hope that you share my opintion that the main title of the book holds its promise.

For those readers who are curious about the answers in the title questions, I would like to answer them right away: for OSGi and micro-services my answer is “Yes, but No” and for the Java Module System “No, but Yes”. While reading you will note the arguments for these answers differ of course, because micro services, OSGi and Java Modularity are different things.

Finally, I would like to emphasise that this book can be taken as a practical guide to modularity in Java written by a developer for developers. I do hope it helps you gain new insights and find adequate, good solutions in your daily work.

Postface

If you find the layout not optimal, please consider that different types of devices may render content in slightly different ways. However, the PDF version and the EBUP version should look very much readable.

You may perhaps find technical details that need improvement or even correction. Feel free to let me know.