REACTIVE DESIGN PATTERNS

反应式设计模式

罗兰·库恩(Roland Kuhn)
[美] 布赖恩·哈纳菲(Brian Hanafee) 著
杰米·艾伦(Jamie Allen)

何品 邱嘉和 王石冲 译

林炜翔 审校

MANNING

清华大学出版社

反应式设计模式

罗兰·库恩(Roland Kuhn)
[美] 布赖恩·哈纳菲(Brian Hanafee) 著
杰米·艾伦(Jamie Allen)
何 品 邱嘉和 王石冲 译
林炜翔 审校

清华大学出版社
北 京

Roland Kuhn, Brian Hanafee, Jamie Allen
Reactive Design Patterns
EISBN: 978-1-61729-180-7
Original English language edition published by Manning Publications, 178 South Hill Drive, Westampton, NJ 08060 USA.

北京市版权局著作权合同登记号　图字：01-2017-4218

图书在版编目(CIP)数据

反应式设计模式 / (美)罗兰·库恩(Roland Kuhn)，(美)布赖恩·哈纳菲(Brian Hanafee)，(美)杰米·艾伦(Jamie Allen) 著；何品，邱嘉和，王石冲 译. —北京：清华大学出版社，2019
书名原文：Reactive Design Patterns
ISBN 978-7-302-51714-6

Ⅰ. ①反…　Ⅱ. ①罗…　②布…　③杰…　④何…　⑤邱…　⑥王…　Ⅲ. ①软件开发　Ⅳ. ①TP311.52

中国版本图书馆 CIP 数据核字(2018)第 259605 号

责任编辑：王　军　韩宏志
封面设计：周晓亮
版式设计：思创景点
责任校对：牛艳敏
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市铭诚印务有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　印　　张：24.75　　字　　数：471 千字
版　　次：2019 年 1 月第 1 版　　印　　次：2019 年 1 月第 1 次印刷
定　　价：98.00 元

产品编号：075684-01

致 Reactive Design Patterns 中文版读者

能看到《反应式设计模式》以其他语言出版，是我的荣幸；中文版的推出，将使中国的众多软件工程师以及系统架构师更便捷、更充分地获知书中内容。在见证了译者 Wayne Wang(王石冲)、Kerr(何品)、Hawstein(邱嘉和)以及他们的技术评审 Neo Lin(林炜翔)在翻译过程中投入的饱满热情以及对细节的极致追求后，更让我深感荣幸。亲爱的读者，你们此时手中所握，正是他们辛勤付出的结晶，而我也深受感动和启发。我衷心期望，本书能为你们开拓“论证回弹性分布式系统设计”的新思路，在全球人类文明互联的当下，能帮助你们继续挑战软件工程进步的极限！

Roland Kuhn 博士
《反应式设计模式》的主要作者

Letter for the Chinese Translation of Reactive Design Patterns

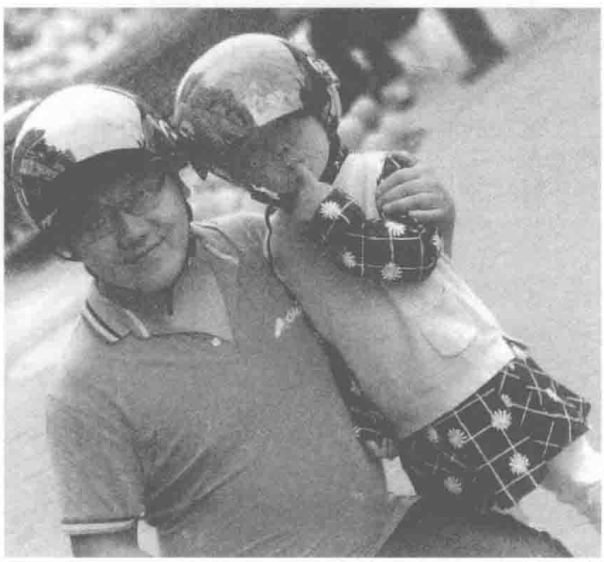
It is a great pleasure to see *Reactive Design Patterns* translated into foreign languages, in particular for China where many software engineers and architects are not adequately reached by the English version. An even greater pleasure was to witness the enthusiasm and attention to detail put into this effort by 王石冲 (Wayne Wang), 何品(Kerr), 邱嘉和(Hawstein), and their technical editor 林炜翔(Neo Lin). I am very grateful and inspired by their hard work which you, dear reader, are now holding in your hands. May this book open up new ways of reasoning about resilient distributed systems for you and help you in continuing to push the envelope of software engineering in the globally connected age of human civilization.

Dr. Roland Kuhn

Main author of *Reactive Design Patterns*

译者简介

何品　热爱反应式编程，也是 Akka 和 Netty 等项目的贡献者，活跃于 Scala 社区，目前就职于淘宝。



邱嘉和　热爱编程与开源，Akka 贡献者，活跃于 Scala 社区，曾就职于豌豆荚与 GrowingIO，目前在创业中。

王石冲　Scala 程序员，Lightbend 全家桶爱好者，反应式宣言践行者，目前在数云公司任架构师一职，从事流式数据引擎开发。

技术审校简介：

林炜翔　对软件系统工程化精益求精的中年程序员，曾参与多种业务场景下的软件开发。他对跨领域的软件系统设计有独到的见解。

译者序一

我从 2012 年开始接触 Netty，并随后了解到 Scala 和 Akka。随着对这些技术栈的深入学习，逐渐建立了自己的知识脉络和体系。源于对这些项目的热爱，贡献了一些代码，也结识了一些朋友。

我本来打算与嘉和写一本关于 Akka 的书，甚至已经设计好封面，不过当时由于 Akka 的前技术领导者主笔的这本书快要出版，嘉和就鼓动我翻译，我们一拍即合。不过本书的翻译难度颇高，涉及多个领域的相关知识，王石冲的加入为我们注入更大信心，而林炜翔一丝不苟的审校和陈涛的持续投入则极大地提高了本书的翻译质量。

本书围绕反应式宣言(https://www.reactivemanifesto.org/zh-CN)展开，讲述什么是反应式、为何需要反应式，以及反应式系统设计与开发中的一些常用模式，无论是软件开发者还是系统架构师，都可从本书中汲取知识养分。本书提出许多真知灼见，勾勒出反应式应用程序、反应式系统以及反应式平台等概念，有场景、有故事、有概念、有实践，令人沉醉其中，流连忘返。只可惜，受限于本书的篇幅，有些细节的讲解还不够全面透彻，还需要读者参考书中的指引自行研习。

由于篇幅所限，本书的主要例子都用 Scala 语言描述，运用了现成的 Akka 套件中提供的一些功能，但本书提及的概念、知识点、设计和模式并不局限于特定的语言和框架，读者可任意选择并亲手实践。以反应式流(https://github.com/reactive-streams/reactive-streams-jvm)为例，读者可选用 Akka-Stream、Reactor 或 RxJava 进行实践。读者也可多方参考、兼收并蓄，进而融会贯通。

作为本书的译者之一，我非常希望，本书可帮助读者构建出反应式系统设计的体系化知识结构，无论大家最终选用何种工具，都能得心应手地构建自己的反应式系统。当然，我也鼓励读者更多地参与到开源项目中，贡献自己的力量。作为本书的早期读者之一，我认为本书作为一本模式书已经足够，而实际开发对领域知识、编程技能以及架构能力都有较高的要求，需要读者进行更多的思考和实践。

我们将本书的相关源代码放到 GitHub，以方便读者下载，并提出问题和反馈意见。另外提供在线版代码清单。在技术审校林炜翔的帮助下，本书翻译质量得到极大提升；作为深耕电信、金融领域多年的外籍专家，他对一些英语文化梗的拿捏也

恰到好处。我们也得到了社区的极大帮助，特别是杨云、沈达、欧宁猫。而在反应式宣言的翻译上，更是得到了多方的帮助。

当然，最应该感谢的是我的爱人和两个女儿，感谢她们的体谅和支持，她们是我一切动力的来源。

何　品

2018 年 7 月于杭州

译者序二

我就读研究生期间就开始接触函数式编程，涉猎过几种函数式语言后，最终情定 Haskell。我对 Haskell 巧妙、灵活地解决问题的能力极为着迷，以至于参加工作成为一名 Java 程序员后，仍在利用业余时间学习 Haskell。再后来，受邓草原老师的影响，开始进入 Scala 世界，踏上学习、使用 Akka 进而为 Akka 项目贡献代码的道路。

Roland 做了多年 Akka 项目的技术领导，对反应式系统和 Akka 有深刻的思考和独到的见解。我在给 Akka 贡献代码的过程中得到其悉心指导，所以得知他正在编写一本关于“反应式系统”设计模式的书籍时，我第一时间在 Manning 上购买了其 MEAP 版本(Manning Early Access Program)，跟随 Roland 的指点，一点点地整理零散的架构模式和原则并形成系统。那时，我曾与何品讨论合写一本关于 Akka 的书籍。当本书英文版临近出版时，我向何品建议翻译本书。这不是一本讲述 Akka 的书，而是讲述反应式架构原则和设计模式，但这些原则和模式却在 Akka 发展过程中不断注入 Akka 中。可以说，Akka 被注入“反应式”灵魂。反过来，用 Akka 构建反应式系统是如此自然和简单。随着 Scala 社区的不断壮大，越来越多的人接触到 Akka，并开始用它构建应用程序。然而，我看到太多人还抱着旧思维模式，在不适当地使用 Akka，导致 Akka 的先进性无法发挥出来。正因为如此，我觉得更有必要翻译这本讲述反应式系统“内功心法”的书籍，以飨中文读者。

翻译书籍是一项艰辛的工作，翻译本书占用了我绝大部分业余时间。这本书最终是何品、石冲和我共同翻译完成的。没有他们，本书也就不可能在如此短的时间内完成。特别是何品，除了翻译书籍本身，还负责处理沟通和协调等事项。另外，特别感谢林炜翔和陈涛，他们怀着极大的热情对本书的译稿进行审校，进一步提高了本书的翻译质量。

当然，最应该感谢的是我的爱人，感谢她在本书翻译过程中给予的理解与支持。

邱嘉和

2018 年 7 月于北京

译者序三

我从 2013 年开始接触 Scala。别人因为畏惧 Scala 望而却步，我却因为觉得简单而欲罢不能。后来才领悟出，这是因为 Scala 是一门可以学习和实践的语言。我能通过理论学习迅速找到 Scala 以及基于它构建的框架的正确用法和最佳实践，而不需要通过“踩坑”来积累经验；我可以迅速地利用 Play 和 Akka 构建高性能、高可用的项目，并在生产中发挥作用。这里必须特别感谢数云公司以及技术总监韩铮对我的信任和“放纵”，使得我可以迅速积累 Play 和 Akka 实践经验，并逐渐整理出自己对使用 Actor 模型的心得。后来因为在何品的 Akka 群里逐渐变得活跃，并因而与他相识。也是机缘巧合之下，由于特别想感受创业公司氛围，我从数云离职，在北京的 GrowingIO 认识了嘉和，并和他一起经历了一段“一年抵三年”的岁月。在 2017 年 6 月，嘉和提议翻译，何品寻找到版权方时，我积极响应。之后，我们便开启了《反应式设计模式》的翻译之旅。

说实话，本书在 Akka 爱好者中可以说是万众期待。Akka 因为使用了 Actor 模型而闻名，但为什么使用 Actor 模型就能带来诸多好处呢？“放任崩溃”是好模式，但崩溃后怎么恢复呢？Actor 在分布式中是一把好手，但分布式下的 Actor 又有什么不同呢？然后，又如何基于 Actor 的消息发送机制得到 Akka Stream，这种流式处理又有哪些优点呢？还有，应该如何应对分布式数据复制、资源管理、流量控制、状态管理等不同场景？Akka 文档足够详尽，但很多功能的设计思想并没有阐释清楚，很容易让人落入只知其然不知其所以然的状态。

而本书能最好地解答你的几乎所有疑问。其中很多理论并不局限于 Akka，在构建高性能、低耦合、健壮的分布式系统中，也能大放异彩。作为还算资深的 Akka 使用者，我也是跟随本书的讲解，才真正弄懂 Akka 在设计上的许多精妙之处，以及反应式流的基本原理和 Akka Stream 的正确用法。这里必须感谢原作者们奉献了这本我们认为必成经典的书籍。

当然，不可否认的是，书里面仍有很多不足之处。有时读着会忽然发现意犹未尽，却戛然而止。我们曾就此询问 Roland 博士，他坦言当时因为篇幅和个人精力所限，很多地方未能深入展开。不过，大致方向已经列出来了。所以读者在阅读时，如果发现有些地方不够深入，可自行查找资料，以便更深入地理解和应用。

如果遇到任何问题，可以访问 Github 向我们提问；如果问题得到确认，我们会请你喝可乐。衷心希望所有阅读本书的读者，都可与我们共享编程乐趣！

在本书翻译过程中，需要特别感谢技术审校林炜翔、喜马拉雅 fm 的陈涛以及来自何品 Akka 群的各位帮忙评审的朋友，正是在他们的帮助和监督下，本书翻译质量才能得到保证。尤其是林炜翔，因为他加拿大华人的身份，帮助我们解读了很多也许只有土生土长的外国人才能理解的梗和俗语。希望有今后的工作中，我能把从翻译过程中学到的知识成功应用到公司的产品开发中，创造更大价值。

最后感谢我的妻子。你是我做一切事情的动力。

王石冲

2018 年 7 月于上海

技术审校序

在2013年，因为 Spark 初创团队在加拿大从事业务推广，我才了解到 Scala 以及其生态圈的存在，而那时的 Spark 只是蒙特利尔小众技术 meetup 里面的一个小话题。我并没有因为很早就接触 Spark 而成为大数据风口的追风人，反而在那间昏暗的地下室里看到了 Scala 带来的无限潜力。作为一个中年人，一个资深编程人员，在代码质量和业务效率的平衡上，我一直有自己独到见解和追求，而且这一路走来，其实累积了不少疑惑。而 Scala 及其充满活力的社区为我敞开了一扇新的大门，得以验证我长期以来的一些想法，以便更好地完善和沉淀自己的知识体系。

在寻找这些答案的路途上，我甚至完全偏离了自己原先的职业轨迹，投身 IT 领域做电商开发，再甚至从加拿大回到中国，尝试一些分布式、纯函数式编程的高级商业应用实践。

至于参与本书的技术审校，也是在这条道路上的一段加速小跑。本书尝试建立(尚需完善)的理论体系基础，是原作者多年来理论和实践的升华，他希望通过把这些认知提炼为落到纸上的文字，以便为追求工程质量的同行助力。然而对于大部分读者而言，原版书中有不少技术名词或理论显得相对生僻，一些细心的读者甚至在一些中文名词的选用上另有看法，对于这一点，我希望读者多花点时间通读全书，再自行判断；希望读者能在这个过程中，审视、回顾甚至推翻自己的某些知识结构，构建出更强大的知识体系。毕竟，每个软件工程师都有改变世界的梦想，可我们很多时候只是获得了改变旧世界的工具，但缺少一个构造新世界的蓝图。

希望本书的面世，能成为广大软件工程工作者勾勒各个宏大蓝图时的有益借鉴。

最后，感谢远方的家人，对我的任性给予理解、包容和支持。

林炜翔

2018年7月于北京

序　言

非常感谢 Roland 耗费心血撰写这本基础书籍，我实在想不出还有谁更有能力胜任这项工作。Roland 是一个思维敏锐、知识渊博的人。Roland 与其他人合著了《反应式宣言》(*Reactive Manifesto*)，并多年来一直是 Akka 项目的技术领导。他还参与编辑和讲授了 Coursera 平台上广受欢迎的“反应式编程和设计”课程，是我遇到过的最优秀的技术作家。

诚然，我对本书的出版感到兴奋。本书概述了反应式架构/设计的含义，并且务实、透彻地分析了反应式的各种第一原理/公理。此外，本书也给出了反应式设计模式的分类目录(catalog)，如何构思系统设计，以及这些模式之间的整体关联——就如 Martin Fowler 的《企业应用程序架构》在 15 年前所做的一样。

在自己的职业生涯中，我亲眼看到了弹性、松耦合、消息驱动的系统可带来巨大效益，与那些隐藏了分布式系统本质的传统方案相比更甚。2013 年左右，我想将这些经验教训规范化，《反应式宣言》应运而生。我记得，《反应式宣言》最初只是我在 Typesafe 公司(现在的 Lightbend 公司)的内部技术会议上分享的一些草稿。恰巧，这次技术会议和 Scala Days New York 同场，期间又遇上了 Roland、Martin Odersky 和 Erik Meijer，他们在那里拍摄了他们那个“不专业”但又蛮有趣的反应式编程 Coursera 课程宣传片。宣言中基于各项反应式原则的探讨引起了其他工程师的共鸣，在 2013 年 7 月发布了初版。此后，该宣言从开发社区收到了大量反馈意见。Roland、Martin Thompson、Dave Farley 和我对该宣言进行了大量改进和修正，并最终在 2014 年 9 月发布了第 2 版。到目前为止，已有超过 2.3 万人签署了这个宣言。在此期间，我看到“反应式”进展显著，从一项几乎不被承认的、只在少数公司的一些边缘系统中使用的技术，发展为众多大公司的整体平台战略，涵盖中间件、金融服务、零售、社交媒体、彩票以及游戏等各个不同领域。

《反应式宣言》将“反应式系统(Reactive Systems)”定义为一系列架构设计原则，旨在解决系统当前和未来面临的挑战。这些原则也非首开先河；可追溯到 20 世纪 70 年代和 80 年代由 Jim Gray 和 Pat Helland 在 Tandem System，以及 Joe Armstrong 和 Robert Virding 在 Erlang(及 OTP)上所做的开创性工作。然而，这些先驱者都领先于他们所处的年代，直到过去五年，科技产业才被迫重新思考当前企业软件开发的最佳做法，并学会将得来不易的反应式知识框架应用到当今的多

CPU 核心架构、云计算以及 IoT 领域。

至今，反应式的多项原则已对行业产生巨大影响；与许多成功想法一样，它们被过度使用和重新诠释。这不纯粹是坏事；思想需要不断进化，从而保持相关性。但这也会引起混淆，导致原始意图被淡化。例如，一种不幸的流行误解是：反应式编程无非就是使用回调或面向流的组合子进行异步和非阻塞的编程(这些技术归结成反应式编程)。如果仅关注这个角度，就意味着将错过反应式各项原则的其他许多好处。本书的贡献之一便是从更宏大的视角(系统视图)将焦点从一个模块如何独立提供服务，聚焦到如何设计一个具有协作性、回弹性和弹性的系统：反应式系统。

与 *Design Patterns: Elements of Reusable Object-Oriented Software* 和 *Domain-Driven Design* 一样，这本明日经典书籍也是每位专业程序员的案边必备。预祝你享受到阅读和学习的乐趣！

——Jonas Bonér

Lightbend 公司 CTO 和创始人

Akka 创始人

自　序

早在正式加入 Akka 团队前，Manning 出版社的 Mike Stephens 就试图说服我写一本关于 Akka 的书籍。我很想说："好啊！"，但我当时正要变动工作和国籍，我的妻子也提醒我：写一本书需要付出大量心血。但此后，写一本书的想法便在我心中扎根了。又是三年，在《反应式宣言》发表后，Martin Odersky、Erik Meijer 和我在 Coursera 平台上讲授 Principles of Reactive Programming 这门公开课程，期间参与学习的学生逾 12 万人。这门课程的最初想法来自于 Typesafe 的一次开发者会议，在那次会议上，我向 Martin 建议，我们应通过演示如何在规避陷阱的同时高效地使用这些反应式工具，来促进反应式编程的蓬勃发展——结合我自己在 Akka 的邮件列表上答疑的经验，我知道人们通常都会有哪些疑惑。

视频课程效果不错，覆盖面广，和学生们在讨论组互动，能普遍改善大家的"生活水准"。不幸的是，由于受形式上的限制，在这个主题上进行的讨论的深度和广度都是有限的。毕竟在七周时间里，只能展示那么多内容。因此，我还是渴望写一本书来传达我对反应式系统(Reactive system)的思考。如果只描述 Akka，内容将过于简单，我觉得，要是我写一本书，那么它应涵盖更大范围。我喜欢研究 Akka，它确实改变了我的生活，但 Akka 仅是一种表达分布式和高可靠系统的工具，并非所需要的唯一工具。

于是，我就开始了你手中这部作品的创作之旅，这项任务十分艰巨，我知道我需要助力。幸运的是，那时 Jamie 刚完成他的 *Effective Akka*，所以立刻加入了创作队伍。可白天写书对我们来说太奢侈了，导致本书的启动过程很慢，并且一直都在延期。原计划在 Principles of Reactive Programming 课程的第一次迭代期间就将前三章内容放在网上，并开启早期预览计划，可最终不得不延后数月。令我们惊讶的是，即使我们知道某个观点主要写哪些内容，可当将心中的想法真正转化成文字时，却发现缺少很多细节。随着时间的推移，Jamie 的日常工作越来越繁忙，不得不完全停止参与创作。再后来，Manning 的技术开发编辑 Brian 参与到了这个项目，很快变得很明显的就是：他不仅提出了非常好的建议，并且以身作则，所以也将他作为合著者之一写在封面上。最终，我在 Brain 的帮

助下完成了本书手稿的撰写。

本书不仅包含关于何时以及如何使用反应式编程以及相关工具的建议，也解释背后的缘由；这样你就可以根据自己的需要做适当的调整，从而满足不同的需求以及新的应用场景。我希望本书能激励你更多地学习，并进一步探索“反应式系统”的奇妙世界。

Roland Kuhn