恰如其分的软件架构

JUST ENOUGH SOFTWARE ARCHITECTURE

风险驱动的设计方法

A RISK-DRIVEN APPROACH

David Garlan教授作序推荐

[美] George Fairbanks 著
张逸 倪健 译
高翌翔 审校

華中科技大學出版社
http://www.hustp.com

恰如其分的软件架构

JUST ENOUGH SOFTWARE ARCHITECTURE

风险驱动的设计方法

A RISK-DRIVEN APPROACH

[美] George Fairbanks 著
张逸 倪健 译
高翌翔 审校

華中科技大學出版社
http://www.hustp.com

内 容 简 介

本书描述了一种恰如其分的架构设计方法。作者建议根据项目面临的风险来调整架构设计的成本，并从多个视角阐述了软件架构的建模过程和方法，包括用例模型、概念模型、域模型、设计模型和代码模型等。本书不仅介绍方法，而且还对方法和概念进行了归类和阐述，将软件架构设计融入开发实践中，与敏捷开发方法有机地结合在一起，适合普通程序员阅读。

湖北省版权局著作权合同登记　图字：17-2013-093 号

图书在版编目(CIP)数据

恰如其分的软件架构/[美]George Fairbanks 著；张逸，倪健 译；高翌翔 审校. —武汉：华中科技大学出版社，2013.9
ISBN 978-7-5609-9075-0
Ⅰ.恰… Ⅱ.①G… ②张… ③倪… ④高… Ⅲ.软件设计 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2013)第 113598 号

恰如其分的软件架构

[美]George Fairbanks 著　张 逸 倪 建 译
高翌翔 审校

策划编辑：徐定翔　　责任校对：李　琴
责任编辑：熊　慧　　责任监印：周治超
出版发行：华中科技大学出版社(中国·武汉)
武昌喻家山　邮编：430074　电话：(027)81321915
录　　排：华中科技大学惠友文印中心
印　　刷：湖北新华印务有限公司
开　　本：787mm×960mm　1/16
印　　张：23.5
字　　数：447 千字
版　　次：2013 年 9 月第 1 版第 1 次印刷
定　　价：88.00 元

本书若有印装质量问题，请向出版社营销中心调换
全国免费服务热线：400-6679-118　竭诚为您服务

读者对本书的赞誉

What Readers Are Saying About Just Enough Software Architecture

如果你打算阅读一本关于软件架构的书，那就选择这一本吧。《恰如其分的软件架构》涵盖了每一位程序员、开发人员、测试人员、架构师、经理都必须知道的软件架构的基本概念，它提供了很多在实战中非常实用的建议，而这只需要你花几个小时去阅读！

——Michael Keeling，专业软件工程师

本书反映了作者造诣高深的软件架构知识和丰富的行业经验。如果你是一位架构师，一定会希望公司里的开发人员都能读一读这本书。如果你是一位开发人员，请仔细阅读。本书介绍了在软件项目实战（没有你想象中那么理想）中的架构工作。它描述了你将会碰到的环境，以及如何在这样的环境中提升自己的设计实战能力。

——Paulo Merson，在职软件架构师、软件工程学院访问学者

Fairbanks 把笔墨聚集在“恰如其分”的软件架构上，这对于每一位想要使架构过程变得更容易的开发人员来说，都是极具吸引力的。本书通过详细的案例和建议，展示了如何用风险驱动来管理架构的建设和范围，重点突出，易于理解。同时，作者提供了软件架构学术方面的很多细节，这对那些对理论和实践都很感兴趣的开发人员非常有益。

——Bradley Schmerl 博士，卡内基·梅隆大学计算机科学学院资深系统科学家

George Fairbanks 的《恰如其分的软件架构》一书中的风险驱动建模方法已经被 NASA Johnson Space Center（JSC）成功地应用于 eXtensible Information Modeler (XIM) 项目。项目组的所有成员，从项目管理人员到开发人员，都必须遵循。实际上，这本书应该是每一位开发人员的必备工具。仅仅是讲述代码模型和反模式的部分，就值回书价了。

——Christopher Dean，
美国国家航空航天局约翰逊空间中心工程科学团队 XIM 首席架构师

《恰如其分的软件架构》教你如何在战略和战术上使用工具，以及如何为你的软件项目选择架构策略。无论你是一位开发人员还是架构师，本书都是你在架构过程中的必备参考资料。

——Nicholas Sherman，微软项目经理

Fairbanks 将过程、生命周期、架构、建模及服务质量方面的最新理念集成在一个条理清楚的框架中。这个框架可以立即应用于你的 IT 应用。Fairbanks 的写作异常清晰、精确，同时具有很强的可读性和趣味性。《恰如其分的软件架构》是 IT 应用架构方面一个具有重要贡献的文献，对于企业应用架构师来说，也许会成为他们的标准参考资料。

——Ian Maung 博士，花旗企业架构部门资深副总裁，Covance 前企业架构总监

本书完全满足了那些软件开发实践者的关键需求，即如何有效地创建更加实际的系统。Fairbanks 常常运用自己的经验，并与学术理论相结合，为我们提供一个又一个概念模型、领域（或更广范围）内的最佳实践，以及在软件架构方面如何体现其现实意义，具有指导性作用。他在书中提出了基于风险的架构方法，并帮助我们认识到怎样才是“恰如其分”的。本书的问世为软件架构领域又增添了一份重要的文献。

——Desmond D'Souza，《MAp and Catalysis》一书的作者，Kinetium, Inc.

本书展现的软件架构将帮助你构建软件，而不会阻碍软件的构建；本书能够让你关注那些真正值得关注的关键性架构工作，从而避免影响编码工作。

——Kevin Bierhoff 博士，专业软件工程师

很多系统和软件开发人员常常追问为什么要做，以及针对什么做软件架构，他们一定会感谢本书的作者在这本书中呈现了清晰的论证和精彩的推理；对于纠结何时，以及如何做架构的开发人员，也会在本书中找到恰如其分的指导，当然还有很多概念和思想。总之，本书简洁易懂，涵盖了很多可供参考的内容。的确，这是一本架构精到、设计精心的好书！

——Shang-Wen Cheng 博士，航空软件工程师

序

Foreword

20 世纪 90 年代，软件架构成为软件工程的一个分支，引起人们广泛的关注。好的软件架构也成为构建成功软件系统的一个关键性因素。随之而来，为了支持架构设计，一大批让人眼花缭乱的符号、工具、技术、过程被引入现有的软件开发实践。

然而，尽管产生了很多关注于软件架构的理论和原则，但在很多情况下仍然无法找到通用的实践性方法。部分原因是在架构的角色定位上产生了分歧。一部分人提倡以架构为中心的设计，即架构在整个软件开发过程中扮演最核心、最关键的角色。持这种观点的人倾向于建立包含全面细节的架构设计、严格定义的架构里程碑及标准化的架构文档。另一部分人则希望弱化架构，主张架构随着产品设计自然呈现，也就是说，架构作为一种特殊的系统类，根本没必要重点关注。持这种观点的人倾向于尽量不要把架构设计活动从实现中分离出来，同时减少，甚至完全忽略架构文档。

显然，这两种方法都不能完全适合所有的系统。实际上，最核心的问题应该是：“对一个特定的系统，应该做多少相应的架构设计工作？”

在本书中，作者 George Fairbanks 给出了一个答案：“恰如其分的架构”。人们对这个答案的第一反应可能是不以为然，因为这种说法可大可小。当然，本书不仅仅只是给出这个答案，还对这个答案在原理上作出了严谨的论述，并指出“恰如其分”的真正内涵。

本书为实现软件架构指出了一条清新的、独具匠心的道路，具有巨大的实践价值。

Fairbanks 认为，决定架构工作是否充分的核心标准是，看它是否能够降低风险。如果设计中的风险很小，就不需要做多少架构工作。如果系统设计有很大的问题，架构就是一个很好的工具。本书真正站在工程的角度，从成本和收益方面来选择技术。尤为可贵的是，它通过关注风险的降低，平衡工程上的收益和进行架构设计的成本，从而获得最好的结果。

很自然，还有很多派生出来的问题需要回答。哪些风险是最好通过软件架构来解决的？如何采用架构设计原理来解决一个设计问题？哪些架构承诺要写下来以便其他人参考？如何确保架构承诺被实现者所遵循？

本书不仅回答了上面的这些问题，而且还回答了实践中更多值得思考的问题。正是由于这些内容，本书为软件架构领域贡献了一种独特的实践。对于正在使用创新方式构建软件系统的人，对于正在面临艰难进行设计权衡决策的人，对于正在尝试在敏捷过程和传统方法间找到平衡的人，简而言之，对于几乎所有的软件工程师来说，这都是一本必备的读物。

David Garlan
计算机科学学院教授
专业软件工程系主任
卡内基·梅隆大学
2010 年 5 月

前言

Preface

在我走上软件开发道路之初，就希望能拥有这样一本书。在那时，介绍语言及面向对象编程的书籍可谓汗牛充栋，而关于设计的书却如凤毛麟角。了解 C++ 语言的特性并不意味着你能设计出一个好的面向对象系统，熟知统一建模语言（UML），也未必能设计出一个好的系统架构。

本书不同于其他介绍软件架构的书籍，区别在于：

风险驱动的架构设计 当风险很小时，设计无须谨小慎微，但当风险威胁到项目的成功时，就没有任何借口进行草率的设计了。许多资历丰富的敏捷软件支持者都认为进行适度的预先设计是有裨益的，而本书则描述了一种恰如其分的架构设计方法。它避免了以“一招鲜，吃遍天”的方式来解决“焦油坑”问题，建议根据面临的风险来调整架构与设计的成本，摒弃仓促草率的做法，通过更为严谨的方式来调整大多数技术的精确度。

促进架构设计的民主化 你所在的团队可能拥有软件架构师——事实上，你可能正是其中一位。我认识的每一位架构师都希望所有开发者能够理解架构。他们抱怨开发者无法理解约束存在的原因，无法认识到表面看来细小的变化怎么会影响系统的属性。本书力求将架构与所有软件开发者联系起来。

积累陈述性知识 能够击中网球与知道为何能击中网球明显不同，心理学家将其分别称为过程性知识（procedural knowledge）与陈述性知识（declarative knowledge）。如果你已经善于设计和构建系统，你会用到本书提供的许多技术，但是，本书更要让你认识到你能做到的事情，并为这些概念命名。这些陈述性知识可以提高你指导其他开发者的能力。

强调工程实践 软件系统的设计者与构建者要做的事情很多，包括安排日程计划、协调资源的承诺及满足利益相关人的需求。诸多软件架构书籍业已涵盖了软件开发过程与组织结构。相对而言，本书将重心放在软件开发的技术部分，处理开发者要做的事情，以确保系统可以工作，即工程学的范畴。它为你展现了如何构建模型，如何分析架构，并在原则的指导下进行设计权衡。它还描述了软件设计者用来分析从中等到大型规模问题的技术，指出了在哪里才能学到专业技术的更多细节。因此，通观全书，软件工程师指的就是开发者，并没有将架构师从程序员中区分出来。

提供实践指导 本书提供了架构的实践方法。软件架构是一种软件设计，设计决策会影响到架构，反之亦然。最优秀的开发者所要做的事情就是深入那些障碍的细节，理解它们，再提炼出这些障碍的本质，从整体上将它们与架构相关联。书中采用的方式是，从架构到数据结构设计，描述具有不同抽象层级的模型，并遵循了这种向下深挖，继而向上提升的行为。

关于我
About me

我的职业生涯源于我对如何构建软件系统的渴求。这种渴求引导我游走于学术研讨与行业软件开发之间。我拥有完整的计算机科学学位：学士、硕士及博士（获得卡内基·梅隆大学软件工程学的博士学位）。我的论文专注于软件框架领域，因为它是许多开发者都要面临的问题。我开发了一种新的规格，称为设计片段（design fragment），它可以用来描述如何使用框架。同时，我还构建了一个基于 Eclipse 的工具，用于验证它们的使用是否正确。我非常荣幸能够得到 David Garlan 与 Bill Scherlis 的指导，并邀请到 Jonathan Aldrich 与 Ralph Johnson 作为论文的评审委员。

我受益于学术的精确与严密，但我的根还是在工程界。我作为软件开发者，参与了多个项目，包括：Nortel DMS-100 中央办公电话交换机、驾驶模拟器的统计分析、时代华纳通信公司的 IT 应用系统、Eclipse IDE 插件，还有我自己创建的网络初创公司开发的每一行代码。我作为一名业余的系统管理员捣鼓着自己的 Linux 机器，拥有一间闪烁着灯光、用电力供暖的小房间。

我在敏捷技术的早期就成为它的拥趸，1996 年，我成功地鼓动我的部门将开发周期从 6 个月切换为 2 周，并在 1998 年开始测试先行的开发。

本书适合谁？
Who is this book for?

本书的主要读者是那些实践中的软件开发者。读者应该对基本的软件开发思想，包括面向对象软件开发、UML、用例与设计模式等有所了解。若能拥有实际的软件开发过程的经验，对阅读本书会更有帮助，因为本书的许多基本主张都基于这些常见的经验。若你看到开发者编写了太多的文档，又或者未经深思熟虑就急于编写代码，一定会认识到这种软件开发方式的谬误，需要寻找像本书提供的那些治病良方。本书同样可以作为大学高年级学生或研究生的教材。

对于不同的读者，这里提出了一些期望：

开发新手或学生 如果你已经了解软件开发的基本机制，例如，编程语言和数据结构设计，理想情况下，已经学过通用的软件工程学课程，本书会为你介绍软件的特定模型，帮助你形成软件架构的概念模型。无须绘制大量图形、编写大量文档，这一模型就能帮助你从大型系统的混乱中走出来，理清思路。它还为你提供了诸如质量属性和架构风格等理念的初次体验。你可以学会如何从对小程序的理解，上升到对整个行业规模与质量的理解。它能加速你的成长，使你成为一位高效的、富有经验的开发者。

经验丰富的开发者 倘若你善于开发软件系统，可能会被频繁要求去指导别人。然而，你可能发现你所掌握的架构知识多少有些异于寻常，或许还使用了独一无二的图形标记或术语。本书将提高你指导他人的能力，理解为何你能够在别人苦苦挣扎的领域取得成功，并教给你标准的模型、标记与名称。

软件架构师 在你所在的软件组织中，一旦其他成员无法理解身为架构师的你究竟做了什么，以及为何要这样做，这个角色就会变得处境艰难。本书不仅教会你构建系统的技术，还提供了一些办法帮助你向团队解释你的工作内容与工作方式。或者，你甚至可以将本书分享给同事，使他们成为真正的团队伙伴，以便能够更好地完成工作。

学术研究人员 本书为软件架构领域做出了多个贡献。它引入了软件架构的风险驱动模型，这是一种决定为项目作出多少架构和设计工作的方法。它描述了三种架构方法：架构无关的设计、专注架构的设计与提升架构的设计。它还整合了软件架构的两种视角：功能视角与质量属性视角，从而形成一种单独的概念模型。本书还引入了架构明显的编程风格（architecturally-evident coding style）的理念，通过阅读源代码使架构显现。

致谢
Acknowledgments

没有众人的鼎力帮助，本书不可能完成。好几位与我共事的朋友参与了其中的一些章节，对于他们的帮助，我必须致以真挚的谢意。这些朋友包括 Kevin Bierhoff、Alan Birchenough、David Garlan、Greg Hartman、Ian Maung、Paulo Merson、Bradley Schmerl 和 Morgan Stanfield.

还有其他人为我早期糟糕的草稿殚精竭虑，他们发现了一大堆问题，并提供了非常有益的指导。Len Bass、Grady Booch、Christopher Dean、Michael Donohue、Daniel Dvorak、Anthony Earl、Hans Gyllstrom、Tim Halloran、Ralph Hoop、Michael Keeling、Ken LaToza、Thomas LaToza、Louis Marbel、Andy Myers、Carl Paradis、Paul Rayner、Patrick Riley、Aamod Sane、Nicholas Sherman、Olaf Zimmermann 和 Guido Zgraggen，谢谢你们！

对于这些年来为我的成长提供指导的所有人，我若未能一一致谢，一定是我的疏漏。首先是我的父母，他们对我的支持远远超过了文字描述所及。我的专业导师包括 Desmond D'Souza 及 Icon Computing 团队等，我的论文指导老师 David Garlan 与 Bill Scherlis，还有卡内基·梅隆大学的师生们，在此表示感谢！

如此精美的封面，从创意到绘制都出自于我朋友 Lisa Haney(http://LisaHaney.com)之手，而 Alan Apt 在整个写作过程中都对我给予了支持。

本书的编写与排版主要使用了一些开源工具，包括 Linux 操作系统、LYX 文档处理器、Memoir LATEX 的样式、LATEX 文档排版系统及 Inkscape 绘图编辑器。大多数图表均使用了 Microsoft Visio 及 PavelHruby 的 Visio UML 模板。

目录

Contents