架构师书库

软件架构设计

实用方法及实践

[墨] 温贝托·塞万提斯（Humberto Cervantes）
[美] 里克·卡斯曼（Rick Kazman） 著
刘旭斌 陈瑶 邵元英 栾云杰 译

DESIGNING SOFTWARE ARCHITECTURES
A Practical Approach

机械工业出版社
China Machine Press

架构师书库

DESIGNING SOFTWARE ARCHITECTURES

A Practical Approach

软件架构设计

实用方法及实践

[墨] 温贝托·塞万提斯（Humberto Cervantes）
[美] 里克·卡斯曼（Rick Kazman） 著

刘旭斌 陈瑶 邵元英 栾云杰 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

软件架构设计：实用方法及实践 /（墨）温贝托·塞万提斯（Humberto Cervantes）等著；刘旭斌等译 . —北京：机械工业出版社，2017.7
（架构师书库）
书名原文：Designing Software Architectures: A Practical Approach

ISBN 978-7-111-57381-4

I. 软… II. ① 温… ② 刘… III. 软件设计 IV. TP311.5

中国版本图书馆 CIP 数据核字（2017）第 165463 号

本书版权登记号：图字：01-2016-8667

软件架构设计：实用方法及实践

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：关　敏　　责任校对：李秋荣
印　　刷：北京市荣盛彩色印刷有限公司　　版　　次：2017 年 7 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：13.5
书　　号：ISBN 978-7-111-57381-4　　定　　价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

为什么要翻译这本书

从机械工业出版社华章公司王春华老师那里获知有一本有关软件架构设计的图书正在征集译者，作为软件从业人员，我表示出极大的兴趣。在看过英文版并试译完样章后，我和几位志同道合的软件工程师一起开始了本书的翻译工作。

在此要感谢机械工业出版社的关敏老师、王春华老师给予我们的支持和信任。因为这份信任，我们才有机会来翻译这本关于软件架构设计的书籍。

本书系统地讲解了一个通常我们认为很复杂的问题——软件架构设计，里面提供的案例有很强的可参照性，所涉及的工具也具有很好的可操作性。

书中介绍了架构的设计过程以及设计方法：属性驱动设计（ADD）。利用 ADD，可以帮助使用者在设计过程中不断重构设计。作者通过介绍 ADD 的概念和 ADD 的几个应用实例，展示了如何执行架构设计，如何重用设计概念，即借用其他成熟的解决方案。本书特别适合想要“从入门到精通”掌握软件架构设计的读者。

在介绍架构设计的概念、方法、流程和理论时，本书结合了非常新鲜、实用的实例，一步一步演示使用 ADD 方法完成架构设计的方方面面，通俗易懂。在学习理论方法的同时，还能够了解时下广泛发展的一些框架和技术，相信读者会从本书中学习到翔实的架构知识，从而受益匪浅。

作为软件从业人员，我参与过软件开发、测试和项目管理，在实际工作中深刻地认识到好的软件架构对于软件产品的决定性意义。参与翻译本书的过程也是一次满足自己好奇心和求知欲的过程。希望读者在阅读本书的过程中也能获得和我一样的愉快体验。

最后，由于译者水平有限，书中难免出现疏漏之处，恳请读者批评指正。

读者对象

本书适合以下几类读者阅读：

- 软件架构师

- 有志于成为软件架构师的软件从业人员
- 计算机相关专业的在读学生

勘误支持

虽然译者试图努力保证本书中不出现错误，但鉴于译者的知识水平和视角，本书中难免出现用词错误或者技术问题。在此，恳请读者不吝指教，指出错误。请读者发送邮件到 cy.yss@163.com，帮助我们修正错误。

译者分工

本书由来自 IBM 中国开发中心的软件工程师及项目经理联合翻译完成。其中刘旭斌翻译了第 1 章、第 7 章和第 9 章；陈瑶翻译了第 2 章和第 3 章；邵元英翻译了第 4 章和第 8 章；栾云杰翻译了第 5 章、第 6 章和第 10 章。附录部分由上面四位共同负责。

致谢

感谢华章公司引进本书的中文版版权，这是本书中文版得以面市的核心要素。

感谢华章公司的关敏和王春华老师，她们专业的编辑水平为本书提供了重要的质量保证。

感谢本次翻译组的小伙伴，他们的热忱使翻译过程变得有趣而且顺畅。

前　言

提起软件架构，人们常常会想到模型——模型表示构成软件架构的基本结构。偶尔，人们才会思考这些结构产生的过程，到底经过什么样的思考过程才有了这些结构，也就是说，设计的过程是什么。设计是一种完成起来很复杂的活动，关于设计的主题也比较复杂，不容易写清楚，因为这需要针对系统的方方面面来考虑并做出决策。这些方面往往很难表达，尤其当它们来自于以往实战性的软件开发项目时，从这样的项目中得来的经验和知识是很难言传的。尽管如此，因为设计行为本身是建立软件架构的基础，所以它亟待被解释。虽然经验很难通过一本书来传授，但是我们可以通过分享一种方法，来帮助读者以系统化的方式完成设计过程。

本书的主旨是介绍设计过程和一种特殊的设计方法，这种方法称为属性驱动设计（Attribute-Driven Design，ADD）。我们相信这种方法非常有效，能帮助读者以有原则、有纪律和可重复的方式完成设计。在本书中，列举了属性驱动设计及现实生活中的几个有关属性驱动设计的真实案例。我们将通过这些案例演示如何进行架构设计。即便你目前没有足够的设计经验，我们会举例说明如何借助该方法来复用设计概念，即那些历经考验的经典方案。

尽管属性驱动设计十多年前已经提出，关于它的文字资料却很少，也很少有资料可以提供属性驱动设计的实例并对其具体实现过程加以解释。因为公开信息的缺乏，人们很难使用该方法或将该方法传授给他人。此外，一些已经发表的关于属性驱动开发的文档也都比较概括，很少涉及架构师日常使用的概念、实践和技术。

我们已经跟职业架构师一起工作了多年，曾指导他们如何进行设计，以及如何在设计过程中学习。同时我们也学到了很多，例如，我们了解到职业架构师在设计过程的早期会考虑哪些技术因素，这一点在之前的属性驱动设计版本中是没有的。就因为这个原因，该方法被很多实践者认为跟实际脱节。本书提供了一个修正过的属性驱动设计新版本。在该版本中，我们试图不遗余力地在理论和实践之间架设桥梁，缩小理论和实践之间的差距。

虽然我们已经教授了多年软件架构和设计软件，但是一路走来我们认识到，对没有经验的人来说，软件架构和软件设计太难了。这种认识促使我们去创建设计路线图，可

以肯定的是，这样可以有效引导人们完成相关设计过程。我们同时设计了一种针对软件设计教学的游戏，可以作为本书的配套部分。

本书面向的读者首先是那些对软件架构设计感兴趣的人，尤其是那些必须展开这项设计任务现阶段却不得不使用某些临时性方案的行业内人士，本书定会对他们别有益处。而对于有经验的软件架构设计者来说，他们已经有了一套逐步建立起来的设计方法，相信这些读者也能通过本书找到新的思路。例如，如何用看板（Kanban）追踪设计进度，如何利用基于策略的问卷调查分析一个设计理念，如何通过设计方法完成早期的评估预测。再者，对于已经在软件工程学院熟知其他架构方法的读者，则可以得到属性驱动设计与其他设计方法的关联信息。例如，与质量属性工作坊（Quality Attribute Workshop，QAW），与架构权衡分析方法（Architecture Tradeoff Analysis，ATAM），以及与成本效益分析方法（Cost Benefit Analysis Method，CBAM）之间的联系。最后，本书也适合计算机科学或者软件工程专业的学生和老师阅读。我们深信本书中列举的案例研究可以帮助读者理解如何更轻松地完成一系列的设计过程。可以肯定的是，我们已经在课程中运用了相似的案例，并且效果显著。就像爱因斯坦所说的，“举例不是教学时可供选择的方式，而是唯一的方式。”

我们期望本书能够让读者明白，设计其实是有套路可依的，按照这样的方法或者套路，你能够在今后的软件架构设计中设计出更优秀的软件产品。

本书各章内容如下：

- 第 1 章简明地介绍了软件架构和属性驱动设计方法。
- 第 2 章讨论软件架构设计的细节，设计过程的主要输入——架构驱动因子，以及设计的概念，这些概念会帮助你明白如何利用已经过验证的方案来理清这些驱动因子有哪些。
- 第 3 章详细介绍属性驱动设计方法。重点讨论属性驱动设计方法的各个步骤，以及能够用来完成这些步骤的多项技术。
- 第 4 章解释了“绿地”（greenfield）系统的开发实例。在该案例研究中，我们尽力解释如何将第 3 章描述的大多数概念运用到设计过程中，因此，你可以自然地认为该案例研究比较“学术”（虽然该案例源于真实存在的系统）。
- 第 5 章阐述第二个案例研究，该案例是与职业软件架构师合作完成的，因而更加专业、更加详细。它将以翔实的细节展示属性驱动设计如何应用于涉及多种技术的大数据系统的设计中。该案例展示了如何在“新”领域中开发系统，而不是在第 4 章提到的传统领域。
- 第 6 章是一个较短的案例研究，展示如何将属性驱动设计应用于常见的遗留（或棕地，brownfield）系统的扩展设计中。该实例说明架构设计并非是在系统开发第一版时一次完成的，而是在开发过程的不同阶段实施的。
- 第 7 章展示了其他一些设计方法。在属性驱动设计的修正版本中，我们采纳了其

他设计过程研究者的想法，在此简要总结了他们的方法，在向他们的工作致敬的同时，也比较了属性驱动设计与其方法的不同。

- 第 8 章深入讨论了分析这个主题（尽管这是一本关于设计的书）。分析本来就是设计的一部分，所以本章讲述了一些技巧，它们既可以用于设计过程当中，又可以用于部分设计完成后。我们专门介绍了基于策略问卷调查方法的使用，该方法能帮助我们简单有效地理解设计过程中的种种决定。
- 第 9 章展示了设计过程如何适应组织级别的应用。例如，在项目周期的最早期进行一些架构设计有助于评估目标。同时，还展示了属性驱动设计如何与其他软件开发方法协同工作。
- 第 10 章总结了全书内容。

本书附有两个附录。附录 A 给出了各种设计概念的目录，这些设计概念可用于特定的应用领域。该目录集合了我们从各处收集的设计概念，反映了现实中那些经验丰富、训练有素的架构师是如何工作的。目录包含了第 4 章案例研究中使用的设计概念的样本。附录 B 针对 7 个最常见的质量属性提供了一套基于策略的问卷调查（详见第 8 章），同时针对 DevOps 额外提供了一份问卷调查。

致谢

希望能够在此表达我们对审阅人员 Marty Barrett、Roger Champagne、Siva Muthu、Robert Nord、Vishal Prabhu、Andriy Shapochka、David Sisk、Perla Velasco-Elizondo 和 Olaf Zimmermann 的感谢，感谢他们慷慨地提出他们的观点和意见。我们也要感谢 Serge Haziyev 和 Olha Hrytsay，他们帮助我们完成了本书第 5 章。此外，如果漏掉 Serge、Olha 和 Andriy 在内的许多 Softserve 的架构师就是我们失职了，他们对整本著作提供了很多帮助。

Humberto 希望感谢 Quarksoft 公司的主管和架构师小组。关于修改属性驱动设计的很多想法和本书中的一个案例研究都来源于该方法在这家公司的实践。感谢我有幸合作过和交换过意见的其他公司的架构师及开发者，我从他们身上学到了很多。我也希望感谢软件工程学院，他们多年来一直邀请我和其他学者参加他们的精英教育研讨会（ACE Educators Workshop）。我还要感谢我的母校，墨西哥首都伊斯塔帕拉帕自治大学，它一直在支持我。感谢我的同事 Perla Velasco-Elizondo 和 Luis Castro，他们已经在架构之旅中陪伴我多年。感谢 Alonso Leal，是他在多年前给了我成为一个职业架构师的机会。感谢 Richard S. Hall，他教了我许多写作本书时很有价值的技巧。最后，我要感谢我的合作者 Rick，他是个好人，也是个好同事，很高兴能和他一起工作并交换意见。

Rick 希望感谢软件工程学院的 James Ivers 和他的研究小组。我还要特别感谢 Rod Nord 悉心的审校和宝贵的建议。我也要感谢我的长期合作者和导师 Len Bass，在许多年前他引领我开启了软件架构之旅。没有 Len，我不知道自己今天会在哪里。此外，我要感谢 Linda Northrop，她多年来一直大力支持我的研究，并提供给我许多宝贵的机会。最后，我要感谢我的合作者 Humberto，他总是朝气蓬勃，和他共事是一件真正的乐事。

目　录

第1章 引 言

在本章中我们会概述软件架构这一主题。我们会简要探讨架构是什么以及为什么必须在软件系统开发时考虑它。我们还会探讨同软件架构开发相关的不同活动和行为，架构设计——本书的主旨——可以理解为以这些活动为背景进行。我们也会简要地讨论架构师这个角色，该角色负责创建设计。最后，我们会引入属性驱动设计（Attribute-Driven Design，ADD）方法，并在本书中大量讨论该架构设计方法。

1.1 写作动机

本书的目标是教会你如何通过一种系统化的、可预测的、可重复的、高性价比的方法进行软件架构设计。如果你正准备读这本书，那说明你或许对架构设计感兴趣并有志成为一名架构师。好消息是：你的目标并不遥远。要就这一点说服你，我们会花些功夫来谈论设计的想法——针对任何事物的设计——然后我们都会明白架构设计在如何做上是一致的，在为何做上也是一致的。在很多领域，“设计”包含相同的挑战和思考——满足利益干系人的需求，坚守预算和进度，处理约束条件，等等。尽管因为所涉及领域的不同，涉及的基本事物和设计工具也可能不同，但是设计的目标和步骤却并无差异。

这是个令人鼓舞的好消息，因为这意味着设计不只是行家的专有领地。也就是说，设计既可以教，也可以学。大多数设计，特别是在工程领域，由已知的（有时是创新的）基本设计组成，这些设计可以实现可预见的成果。当然，细节最令人头疼，但这就是我们的方法存在的意义。起初这似乎很难想象，像设计这种创造性的工作可以用一种循序渐进的方法来实现；然而，这不仅可能而且还是有价值的，如同 Parnas 和 Clements 在他们的论文《A Rational Design Process: How and Why to Fake It》中讨论的那样。当然，不是每个人都可以成为伟大的设计师，就像不是每个人都可以是托马斯·爱迪生、勒布朗·詹姆斯或罗纳尔多一样。我们要说的是，每个人都可以成为更好的设计师，本书提供了结构化的方法，这些方法源于一些可重用的设计知识，可以帮助你从平凡走向卓越。

我们为什么要写一本关于软件架构设计的书呢？虽然已经有很多关于通用设计的著作，也已经有一些关于软件架构设计的著作，但是还没有一本专门致力于架构设计的书。

此外，大多数已有的关于架构设计的书都比较抽象。

我们写这本书的目标是提供一种实用的方法，可以由任何一个称职的软件工程师来执行，也（同样重要的）提供了一系列丰富的案例研究来帮助你了解该方法。阿尔伯特·爱因斯坦曾经说过，“举例不是教学时可供选择的方式，而是唯一的方式”。我们坚信是这样的。对大多数人来说，比起从规则或步骤或原则中学习，从实例中学习的效果会更好。当然，我们需要用步骤、规则和原则指导我们的行为来创建实例，这些例子围绕着我们日常所关心的事情，并帮助我们把步骤具体化。

这并不意味着架构设计永远都那么简单。如果你正在构建一个复杂的系统，那么你可能正试图平衡多个相互竞争的力量，如上市时间、成本、性能、可进化性、可用性、可靠性等。如果你正在调整任何一个维度的边界，那么你作为一个架构师的工作将更加复杂。不只软件，任何工程学科都是如此。如果你研究的是建造大型船舶、摩天大楼或其他复杂的“系统”的历史，你会看到这些系统的架构师如何艰难地做出适当的决策和取舍。的确，架构设计可能始终是困难的，但我们的目的是对于训练有素、受过良好教育的软件工程师来说，让架构设计易于处理和实现。

1.2 软件架构

关于什么是软件架构已经有很多论述。在这里我们采用《Software Architecture in Practice》（第3版）中软件架构的定义：

一个系统的软件架构是构成该系统所需结构的组合，它们由软件元素、元素之间的关系以及元素和关系两者的属性组成。

正如你将要看到的，我们的设计方法结合了这一定义，并帮助设计者创建出一个具有理想属性的软件架构。

1.2.1 软件架构的重要性

关于软件架构的重要性也已经有很多论述。同样，我们还是采用《Software Architecture in Practice》中的论述。我们注意到软件架构之所以重要有各种各样的原因，以及类似的多种多样的来自这些原因的各种后果：

- 架构会妨碍或支持系统质量属性的实现。
- 在架构中做出的决定允许你在系统发展的过程中分析改变的原因并管理它们。
- 对软件架构的分析使我们可以在系统开发的早期预测系统的质量。
- 书面记录的架构加强了利益相关者之间的沟通。
- 软件架构是最早的设计决策的载体，这些设计决策也是最基本、最难修改的。
- 软件架构定义了一系列的约束条件以及后续的实现。

- 软件架构会影响一个组织的结构，反之亦然。
- 软件架构能够为可改进甚至是一次性的原型设计提供基础。
- 软件架构是架构师和项目经理估算成本和进度的关键工件。
- 软件架构可以作为一个可转移、可重复使用的模型，该模型可以构成产品线的核心。
- 基于架构的开发专注于组件的组装，而不仅仅关注它们的创建过程。
- 通过限制软件架构的备选方案，架构师可以激发开发人员的创造性，减少设计和系统的复杂度。
- 软件架构可以为培养新的团队成员打基础。

如果一个软件架构因为以下这些原因而变得重要：它会影响组织的结构、系统的质量，以及参与它的创建者和改进者，那么一定要在设计这个关键工件时非常小心。可悲的是，大多数情况下往往不是这样的。软件架构经常“演变”或“突现”。虽然我们并不反对演变或突现，且强调不主张“大规模预先设计”，但除非是最简单的项目，不设计任何软件架构往往都是非常危险的。你愿意开车经过一座没有经过仔细设计的桥吗？或者你愿意坐在一架没有经过仔细设计的喷气式飞机上吗？当然不。但你每天使用的软件都是充满缺陷、昂贵、不安全、不可靠、容易出错和缓慢的，许多这些讨厌的特性本来都是可以避免的！

本书的核心信息是，软件架构设计不一定是困难或可怕的，它并不只是行家的专有领地。它不一定是昂贵的，不一定需要预先把所有的事情做完。我们的工作就是告诉你应该怎么做并让你相信这么做是在你的能力范围以内的。

1.2.2 生命周期活动

软件架构设计是软件架构生命周期的活动之一（图 1.1）。在任何一个软件项目的生命周期中，该活动关注的都是将需求转化成设计然后再转化成实现。具体来说，软件架构师需要操心以下问题：

- 架构需求。在所有的需求中，有些需求在软件架构方面特别重要。这些软件架构方面的重要需求（architecturally significant requirement，ASR）不仅包含系统最重要的功能和需要考虑的约束条件，也包含了最重要的一点——质量属性，如高性能、高可用性、容易改进和铁甲一样的安全性。这些需求，以及一个明确的设计目的和其他可能永远不会被写下来或可能对外部利益相关者来说不可见的与软件架构有关的问题，将指导你对软件架构的结构和组件做出选择。我们将把这些软件架构方面的重要需求和关注点作为驱动因子，因为可以说是它们在驱动设计。
- 架构设计。设计是一种从需要的世界（需求）到解决方案的世界的转化。它在结构方面由代码、框架和组件组成。一个好的设计是满足了驱动因子的设计。软件架构设计是本书的关注点。