
PEARSON
HZ BOOKS
The Process of
Software Architecting
架构实战
软件架构设计的过程
（英）Peter Eeles Peter Cripps 著
蔡黄辉 马文涛 译
机械工业出版社
China Machine Press
PEARSON
Education

The Process of Software Architecting

架构实战
软件架构设计的过程

（英）Peter Eeles　Peter Cripps 著
蔡黄辉 马文涛 译

机械工业出版社
China Machine Press

本书从基本原理入手，介绍软件架构设计过程中涉及的一些概念、流程、方法、用到的工作产品及可重用的资源，从第6章开始，通过介绍一个具体的案例来阐述如何定义需求、创建逻辑架构、创建物理架构。在第10章“进阶”中，作者补充说明了架构师和软件开发项目其他方面的关系，后面又说明了各种软件开发项目可能存在的困难及相应的处理方法。

本书理论结合实践，介绍了一些可以应用到整个或部分的架构设计流程中的最佳方法。不管你是一位资深的架构师还是一位有志于成为架构师的初级使用者，通过阅读本书都能从中获益。

本书版权登记号：图字：01-2009-5640

图书在版编目(CIP)数据

架构实战：软件架构设计的过程/(英)伊乐斯(Eeles,P.),克里普斯(Cripps,P.)著;蔡黄辉,马文涛译. —北京：机械工业出版社,2010.4

书名原文：The Process of Software Architecting

ISBN 978-7-111-30115-8

Ⅰ.架…　Ⅱ.①伊…　②克…　③蔡…　④马…　Ⅲ.软件设计　Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（2010）第044629号

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：张少波

北京市荣盛彩色印刷有限公司印刷

2010年4月第1版第1次印刷

186mm×240mm · 16印张

标准书号：ISBN 978-7-111-30115-8

定价：45.00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) 88378991；88361066

购书热线：(010) 68326294；88379649；68995259

投稿热线：(010) 88379604

读者信箱：hzjsj@hzbook.com

对本书的赞誉

“软件架构师这个角色在最近几年很盛行，也被认为是项目成功的一个关键因素。然而，即使在今天，人们对如何分析需求、理解关注点、评估可选方案及构建和编写符合目的的架构描述文档等工作仍然缺少一些常规的理解。Eeles 和 Cripps 在他们这本非常有用和有实践性的书中填补了这个空白。书中的内容清楚易懂，遵循从起始到交付的一个逻辑流程，通过研究一个真实的案例对任务和工作产品进行了清楚的解释和阐述。无论对于新的架构师，还是经验丰富的专家，这都是一本重要的手册。”

——Nick Rozanski，《软件系统架构》的作者之一

“如果您需要一本关于软件架构流程的全面和权威的参考书，那就不用再等待了。Peter Eeles 和 Peter Cripps 已经为这个流程编写了一本权威性的指导参考书。本书中介绍的流程利用一个元模型进行了准确的定义，通过一个真实的研究案例进行了阐述，还清楚地关联到像 UML、RUP 和 IEEE 1471 等这样的关键标准，因此为那些大型项目开发中的软件架构提供了颇有价值的指导。我一点都不怀疑本书会成为许多软件架构师的一本很有价值的参考书。”

——Eoin Woods，《软件系统架构》的作者之一

“Eeles 和 Cripps 把多年的经验汇集到这本指导书中，帮助读者不仅理解架构师生产什么，还理解他们如何生产。本书是一本具有很高实践性的指导书，其中详尽阐述了获得的经验和需要避免的陷阱。已经成为架构师的人将参考本书，因为它能够使他们的技术更完善；而期望成为架构师的人通过阅读它能够获得一些需要多年痛苦的经历才能获得的关键见识。”

——Bob Kitzberger，IBM Software Group 的程序主管、战略家

“就我在这个领域的工作经验来看，软件架构给人的感觉有点像妖术，只有精选的少许专家和天才才有天分从事这项工作。本书先介绍行业最佳实践和作者宝贵的经验，然后把架构解决方案带入一个真实的工程学科的范畴。现在，我有了一本可以传授给新从业者的参考书，一本讲授过去需要多年尝试和出错才能体会到的经验的书。”

——Colin Renouf，英国 Websphere User Group 的副主席，企业架构师和技术作家

译 者 序

光阴荏苒，时光如逝，一转眼，我已经工作十年有余。在这十年间，我基本上一直在从事软件开发的工作。从编码到设计，再到需求，再参与一些管理工作。我相信大部分从事软件行业的人都和我有相似的经历。在这个过程中，大家都会时不时地思考一些和软件开发相关的问题：什么是软件项目？软件项目的目的是什么？如何在资源（人力、成本、时间等）有限的情况下尽可能地开发出高质量的软件产品？架构在软件项目的整个过程中起到什么作用？架构师在软件项目中担任什么角色，能起到什么作用？等等。对于这些问题，在本书中都能找到答案。

本书从基本原理入手，介绍软件架构设计过程中涉及的一些概念、流程、方法、用到的工作产品及可重用的资源，从第6章开始，通过介绍一个具体的案例来阐述如何定义需求、创建逻辑架构、创建物理架构。在第10章“进阶”中，作者补充说明了架构师和软件开发项目其他方面的关系，后面又说明了各种软件架构设计过程中可能存在的困难及相应的处理方法。总的来说，本书理论结合实践，介绍了一些可以应用到整个或部分的架构设计过程中的最佳方法。不管你是一位资深的架构师还是一位有志于成为架构师的初级使用者，通过阅读本书都能从中获益。

本书的第1～5章由我翻译，第6～10章由我的朋友马文涛翻译。由于时间仓促，书中有些处理不妥的地方，敬请谅解。

最后，我要感谢我的父母和妻子，感谢他们在我翻译的过程中所给予的帮助和支持。

蔡黄辉

2010年3月5日

序

软件密集型系统的架构研究目前受到很多关注。虽然已经出现了大量描述软件架构概念的书籍和论文，但是其中很少是描述架构实践的。作者介绍的就是这样一本书。

在本书中，作者提出了他们关于架构的适用的观点：什么是架构？它是如何证明自己的？谁设计的？它是如何与一个软件开发项目的其他工件关联起来的？尽管本书包含了描绘一个系统架构的所有重要的基本原理，但是，本书的重点在于架构方法的深入验证。我说“深入”是因为作者通过使用定义良好的模型来解释他们的方法，除此之外，令人高兴的是，他们采用的方式很容易理解，而且很全面，特别注意组成一个系统架构完整构想的各种观点。具体地说，他们讲到了在一个系统的逻辑架构和物理架构之间建起桥梁。令我尤其高兴的是，他们还包含了人件问题——也就是，架构师和架构团队的角色，以及各个团队成员活动的架构工件的联系。

通过一个相当完整的案例研究，作者将他们的观念运用到了实践中。来自于他们在行业中的实践经验的、人们应该避免的缺陷的解释特别有用。毫无疑问，这是一本非常有用的书，因为它包含了任何软件开发组织可以立即运用的观点。

Grady Booch

前　言

几年前，作者们开始注意到 Grady Booch 首创的《软件架构手册》（《Handbook of Software Architecture》，www. handbookofsoftwarearchitecture. com）。Grady 起初的目的是：

> 整理许多有趣的软件密集型系统的架构，以揭示它们的基本模式以及允许在域和架构风格之间进行比较的方式，并把它们呈现出来。

当 Grady 正关注于最终架构的时候，我们感到理解成功架构师创建他们的架构时所遵循的流程同样很有趣。当然，我们最终的目的是复制他们的成功。我们花了好几年的时间才完成这个过程。我们做了许多项目，和许多架构师进行了交流，还对许多开发方法进行了梳理——所有这些都有助于我们理解当构建一个软件系统时起作用和不起作用的因素的本质。本书是我们经历的这个过程的总结。

许多优秀的书讲述了软件架构过程的特定方面，我们借鉴了这些书中的相关内容。例如，其中有些书注重编写一个软件架构的文档，另一些书注重如何评价一个软件架构。其中任意一方面都适合比较大的场景，因为每一方面都呈现了软件架构过程中的一个重要因素。因此，本书的一个目的是通过提供在一个典型软件开发项目的环境中架构的所有方面的概览来呈现这个大场景。

应该指出的是，本书没有指定一个专门的软件开发方法。更确切地说，本书讲述了人们在支持构建过程的任意现代开发方法中可能遇到的关键因素。

本书是为谁准备的

显然，本书是针对那些想了解他们的角色如何适应整个软件开发过程的软件架构师（或者立志成为软件架构师的人）。本书也适合“专门”的架构师角色，如应用架构师和安全架构师。更笼统地说，本书适合那些想更好地了解软件架构师这个角色的人。就这点而言，它对一个软件开发团队的所有成员也都有好处，包括开发人员、测试人员、业务分析人员、项目经理、配置经理和过程控制工程师。本书也特别适合那些在软件开发的尝试中想了解日益重要的软件架构师角色的大学生。

如何阅读本书

本书大致分为三个部分：

第 1 ~5 章为第一部分，概述了架构、架构师、架构设计、编写软件架构文档、可重用架构资源的核心概念。

第 6 ~9 章为第二部分，这部分包含了相关案例研究的章，通过一个基于样例应用程序的典型软件开发项目，重点体现架构师这个角色，提供一个指导性指南。这些章的编写方式，使你很容易找到感兴趣的特定主题。每个相关案例研究的章主要按照任务进行组织，另外，在这些章中我们使用了一些排版体例。特别是，对流程元素的所有引用，如任务、工件和角色，都用黑体加以强调，例如当我们描述**软件架构文档**工件的时候。

第 10 章为第三部分，包含额外讨论的话题和思考，尤其是，在前面的章中描述的概念如何应用于架构复杂的系统。

在本书中，你还会发现一些如下进行分类的有用的补充内容：

- 概念补充内容：介绍与讨论与主题相关的想法或整套想法。
- 检查清单补充内容：包含当执行某一特定任务时，可以进行检查的有用的项目。
- 最佳实践补充内容：介绍已经在实践中证实为有效的方法。
- 缺陷补充内容：介绍最好避免的方法，因为它们会导致负面效果。

我们在本书中大范围地使用统一建模语言（UML）来描述架构的某些方面。所有的 UML 图表都是通过 IBM Rational Software Architect 创建的。

附属站点

本书有一个附属的站点：processofsoftwarearchitecting. com，读者可以在这个站点上找到额外的信息，也可以和作者进行交流。

致　谢

当编写本书的时候有两位关键人物参与进来了。一位是 Grady Booch，他不但为本书奠定了基础，而且还友好地为本书写了序，提供了详细的审稿意见及自始至终的支持。另一位是 Philippe Kruchten，他从未停止过令人惊奇的经历。此外，Philippe 通过提供许多建议，担当被访问者和评论家，在编写本书的过程中为我们引见了很多架构师进行交流等，对本书提供了自始至终的支持。

同样地，我们还要感谢 Brad Appleton、Dave Braines、Alan Brown、Mark Dickson、Celso Conzalez、Holger Heuss、Bobby Higgins、Rich Hilliard、Kelli Houston、Brad Jackson、Robert Kitzberger、Colin Renouf、Nick Rozanski、Rick Smith 和 Eoin Woods 为本书提供了详细的审稿意见。有了他们，本书才变得更加完善。

在访谈的所有架构师和项目经理中，有人放弃大量的个人时间与我们一起工作。他们是 Ian Charters、Philippe Kruchten、Helen McCann、Gordon McClean 和 Nick Rozanski。

我们还要感谢众多的客户以及那些支持我们的工作从而也支持了本书的同事和熟人。很遗憾，有太多的人要感谢以致不能一一提及，但是我们要特别感谢我们许多的 IBM 同事、电机及电子学工程师联合会/IFIP 软件体系结构（WICSA，Working IEEE/IFIP Conference on Software Architecture）会议的参与者、国际信息处理联合会（IFIP，the International Federation for Information Processing）2.10 工作组（软件架构）的成员以及英国计算机协会（BCS，British Computer Society）软件实践改进（SPA，Software Practice Advancement）专家组。这些会议和讨论会专门提供机会让我们在行业和学术界中的重要人物面前得以展现。

最后，我们还要感谢培生教育（Pearson Education）中参与这个项目的所有人。我们特别要感谢我们的组稿编辑 Chris Guzikowski，还有 Raina Chrobak、Sheri Cain 和产品团队。

作者简介

Peter Eeles 是 IBM 的高级 IT 架构师，他就职于 IBM 的 Rational 品牌软件组。在这个职位上，他帮助组织提高软件开发能力，尤其关注和致力于改进架构流程。Peter 从 1985 年开始从事软件行业，其主要工作是进行架构设计和实现大规模、分布式的系统。Peter 是《Building J2EE Applications with the Rational Unified Process》（Addison-Wesley，2002）和《Building business Objects》（John Wiley & Sons，1998）的合著者。他还是英国计算机协会高级会员（FBCS）、工程技术协会（FIET）会员、IBM 技术人员、Open Group Master 认证的 IT 架构师、特许 IT 专家（CITP）。

Peter Cripps 是英国 IBM Global Business Services 的一位 IT 架构师。他从 1980 年开始从事于软件行业，在这期间，他当过程序员、实时软件工程师和跨多个行业（包括电信、财经服务、零售和政府部门）的流程工程师。Peter 的技术特长领域及兴趣是基于组件和服务的开发技术的运用以及优秀架构方法的开发。他是英国计算机协会会员（MBCS）和特许 IT 专家（CITP）。

目　录

第 1 章

导　言

Bjarne Stroustrup 是 C ++ 程序设计语言的发明者，他曾经说："我们的文明建立在软件之上。"软件实际上已经渗入到我们日常生活的许多方面，从简单得像用普通手机打开的唱"生日快乐"的生日卡片，到像飞机和核电站这样非常复杂的系统，都可以看到软件。实际上，如果没有软件，今天我们认为是理所当然的许多创新和像 eBay 或者 Amazon 这样的组织都不会存在。即使是传统的组织，如那些在金融、零售和公共部门中的组织，也很大程度地依赖软件。在如今这个时代，很难找到一个不以某种方式使用软件处理业务的组织。

架构设计流程就是在这种情况下开始形成的。如果希望目前这种对软件日益依赖的情形得以延续，那么，软件必须具有必需的功能、足够好的质量、在承诺的时间内可用，还要以可接受的价格交付。所有这些特性都直接受软件架构的影响，由此得出结论，假如我们把软件架构设计好，那我们就离期望的目标不远了。

本书旨在指导您体验那些在设计一个软件系统的架构的过程中采用的任务和最优方法。在达到这个目标的过程中，本书将回答一些基础性的问题，如下所示：

- 什么是架构？
- 在一个软件开发项目中架构师的角色是什么？
- 架构师与项目的其他角色是什么关系？
- 关于需求，架构师的角色是什么？
- 如何描述架构？
- 架构师在什么时候以及如何才能生成一个初始的架构？
- 架构师如何精炼架构？
- 架构师什么时候会考虑重用？
- 架构是如何验证的？

1.1　流程应用

在回答上述问题的过程中，我们通篇介绍了一系列在角色、任务和工作产品中经过证实的最佳方法。

> **实践**（practice）是解决一个或者多个经常发生的问题的方法。这些方法也有意作为采用、启动和配置的流程"块"（chunk）。（RUP 2008）

这些方法可以合并到您可能使用的任何流程中，包括瀑布式流程、迭代开发流程（如统一软件开发过程），或者敏捷流程（如 Scrum）。同样地，我们没有把这些方法和任意特定的流程联系起来，虽然我们确实按这样的一个顺序来介绍这些方法：该顺序允许我们在项目生命周期的不同时刻展示它们之间的关系和相关性。我们也鼓励您挑选那些能给您的项目（当然还有您的架构）带来最大价值的方法。

1.2 流程概述

为了达到这个目标，我们认为在本书中提供一个描述关键流程元素的概览图来提高您的兴趣是值得的。在这里，我们仅关注本书中详细讲述的任务的单次流程，虽然在一个项目中会把每一个任务执行好多次（我们把这种情况称为迭代）。

组成这个流程的宏观活动的高级概览如图 1.1 所示，这个图与软件工程方法相一致。

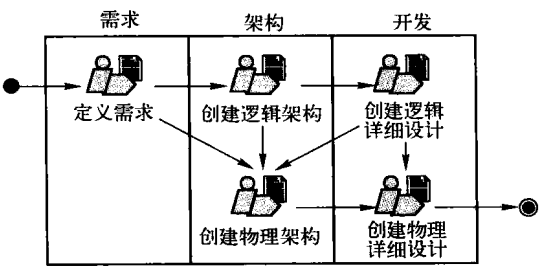

图 1.1 活动概览

这是把定义主要“利益范围和/或协作效率”的任务组织起来的最初分类机制的一个规范。（OpenUP 2008）

正如您所看到的，架构设计活动位于需求和开发的中间。迭代从**定义需求**这个活动中的软件需求定义开始。虽然这个活动主要是业务分析人员的职责，但是，架构师也会参与这个活动的一些详细任务。随后，架构师在**创建逻辑架构**这个活动中创建一个大概的架构。这时候产生的架构不考虑任何技术因素，称为逻辑架构。我们可以把这一步当作是从需求到物理架构的一个跳板，物理架构是针对特定技术的，它组成了实现（编码）的基础。这个逻辑架构会作为**创建逻辑详细设计**活动中执行的任何详细设计的输入。创建逻辑详细设计这个活动会生成在逻辑层面上必需但对架构不必需的任何余留细节的扩充。关于架构与详细设计之间的区别的说明，请参考后面的“概念：架构与设计的比较”部分。

在需求、逻辑架构和逻辑详细设计的基础上，架构师接着在**创建物理架构**这个活动中精炼这种架构，创建物理架构这个活动需要考虑技术因素，最终产生物理架构。这个物理架构会作为**创建物理详细设计**活动中执行的任何详细设计的输入，创建物理详细设计这个活动会形成实现的基础。虽然详细设计和实现这两个活动不是架构师的职责，但是，在需要的时候他们必须给这些团队提供指导。

概念：架构与设计的比较

所有的架构都是设计，但并不是所有的设计都是架构。架构代表塑造一个系统的重要设计决策，这里的重要性通过改变所需要的成本来衡量。(Booch 2009)

换句话说，架构可以看作是战略上的设计，而详细设计可以看作是战术上的设计。

连接活动的箭头不是想表示一个必须遵守的顺序——仅仅是一个比较合适的顺序。例如，基于执行的架构设计，您可能发现必须对需求进行说明，从而最终可能重新回到需求。

现在让我们略微深入地看一下其中的部分活动，以便您可以更好地理解在这个迭代过程中的架构师角色。图 1.2 中显示了组成**定义需求**这个活动的详细任务。这个迭代从**收集利益相关者的需求**这个任务开始，正如这个任务的名称所示，它关注于了解各种各样的利益相关者的需求。这些需求为架构师提供了需要设计系统的范围的一个初始的指示。在**整理常用词汇**这个任务中的术语表完成后，架构师特别感兴趣的就是**定义系统上下文**这个任务，因为它定义了必须与系统交互的外部元素，如最终用户和其他系统。在这个上下文的基础上，**概要说明功能性需求**和**概要说明非功能性需求**这两个任务分别说明功能性需求和非功能性需求。架构师不仅对关键的功能性需求感兴趣，还对系统质量（如性能）和解决方案约束（包括非功能性需求）感兴趣，处理这些非功能性需求通常比处理功能性需求更有挑战性。

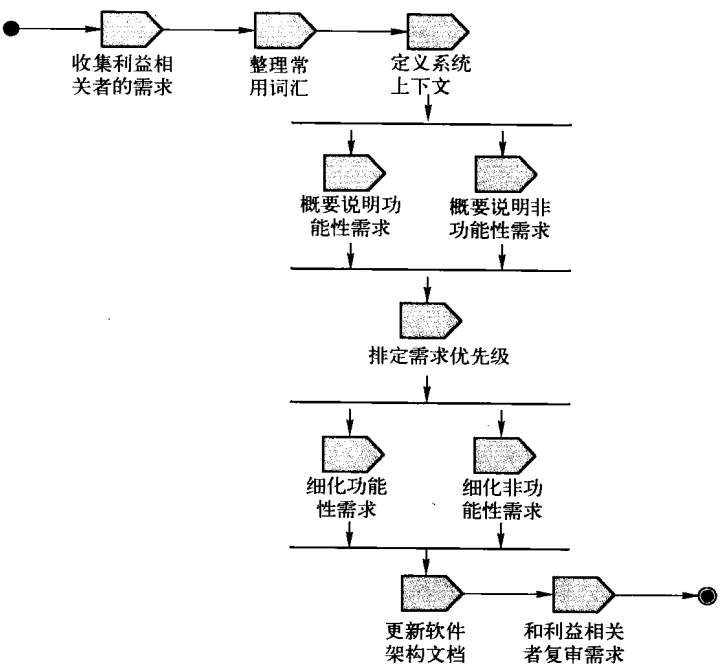

图 1.2　定义需求活动中的任务

从某种程度上，架构师参与整个**定义需求**活动以确保需求能够按通过可用的技术、在指定的时间和预算内就可以实现的方式来指定。这通常要求与利益相关者进行一定程度的沟通，架

构师也要积极地参与进来。就我们关心的需求任务而言，**排定需求优先级**这个任务与架构师关系特别大，在这个任务中，架构师要保证优先级受那些能使架构尽可能快速稳定的需求和风险的影响。

排定了优先级的需求推动了其余内容的迭代，在这种意义上，只有最高优先级的需求会被考虑，因为您处理的需求范围将决定您是否会有一个可行的系统。（低优先级的需求会在随后的迭代中进行考虑。）高优先级的需求会在**细化功能性需求**和**细化非功能性需求**这两个任务中进行细化。接下来，架构师会在**更新软件架构文档**任务中正式地编写架构上重要需求的概要文档。这些与需求相关的任务最终以**和利益相关者复审需求**这个任务结束。就活动来说，连接活动的箭头不是表示一个必须遵守的顺序（仅仅是一个比较合适的顺序），在需要时可以重新回到这些任务。

组成**创建逻辑架构**这个活动的任务如图1.3所示，在**调查架构资源**这个任务中，架构师始终考虑如何使用现有的资源。即使在项目的早期，作为一个架构师，您都可能会选择一个对您的架构有明显影响的资源，如一个参考架构。当执行每个任务时，架构师都在**编写架构决策的文档**这个任务中记录那些决策。

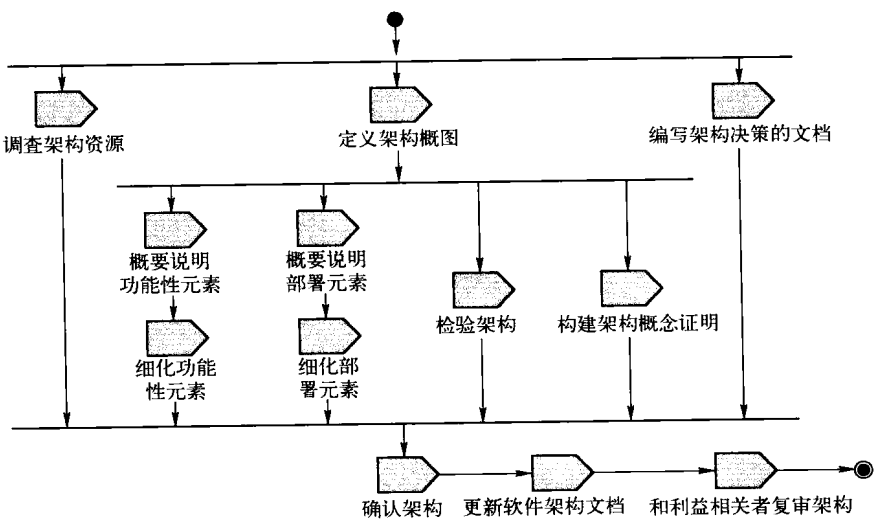

图1.3 创建逻辑架构活动中的任务

基于最高优先级的需求，架构师在**定义架构概图**这个任务中概要说明候选的解决方案。架构概览提供了架构的“整体”轮廓。然后，**概要说明功能性元素**和**概要说明部署元素**这两个任务同时执行，接着根据组件、它们的交互和关系以及它们在节点上的部署精炼这个轮廓。**检验架构**这个任务确保组成架构的各种功能性元素和部署元素相互一致，尤其是确保跨这些元素的任何要点（如既影响功能元素又影响部署元素的性能特性）都被适当地处理。我们承认功能性元素和部署元素未必支配每个架构（例如，并行元素可能会支配一个实时、内嵌的系