HZ BOOKS 华章IT
IBM PRESS

架构师书库

IBM院士、软件工程首席科学家Grady Booch作序鼎力推荐，IBM杰出工程师、首席技术官Tilak Mitra亲笔撰写，Amazon全五星评价，是该领域的第一本完备指南

通过一整套实用的案例研究，逐步讲解系统环境、架构概述、架构决策、功能模型、操作模型、系统设计的集成模式及基础设施

实用软件架构

从系统环境到软件部署

Practical Software Architecture

Moving from System Context to Deployment

[印] 蒂拉克·米特拉（Tilak Mitra）著
爱飞翔 译

机械工业出版社
China Machine Press

架构师书库

Practical Software Architecture

Moving from System Context to Deployment

实用软件架构

从系统环境到软件部署

[印] 蒂拉克·米特拉（Tilak Mitra）著 爱飞翔 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

实用软件架构：从系统环境到软件部署 /（印）蒂拉克·米特拉（Tilak Mitra）著；爱飞翔译．
—北京：机械工业出版社，2016.10
（架构师书库）
书名原文：Practical Software Architecture: Moving from System Context to Deployment

ISBN 978-7-111-55026-6

I. 实… II. ① 蒂… ② 爱… III. 软件设计 IV. TP311.5

中国版本图书馆 CIP 数据核字（2016）第 239245 号

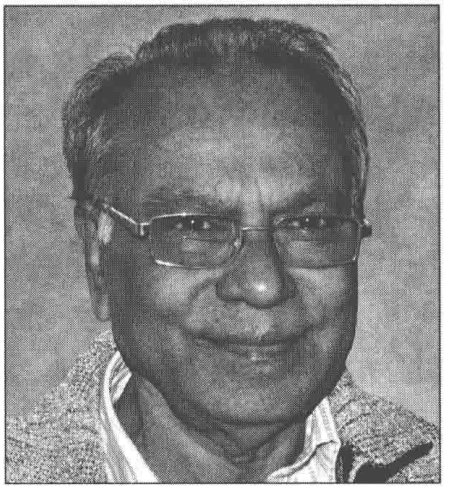
本书版权登记号：图字：01-2016-2696

实用软件架构：从系统环境到软件部署

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：刘诗灏　　责任校对：殷　虹
印　　刷：北京文昌阁彩色印刷有限责任公司　　版　　次：2017 年 1 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：17.25
书　　号：ISBN 978-7-111-55026-6　　定　　价：79.00 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

•• 题献 ••



此书谨献给我过世的父亲 Dibakar Mitra 先生（1940—2015）。2015 年年初，我的父亲离开了我们，我的生命中就此出现了一个悲伤的缺口，我无法自拔，难以接受这个事实。父亲给了我莫大的动力，使我相信自己具有更强大的力量，能够去成就一番事业。作为他的独子，我想让父亲能够因我而骄傲。他的钱夹里装着我的名片，时常在同事和朋友面前夸我（他有时连自己的名片都不带，但总是会带上我的名片）。

就在我成为 IBM 杰出工程师（Distinguished Engineer，DE）的 45 天之前，父亲离开了我们，他是多么想看到我获得这项荣誉啊。我最大的遗憾就是没办法拿起电话告诉他这个消息。他离世前，跟我说的最后一句话是“别担心，你今年肯定会成为 DE 的”。说完这话不久，他就接上了呼吸机。我的家乡，印度的加尔各答，有一家号称医术极好的医院，但就是在这家医院里，顽强求生的父亲最终因为医疗事故离开了我们。我至今依然难掩内心的悲愤之情。

愿父亲安息。我祈求自己能在余生中以某种形式抚慰您的灵魂。儿子永远爱您。

•• 译者序 ••

软件开发工作是由需求引领的，而需求会随着业务的发展逐渐变得庞杂。为了对持续变化的需求进行有效的管理，很多开发者与软件公司都建立了软件架构这样一个概念。尽管软件业与传统的实物产业不同，但它依然可以通过良好的架构指导项目的设计、编码、测试、部署以及维护等诸多阶段。从这个意义上讲，软件架构与其他行业的架构之间有相通的地方。

然而，这并不是一本泛论架构的图书，它谈论的是软件架构，并且尤为关注软件架构中的实际做法。与题材类似的其他书籍相比，本书在核心理念、结构安排以及术语使用等方面都有自己的特色。

本书的核心理念体现在恰到好处（just enough）这个词上。架构固然应该对实现起指导作用，但这个指导作用应该留有一定的余地，使我们可以对架构进行反思，并根据项目的发展情况对其做出调整。软件架构要想做得务实，就需要把握住恰到好处的原则，架构师要知道应该把模块细化到何种程度，才能使开发团队在既定的大方向下灵活地进行发挥。

在结构的安排上，作者首先简介了贯穿全书的 Elixir 项目，后续各章分别用该项目来做案例研究，以演示软件架构的某一个方面，把这些方面拼合起来就可以形成一套完整的案例。把该案例与工作中的实际项目进行对比，或许会对大家有所启发。接下来，作者谈了软件架构的含义和意义，并指出了描述架构所用的几种视点。然后，作者明确提出架构中需要关注的 7 个方面，并且用 7 章的篇幅来详细地进行讲解，使我们明白怎样才能恰到好处地应对这些方面。本书最后给出 3 个专题，分别介绍了大数据时代的分析架构、作者在多年工作中所积累的经验，以及业界经常谈论的 25 个架构话题。

在术语的使用上，作者经常采用两种或三种称呼来指代同一个概念，这不仅反映了该概念所具有的多重意义，而且还使读者感受到一种现象：不同的人在不同的情境下称呼同一个概念时，关注的重点是有所区别的。由于作者就职于 IBM 公司，因此译者

在翻译术语时，优先考虑采用 IBM 网站上已有的译法，对于同时具有多种译法的术语，则酌情采用其中较常见的一种或两种。

纵观全书，作者清晰地描绘了软件架构工作的执行脉络，并指出了从系统环境到软件部署等诸多环节中所应注意的各种问题。需要提醒大家的是，软件架构必须通过适当的代码及硬件配置得以体现，因此，架构应该尽量与实际的编码及测试工作相契合，而且要与后两者保持互动。

在翻译过程中，我得到了机械工业出版社华章公司诸位编辑和工作人员的帮助，在此深表谢意。

由于译者水平有限，错误与疏漏之处，请大家发邮件至 eastarstormlee@gmail.com，或访问 github.com/jeffreybaoshenlee/psa-errata/issues 留言，予以批评指正。

爱飞翔

•• 序 ••

软件架构这个词，有些人听了觉得开心，有些人听了要皱眉头，而更多的人对它漠不关心，尤其是那些整天忙着敲代码，没时间思考设计问题的人。

我们知道，软件密集型的系统都是有架构的。有一些架构是刻意而为的，有一些架构是偶然浮现出来的，还有很多架构隐藏在成千上万个小的设计决策中，而这些设计决策，正源于我们敲出来的那些代码。

Tilak 先生在本书中精彩地讲解了一些切实可行而且非常实用的方式与方法，以帮助我们架构出复杂的系统。作者是一位拥有实际经验的架构师，他通过一系列案例研究，解释了“架构是什么”以及“架构不是什么”这两个问题，同时还讲解了在软件密集型的系统中，如何使架构成为开发、交付及部署过程的一部分。如果大家了解我，那一定知道我对软件架构这个主题有一些强烈的个人观点，然而在我读过的关于这个主题的那么多本书和那么多篇文章中，我确实觉得 Tilak 所说的这套方法是建立在坚实的基础之上的，而且他的方法特别容易理解，也特别容易施行。

软件架构并不是一项纯粹的技术，其中还要考虑人的因素。本书正是抓住了这个重要的因素——Tilak 把自己在架构工作中汲取的经验教训合理地穿插在本书中，我很欣赏这一点。

架构是个重要的过程，这个过程不仅不能妨碍系统的构建，而且还必须在恰当的时机以合适的资源和特别实用的方式构建出正确的系统。

Grady Booch

IBM 院士及软件工程首席科学家

•• 前言 ••

软件架构这个学科已经有半个世纪的历史了。此概念于20世纪60年代引入，它的灵感来源于建筑物的架构，其中涉及在开始盖楼之前拟定的一些蓝图，这些蓝图描述了建筑师对建筑物的结构所制定的设计方案与规格说明。建筑物的蓝图给出了建筑物在功能方面的设计方案，也就是楼层的空间布局示意图，以及每个建筑工件（例如门、窗、房间、浴室、楼梯等）的尺寸。在使建筑物得以运作的那些方面，蓝图也提供了详细的设计方案，例如承载建筑结构的地基、电线、水管和输气管道的设计，以及下水道系统等，要想使建筑物的功能全面运转并发挥效用，这些方面都是不可缺少的。

信息技术（information technology，IT）中的软件架构，其真正灵感来源于建筑架构学中的土木工程（civil engineering）这一学科。据此，我们可以把软件架构大致分成**功能架构**（functional architecture）和**操作架构**（operational architecture）两大类。软件架构在20世纪70年代开始得到大规模实践，到了20世纪90年代，它已经成为IT界的主流，此时各种架构模式也相继涌现。这些模式会随着工作中反复出现的一些用法而演化，所谓反复出现（recurrence），是指这些用法会一直重复地出现在日常应用中。我们之所以能从软件架构中提炼出架构模式，是因为有一个先决条件已经得到了满足。这个条件就是软件架构已经得到了充分的实践，从而成为业界的主流做法，并且已经作为一门正式的研究与实践学科，得到了业界的认可。

IT系统的复杂度越来越高，因此各种IT项目都会频繁而且广泛地运用软件架构技术。软件架构的方式也随着运用面的扩大而变得丰富起来，并且还涌现出了很多流派，它们采用不同的观点来看待软件的架构，并根据其在开发软件系统时所取得的实际经验来总结并推广各自的观点。软件架构的流派和观点变得越来越多，这使很多IT工作者都不知道应该采信哪个流派的观点。大家不妨回想一下，看看自己有没有对下面这些问题表示过困惑？

- 我读过很多架构方面的书籍，也看过很多期刊和杂志，但是我究竟应该怎样把这些互不相同的架构流派汇整起来呢？

- 这些流派中有哪些方面是我比较喜欢的？
- 这些方面是否可以互补？
- 如果我是一名架构师，面对着一个时间和预算都受限制的复杂软件系统，那么应该从哪里开始实现它呢？
- 我是否能成为一名成功的软件架构师？

笔者也曾陷入这样的困惑中。软件架构师所要面对的一项艰难挑战，就是寻找一种最佳的方式，来确定系统或应用程序的架构，并对其进行设计。对软件架构的要义进行把握，既是一种科学，又是一种艺术。我们要用适当的描述语言来定义系统的软件架构，并对其加以分析和理解，从这个层面来看它是科学。同时，我们还要用清晰、明确并且简洁的方式把这个架构描绘出来，以便与不同的利益相关者就系统的解决方案架构进行有效的沟通，从这个层面来看它又是艺术。软件架构师怎样才能抓住关键的架构工件（architecture artifact）[⊖]，从而清晰地描述出整个解决方案呢？这正是难点所在。过度的设计和过多的文档，会拖慢项目的进度，并给项目的交付带来风险，而对软件架构所做的不恰当处理，则会使开发者无法领悟这套架构，这是个很关键的问题。如果开发者不能很好地理解软件的架构，那么他们就无法恰当地遵循技术方面的规范和限制，也无法恰当地使用这套架构来设计并开发系统中的各个部件。在软件开发的整个生命周期中，这个问题只会越来越严重。

2008 年，笔者在 IBM developerWorks 网站上写了一系列专门谈论软件架构的文章。在连续发布 4 部分之后，由于某些个人原因，没有再往下写。接下来的几年，笔者看到了一些网友提出的问题，也收到了一些称赞，然而除此之外，还有另一类信息促使我进行更多的思考。比如，下面这两个问题：

- “先生您好。我正在参考您的系列文章来撰写硕士论文。请问下一部分的文章什么时候发布？”
- “ Mitra 先生，我们采用您所说的框架做了 IT 项目，但是项目暂停了，因为您的下一篇文章还没出来。求助。”

某一天早晨，我忽然感觉读者确实需要一本架构方面的书籍，它必须写得简单、明确、易于理解、便于描述，而且最为重要的是，它必须足够实用，能够执行。这本实用的书籍要能够给 IT 工作者和软件工程专业的学生带来较大的帮助，使他们明白怎样对软件系统进行架构。过了一段时间之后，我终于决定开始写书了。本书代表着软件架构领域中的集体智慧、经验、学问和知识，这些内容是笔者根据自己从业 18 年来的经历收集而成的。本书面对的读者有很多，其中包括：

⊖ 工件也称为制品、产物、成果物。——译者注

- 软件架构师。书中会给出一些实用而且可以反复运用的指导原则，以帮助软件架构师来研发软件的架构。
- 项目经理。本书将会帮助读者理解并领会系统架构中的关键元素，它们是良好的架构所必备的元素，本书还会解释怎样才能在进行项目规划时把架构活动控制得恰到好处。
- 高校学生。本书将会帮助大家理解怎样把软件架构中的理论转述成实际的问题，并对其加以实现。无论技术如何发展，本书都可以当作长期的参考资料。
- 教师。通过本书，教师可以帮助学生把软件架构中的各种理论与实际工作联系起来，使学生变成能够应对实际项目的软件架构师。
- 首席管理者（C-level executive）[⊖]或高层管理人员。本书将会帮助他们意识到研发良好的系统架构所必备的要素，对于 IT 界的任何一种创新活动来说，这种意识都会给公司带来间接的好处，使他们可以更好地领悟 IT 架构在整个公司中的基础地位。

笔者想把这本书写成一本实用的教程，使读者可以按照里面所说的方法，通过多个阶段的演进来迭代式地构建出软件的架构。书中会指出各种架构工件的运用方式，使大家可以把这些清晰、简明、精准而且易懂的工件，恰到好处地运用在实际的应用场景中。在整本书中，笔者会以较为随意的方式来使用“软件”（software）“系统”（system）和“解决方案”（solution）这三个词，由于它们在本书中指的都是架构（architecture），因此这三者之间是可以互换的。笔者之所以要采用这种不拘于字面意思的交替指称方式，是为了使大家明白：在 IT 界，这三个词之间的界限其实是相当模糊的。

从哲学角度来看，东方哲学和西方哲学之间的区别，在于它们对直觉和理性这两种感知形式的接受程度有所不同，前者更强调直觉，而后者更强调理性。西方世界普遍相信，并且主要依赖于理性的、科学的和演绎式的推理。而东方世界则更加看重凭直觉所获取的知识，他们认为，更高形式的意识（在这里指的是知识）是通过观察（也包括反思自己的内心世界）得来的，而不是仅仅通过实验式的归纳得来的。笔者生长于印度加尔各答一个文化较为多元的孟加拉家庭中，十分认同东方式的信仰和知识观念，我认为自觉的意识最终需要通过自觉的自由意志来获得，知识的奥义也要通过直觉和归纳式的推理来领悟。后来，笔者在西方世界生活了将近 20 年，在这段时间里，我开始看重科学和理性的知识形式。我认为，一个普通人要想在这个残酷竞争的世界中生存，就必须掌握由理性与科学手段所得到的知识，对于科学、技术和 IT 领域来说更是如此。等到

⊖ 例如首席执行官（CEO）、首席技术官（CTO）等，由于这些头衔都以字母 C 开头，因此被称为 C 级管理者。——译者注

自己的工作稳定下来之后，可以去深入探索直觉感知力和归纳式推理，这种探索虽然未必会带来回报，但或许会帮助我们从人生的存在中求得解脱。

在这本书中，笔者试着用一种解说的办法，通过归纳式的理性推理来帮助读者掌握实用的软件架构方式。等到掌握了这种理性的知识之后，读者可以把注意力放在归纳式的推理上，以探求更为玄妙的直觉知识。如果把解决最困难的架构问题比喻成寻求圣杯(Holy Grail)，那么用直觉来感知软件的架构就相当于层次更高的开悟了，这种境界，我想应该是大家梦寐以求的吧。

等到看完本书并掌握了它的要义之后，希望你能焕然一新，变成一位务实的软件架构师。软件架构是个有趣的学科，其中的理性知识，我想读完这本书之后，大家应该就可以了解到。而凭直觉才能获得的那一部分知识，则需要以理性知识为基础，继续去探索。在这一方面，连笔者也只是刚刚入门而已。

另外再说一句，每章开头的那些格言，其实都是笔者自己编的。

•• 致谢 ••

首先要感谢妻子 Taneea 和母亲 Manjusree，她们给了我写书所需的时间和灵感。感谢我的叔叔 Abhijit 始终支持我，使我相信自己能够写完这本书。还要感谢我的独子 Aaditya，他总是关心我为什么又要去写一本书。

在专业写作这一方面，我真诚地感谢本书的执行负责人 Ray Harishankar，他从头至尾陪着我走过这段愉快的写作之旅。我还要感谢同事 Ravi Bansal 帮我审阅并完善本书的章节结构，并给我提供相关的专业知识。感谢来自德国的同事 Bertus Eggen，他提出了一个绝妙的数学算法，使我可以针对服务器之间的网络连接度来设计容量模型。感谢 Bertus 允许我在书中使用他的想法。Robert Laird 十分热心地审阅了本书，并提出了相当宝贵的意见，对此我表示衷心的感谢。还要感谢 Craig Trim 给我提供了自然语言处理方面的一些内部细节和技术。

Grady Booch 先生能够为本书作序，令我深感荣幸，在此衷心感谢。

感谢上苍把 Aaditya 赐给我们。2010 年出生的他，给我带来了无尽的快乐，接下来的几年里，我要好好地看着他长大。他已经有了几分“大志”，而且想变成和我一样的人，不过，我还是要引领他，让他更加上进。

•• 目录 ••