MANNING

遗留系统重建实战

Re-Engineering LEGACY SOFTWARE

〔英〕Chris Birchall 著
张喻 张耀丹 禚娴静 译

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

MANNING

遗留系统重建实战

Re-Engineering
LEGACY
SOFTWARE

〔英〕Chris Birchall 著
张喻 张耀丹 禚娴静 译

人民邮电出版社
北京

图书在版编目（CIP）数据

遗留系统重建实战 / (英) 克里斯·伯查尔 (Chris Birchall) 著 ; 张喻, 张耀丹, 禚娴静译. -- 北京 : 人民邮电出版社, 2017.10
书名原文: Re-Engineering Legacy Software
ISBN 978-7-115-46585-6

Ⅰ. ①遗… Ⅱ. ①克… ②张… ③张… ④禚… Ⅲ. ①软件工程 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2017)第212896号

版权声明

◆ 著 [英] Chris Birchall
译 张 喻 张耀丹 禚娴静
责任编辑 杨海玲
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：12.5
字数：262 千字 2017 年 10 月第 1 版
印数：1 – 2 400 册 2017 年 10 月北京第 1 次印刷

著作权合同登记号 图字：01-2016-6800 号

定价：55.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

内容提要

正如本书作者所言，大多数开发人员的主要时间都是花费在与现有的软件打交道上，而不是编写全新的应用程序。相信开发人员或多或少都遇到过与遗留系统相关的问题或者困惑，本书致力于帮开发人员回答这些问题，更重要的是，帮开发人员避免把自己当前开发的系统变成别人将来要面临的遗留问题。

本书篇幅不长，但涵盖的内容很广，例证丰富，有大量的示例代码（主要使用 Java 或 C# 编写），深入浅出地介绍了工作在遗留系统中会遇到的各种问题及应对方法。书中不仅包含技术性的内容——如何选择构建项目的工具，如何自动化构建基础设施，如何决定并进行重构或重写等，也包含非技术性的内容——应该建设什么样的团队文化，如何引入代码评审等活动，如何进行团队知识的传播、改进沟通方式等。

本书面向的读者是有一定经验的开发人员以及项目和技术管理人员。

废墟的召唤

我诅咒废墟，我又寄情废墟。

废墟吞没了我的企盼，我的记忆。片片瓦砾散落在荒草之间，断残的石柱在夕阳下站立，书中的记载，童年的幻想，全在废墟中殒灭。昔日的光荣成了嘲弄，创业的祖辈在寒风中声声咆哮。夜临了，什么没有见过的明月苦笑一下，躲进云层，投给废墟一片阴影。

但是，代代层累并不是历史。废墟是毁灭，是葬送，是诀别，是选择。时间的力量，理应在大地上留下痕迹；岁月的巨轮，理应在车道间辗碎凹凸。没有废墟就无所谓昨天，没有昨天就无所谓今天和明天。废墟是课本，让我们把一门地理读成历史；废墟是过程，人生就是从旧的废墟出发，走向新的废墟。营造之初就想到它今后的凋零，因此废墟是归宿；更新的营造以废墟为基地，因此废墟是起点。废墟是进化的长链。

——余秋雨，《废墟》

软件系统也是如此。回想我自己过去十几年的软件开发工作，曾做过大大小小的项目，有新项目也有几十年的老系统，也曾见证了那些昔日辉煌的新系统如何变成了老旧的遗留系统。就如本书作者所提到的，如果将我过去的工作时间按照读写进行划分，那么绝大多数时间是与别人写的代码打交道，尝试去挖掘当时作者背后的意图，发现其中的蛛丝马迹。而我很多对软件开发的不愉快的回忆也来自这些经历：曾看到大段大段混乱的代码辗转反侧，不得入睡，生怕改错了一行而引起其他未知功能的缺陷；曾对代码中一个几千行方法中某些奇怪的注释——“请勿动这段计算逻辑”而深感无力。

除了代码，还有环境。有一个项目直到我离开都不知如何在本地开发环境搭建好完整的系统，而隔壁的团队戏称他们的系统环境搭建需要至少 3 年的工作经验。

然而，我热爱上软件开发也来自于那段时间的经历。记得当时我维护的一个电信网管系统，其设计精良简单、实现稳定可靠，且容易修改与添加新的功能。其设计思想让我发现了代码的奇妙之处和威力所在，从此点燃了我的软件开发之路，一往直前。

遗留软件是一道美丽艰险的风景线，它离我们并不遥远。昨天的代码即是今天的遗留资产，它诉说的是过去的历史，让人既爱又恨。

本书作者是一位经验丰富的软件开发人员，在本书中，他首先对遗留软件的概念进行了澄清，直面人们对于遗留软件的恐惧与沮丧，从建立度量代码的基础设施入手探索了重新改造遗留软件的切入点。随后，围绕重构从代码级别、模块级别到架构，给出了逐步改善遗留软件的方法，最后，介绍了改善项目的基础设施及工作流程，为我们更好地构建和维护软件奠定了基础。

从整本书来讲，作者给出了改造遗留软件的思路，包括其中一些基础的实践、原则、坏味道以及如何处理，特别提到了代码之外的沟通与文化对遗留软件的影响。从书中的很多内容可以看出，本书是作者多年的经验之谈，虽然没有《修改代码的艺术》那么深，但仍不失为一本实用的处理遗留系统的基础入门书籍，其中第二部分的例子可作为模块重构的参考之一。

这次有幸与我的同事张喻、张耀丹一起完成了本书的翻译。翻译过程中，我们也有诸多的探讨。对我而言，也是一次对遗留软件的系统性学习，受益颇多。其中，让我印象深刻的是作者对于遗留软件乐观积极的态度。就如作者所说，本书是一个开始与战斗的号令，希望能够激发更多的人来探讨和分享更多应对遗留软件的方法和经验，也希望更多正在与遗留软件奋战的程序员朋友们能从中有所收获。

禚娴静

2017 年 7 月

译者简介

张喻，ThoughtWorks 咨询师，热爱技术，热衷编程。目前主要从事后端 API 的开发、部署、维护等相关工作，在整洁代码、敏捷实践和软件开发高效团队方面有丰富的理论和实践经验。

张耀丹，ThoughtWorks 咨询师，曾长期参与大型遗留系统的开发与改进，在 Java 服务器端技术、大型系统架构演进、微服务转型、DevOps 和云计算方面有丰富的经验。

禚娴静，ThoughtWorks 咨询师，乐于知识分享与传播。拥有多年企业和互联网应用的开发实战经验，专注于敏捷实践、软件架构和持续交付领域，在.NET 技术栈和微服务架构演化等方面有丰富的积累。

前言

作为软件开发人员，在我的职业生涯中，写这本书的想法越来越强烈。像许多其他开发人员一样，我花了大部分时间与其他人编写的代码打交道，处理它们引起的各种问题。我想学习和分享关于如何维护软件的知识，但却找不到太多愿意讨论它的人。遗留软件似乎是一个禁忌话题。

我发现这个结果相当令人惊讶，因为大多数的开发人员都将主要的时间花费在了与现有软件打交道上，而不是编写全新的应用程序。然而，看科技博客或书籍时，大多数人都在写关于使用新技术来构建全新的软件。这是可以理解的，因为我们开发者就像喜鹊一样，总是寻找下一个闪亮的新玩具来娱乐自己。同样，我认为人们应该更多地谈论遗留软件，所以本书的一个动机是开启这种讨论。如果读者有任何改进本书的建议，请就其写成一篇博客，并公之于众。

同时，我注意到，很多开发人员已经放弃了任何改进遗留软件和让它更易于维护的尝试。许多人似乎害怕他们维护的代码。所以我还想让这本书成为一个战斗的号令，鼓舞开发人员对他们的遗留代码库负责。

在有了十多年的开发人员经历之后，盘旋在我脑海里的很多想法以及一些零散的笔记，使我有了希望有一天能将这些变成一本书的想法。然后，突然有一天，Manning 出版社与我联系，问我是否想写一本不同的书。我向他们讲述了我的想法，他们表示对此非常感兴趣，之后，我记得我签署了一份合同，然后这本书就成为了现实。

当然，这只是一个漫长旅程的开始。我要感谢每一个帮助过这个项目的、将这个模糊的想法变成一本完整的书的人，没有你们，我自己是无法完成它的！

致谢

本书的完成离不开许多人的支持，我很幸运能够与许多技艺精湛的开发人员一起工作数年，他们为本书间接地贡献了不计其数的想法。

感谢 Infoscience 的每一个人，特别是经理和高级开发人员，他们让我有机会尝试新技术和开发方法。我想我对产品做出了积极的贡献，但我也从其中学到了很多。特别值得一提的是 Rodion Moiseev、Guillaume Nargeot 和 Martin Amirault 的一些了不起的技术讨论。

我也要感谢 M3 项目的每一个人，在那里，我第一次尝试用天而不是月来度量发布周期。我学到了很多，特别是从“老虎” LIoyd Chan 和 Vincent Péricart 那里受益匪浅。也是在 M3 的时候，Yoshinori Teraoka 给我介绍了 Ansible。

现在我在 Guardian，我非常幸运能与这么多有才华有激情的开发人员合作。更重要的是，他们教会了我真正以敏捷方式进行工作的意义，而不仅仅是走过场。

我还要感谢那些花时间阅读本书初稿的审稿人：Bruno Sonnino、Saleem Shafi、Ferdinando Santacroce、Jean-FrançoisMorin、Dave Corun、Brian Hanafee、Francesco Basile、Hamori Zoltan、Andy Kirsch、Lorrie MacKinnon、Christopher Noyes、William E. Wheeler、Gregor Zurowski 和 Sergio Romero。

这本书也收到了来自整个 Manning 编辑团队的很大帮助。策划编辑 Mike Stephens 帮助我把书从想法落实到纸面，我的编辑 Karen Miller 不知疲倦地审阅初稿，我的技术开发编辑 Robert Wenner 和技术校对员 René van den Berg 都做出了不可估量的贡献。Kevin Sullivan、Andy Carroll 和 Mary Piergies 帮助完成了从初稿到最后的生产的过程。还有无数的其他人审查了我的手稿，或者以其他的方式帮助了我，其中一些我可能甚至都不认识！

最后，我要感谢我的妻子 Yoshiko、我的家人、我的朋友 Ewan 和 Tomomi、Nigel 和 Kumiko、Andy 和 Aya、Taka 和 Beni，以及其他在我写作时让我保持清醒的每一个人。尤其是 Nigel，他很棒。

关于本书

本书的设定目标是，教会你将一个被忽视的老旧代码库转变为一个可维护的、功能良好的、能为你的组织产生价值的软件所需要做的一切事情。当然，在一本书里覆盖全部内容是一个不可实现的目标，但是我已经尝试着用多种不同的角度来处理遗留软件的问题了。

代码变成遗留资产（我是指大致上难以维护）的原因有很多，但大多数原因与人有关而非技术。如果人们彼此之间没有足够的沟通，当他们离开组织时，有关代码的信息就可能会丢失。同样，如果开发人员、管理者和组织作为一个整体，并没有正确地划分他们工作的优先级，那么技术债务会累积到不可持续的水平，开发的速度几乎会下降为零。正因为如此，本书将反复涉及组织方面，尤其是重点关注信息随时间流失的问题。仅仅意识到这个问题，便是解决它的重要的第一步。

这并不是说，本书没有技术内容——远远不是。在本书中，我们将会涵盖范围广泛的技术和工具，包括 Jenkins、FindBugs、PMD、Kibana、Gradle、Vagrant、Ansible 和 Fabric。我们将详细讨论一些重构模式，讨论各种架构的相关方法，从单体结构到微服务，以及在重写期间处理数据库的策略。

组织结构

第 1 章是一个入门介绍，解释了我所说的遗留软件的意思。每个人都有自己对于一些术语的定义，如“遗留”（legacy），所以有必要在开始前确认我们是相互理解的。此外，我还谈论了一些促使代码变为遗留资产的因素。

在第 2 章中，我们将使用诸如 Jenkins、FindBugs、PMD、Checkstyle 和 shell 脚本等工具来搭建用于审查代码库质量的基础设施。它们将为你提供可靠的数字数据来描述代码的质量，这些数据很有用，原因有很多。第一，它允许你定义清晰的、可度量的目标以提高质量，这为重构提供了结构。第二，它可以帮你决定应该把工作重点放在代码的什么位置。

第 3 章讨论了如何在开始一个大型的重构项目之前让组织中的每个人都参与进来，并提供一些关于如何解决最困难的决策的建议：重写还是重构？

第 4 章深入重构的细节，介绍了一些我以前经常看到的在遗留代码里成功应用的重构模式。

在第 5 章中，我们将看一下我所说的重搭架构（re-architecting）。这是大范围的重构，即在整个模块或组件的级别，而不是单独的类或方法上重构。我们还将看一个案例研究，讲述如何将单体代码库架构重搭成多个独立的组件，并比较各种应用程序架构，包括单层架构，面向服务架构（service-oriented architecture，SOA）和微服务架构。

第 6 章致力于完全重写一个遗留应用程序。本章涵盖了防止特征蠕变所需的预防措施，现有实现对其替换实现的影响程度以及如果应用程序具有数据库时该如何顺利迁移。

接下来的 3 章没有继续代码的话题，而是探讨了基础设施。在第 7 章中，我们将讨论自动化是如何大大提高新开发人员的入职流程的，这将鼓励团队之外的开发人员做出更多贡献。本章介绍了 Vagrant 和 Ansible 等工具。

在第 8 章中，我们将继续使用 Ansible 进行自动化工作，这一次将其扩展到了预生产（staging）环境和生产环境。

第 9 章通过展示如何使用像 Fabric 和 Jenkins 这样的工具来自动化软件的部署，完成了基础设施自动化的讨论。本章还提供了更新项目工具链的示例，在这种情况下，如何将构建工具从 Ant 迁移到 Gradle。

在第 10 章即本书的最后一章中，我将提供一些可以遵守的简单规则，希望它们能够帮助你防止代码成为遗留资产。

源代码

书中的所有源代码都是等宽字体，以便与周围的文本区分开来。在很多代码清单中，为了指出关键概念，对代码加了评注。还有一些代码清单中，注释被设置在代码中，告诉开发人员在“真实世界”中将看到什么。

我们试图通过添加换行符并仔细使用缩进来格式化代码，使其适合于书中的可用页面空间。

本书中使用的所有代码都可以从 www.manning.com/books/re-engineering-legacy-software 下载，也可以在 GitHub 上（https://github.com/cb372/ReengLegacySoft）找到。

作者在线

购买本书的读者可以免费访问 Manning 出版社运行的私有网络论坛，在那里，可以对本书发表评论，提出技术问题，还可以从作者和其他用户那里获得帮助。要访问论坛并订阅它，只需在 Web 浏览器中访问 www.manning.com/books/re-engineering-legacy-software 即可。此页面提供了一些信息，包括在注册后如何访问论坛，网站提供了什么样的帮助以及论坛上的行为规则。它还提供了本书中示例的源代码、勘误表以及其他下载资源的链接。

Manning 出版社承诺为我们的读者提供一个平台，在这个平台上，不同的读者之间、读者与作者之间可以进行有意义的对话。这个承诺并不包括本书作者的参与度，作者对该论坛的贡献仍然是自愿的（无偿的）。我们建议读者尝试向作者提出具有挑战性的问题，以免作者丧失对论坛的兴趣！

只要本书还在发行，读者就可以在 Manning 出版社的网站上访问作者在线论坛和以前讨论的存档。

作者简介

Chris Birchall 是伦敦《卫报》的一名高级开发人员，致力于为网站提供支持的后端服务。此前，他做过很多不同的项目，包括日本最大的医疗门户网站、高性能日志管理软件、自然语言分析工具和许多移动网站。他拥有剑桥大学计算机科学专业的学士学位。

关于封面插图

本书封面上的插画名为“Le commisaire de police”，即“警察局长”。该插画选自 19 世纪多位艺术家的作品集，由 Louis Curmer 编辑，并于 1841 年在巴黎出版。该作品集的标题是《Les Français peints par eux-mêmes》，翻译过来是“法国人民的自画像”。每幅插画都是精心绘制和手工上色的，作品集中丰富多样的作品向我们生动地展现了 200 年前世界上各个区域、城镇、村庄及居民区的文化是多么迥异。人们彼此分开，讲不同的方言和语言。仅仅通过他们的服饰就能很容易地辨别出他们在哪儿生活，住在城镇还是住在乡村，以及他们的职业或身份。

自那以后，服饰的风格已然发生了变化，当时各地丰富多样的风格已经逐渐消失。现在已经很难分辨出不同大陆的居民，更不用说不同城镇或者地区的居民了。也许我们已经用文化的多样性换取了更加多样化的个人生活——当然也是更加多样化和快节奏的科技生活。

在很难将一本计算机图书与另外一本计算机图书区分开的时代，Manning 出版社借助两个世纪以前的多样化的区域生活的图书封面，让该作品集中的插画重现于世，借以赞美计算机行业的创造力和进取精神。

目录

第一部分

开始

如果你正打算重建（re-engineer）遗留代码库，那么不管它的规模如何，你都需要花费一些时间，提前做些功课，并确保你用正确的方式做事。在本书的第一部分，我们会做很多准备工作，这些都是后面会用到的。

在第 1 章中，我们将研究遗留（legacy）指什么，什么因素会导致不可维护软件的出现。在第 2 章中，我们将建立一个检测的基础设施，使我们能够定量地度量软件的当前状态，并围绕重构提供结构和指导原则。

选择什么工具来度量软件质量完全取决于你，这些工具依赖于很多因素，如你的实现语言和你已经有的开发经验等。在第 2 章中，我将使用 3 种流行的 Java 软件质量工具，即 FindBugs、PMD 和 Checkstyle。我还将展示如何将 Jenkins 设置为持续集成服务器。在书中的不同部分我会再提及 Jenkins 相关的内容。