Broadview
www.broadview.com.cn
Pearson
· 原味精品书系

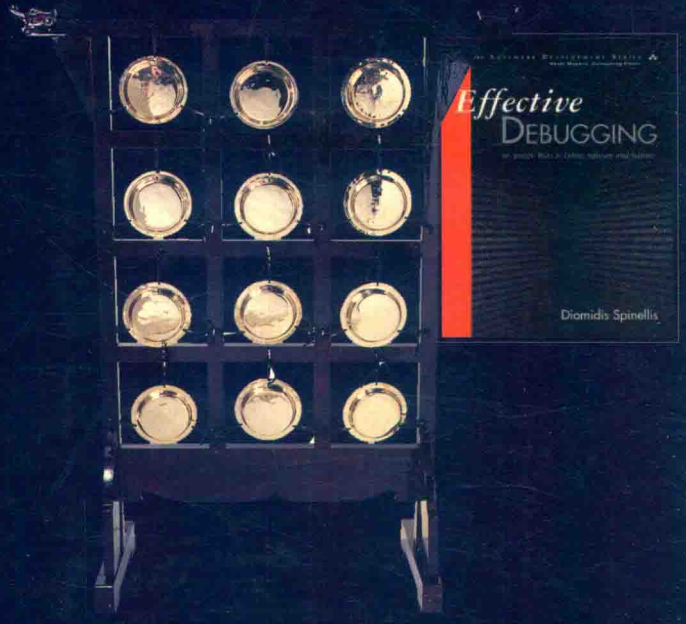
Effective Debugging（英文版）

调试软件与系统的66个有效方法

Effective Debugging

66 Specific Ways to Debug Software and Systems

[希腊] Diomidis Spinellis 著



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

· 原味精品书系 ·

Effective Debugging（英文版）

调试软件与系统的66个有效方法

Effective Debugging

66 Specific Ways to Debug Software and Systems

[希腊] Diomidis Spinellis 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内容简介

本书着眼于一系列可能会在现代软件系统中出现的问题，特别是分散在组件和服务之间，由于复杂的相互影响造成的问题，通过系统地分类，解释并说明最有效的调试方法、策略、技术和工具，来帮助有经验的程序员加快调试进程。无论你是否正在调试独立运行的错误或灾难性的企业系统故障，本书都将帮助你更快、更轻松地完成任务。

版权贸易合同登记号　图字：01-2016-8753

图书在版编目（CIP）数据

Effective Debugging：调试软件与系统的 66 个有效方法 = Effective Debugging: 66 Specific Ways to Debug Software and Systems：英文 /（希）季奥米季斯 · 斯宾奈里斯（Diomidis Spinellis）著. — 北京：电子工业出版社，2017.7
（原味精品书系）
ISBN 978-7-121-31531-2
I. ①E…II. ①季…III. ①调试软件 – 英文 IV. ①TP311.562

中国版本图书馆 CIP 数据核字（2017）第 108512 号

责任编辑：徐津平
印　　刷：北京中新伟业印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　　印张：15.5　　字数：312.5 千字
版　　次：2017 年 7 月第 1 版
印　　次：2017 年 7 月第 1 次印刷
定　　价：75.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
本书咨询联系方式：（010）51260888-819　faq@phei.com.cn。

献给现在及未来的良师益友。

前言

在开发软件或者管理软件系统时，经常会遇到各种故障。这些故障的类型多种多样，一些是可以迅速解决的代码错误，还有一些可能会给公司造成每小时上百万元的损失——如大型系统的宕机。对于以上两种情况，作为一名高效的专业人士，要能够快速识别并修复这些故障。这就是调试的含义，也是本书的主题。

本书并非用于入门，面向的是有经验的开发人员，所以希望你能够理解用多种不同的编程语言编写的小型代码示例，并且可以使用高级图形用户界面和基于命令行的编程工具。但是，本书中仍然包含了详细的调试技术，因为我所见到的一些资深开发人员，即便其熟练地掌握了各种方法，也时不时地需要参考资料。此外，如果你曾经从事过至少几个月的正经调试工作，就能更容易地体会到书中先进的方法。

本书包含的内容

如本书主题，调试工作指的是在开发和运行一个现代化、精密化的计算系统时所用到的策略、工具和方法。在以前，调试主要指的是检测和修正程序错误。然而，现在的程序很少独立运行。即使最小的程序也常常与外部库动态链接。复杂的程序会在应用服务器上运行、调用网络服务、使用关系型数据库或者 NoSQL 数据库、从目录服务器上获取数据、运行外部程序、利用中间应用程序并集成大量的第三方软件包。完整的系统和服务的运行依赖于散布在全球主机上的无故障运行、内部开发的第三方组件。软件开发原则 DevOps 能够解决这一现实问题，它强调了开发人员和其他 IT 专业人士的角色。本书旨在帮助你全面看待程序错误，因为对于最具挑战性的难题，通常很难立即确定软件中的罪魁祸首。

本书从一般性话题出发，引出具体的问题。第 1 章的内容是“策略”，第 2 章的内容是“方法”，第 3 章介绍了可以调试不同软件和系统故障的工具和技术。之后，书中还列举了针对调试工作不同阶段的技术，如使用调试器的阶段（第 4 章）、编程阶段（第 5 章）、编译软件阶段（第 6 章），以及运行系统阶段（第 7 章）。关于多线程和并发代码中那些麻烦的程序错误，第 8 章的内容单独涵盖了专门的工具和技术。

如何使用本书

一般情况下，读书都是从头看到尾。但实际上，还有更好的方法。本书中的建议分为以下三类。

第一，在面对程序错误时要用到的**策略和方法**。这方面的内容在第 1 章和第 2 章中都有罗列。此外，第 5 章中的许多技巧也可归于这一类别。多了解其中的内容，在应用时就可手到擒来。在调试过程中，整体性的思想会反映到你所使用的方法上。遇到“死胡同”时，回想一下已知路径，一般能帮助你找到走出迷宫的新路径。

第二，可以寄予希望的**技术和工具**。这些内容主要集中在第 3 章，其中也包含了解决日常问题的各个要素。具体参见“第 36 条：调整你的调试程序”。你需要找时间学习并逐步把这些条目运用到实践当中。这可能意味着放弃自己熟悉的工具，重新学习更先进的工具。刚开始时，这种经历可能会很痛苦，但是从长远来看，它将会把你塑造成为一名卓越大师。

第三，在面对困难问题时的**技术应用思考**。这些技术和方法都不会是你常常使用的，但是在遇到一个无从下手的难题时，它们能够节省你一天（或至少几个小时）的时间。举个例子，如果你无法解释为什么 C 代码或 C++ 代码编译失败，你就可以参阅“第 50 条：检查生成代码”。在快速阅读这些条目时，你可以把它们作为参考，在之后的应用过程中，再仔细甄别。

如何投身编程世界

尽管书中的所有条目都是对诊断错误和调试已有错误提供建议，但是你仍可以利用从这里学到的大部分知识来尽量减少遇到软件故障的情况，也可以有能力解决突发状况，让生活变得更简单。严格的调试和软件开发实践相互促进，这是一个良性循环。这条建议对你现在或将来的软件开发、设计和管理都大有裨益。

在**软件设计**中，请遵循以下建议。

- 使用最合适的最高级机制（第 47 条：考虑用另一种语言重写可疑代码，第 66 条：考虑用更高层抽象进行重写）。
- 提供调试模式（第 6 条：使用软件的调试功能，第 40 条：添加调试功能选项）。
- 提供监控和记录系统运行的机制（第 27 条：对系统各个独立进程进行监视，第 41 条：添加日志语句，第 56 条：检查应用程序日志文件）。
- 引入 UNIX 命令行工具脚本组件选项（第 22 条：使用 UNIX 命令行工具分析调试数据）。

- 确保是由于内部错误引发的明显故障，而非系统不稳定（第 55 条：快速故障）。
- 提供一种方法来获取事后问题分析所需的内存转储（第 35 条：知道如何利用磁心信息转储，第 60 条：使用后期调试分析死锁）。
- 缩小软件执行中的问题源头和非确定性范围（第 63 条：隔离与移除非确定性）。

在**软件构建**中，请遵循以下建议。

- 从同事那里获取反馈（第 39 条：与同事检查并推理代码）。
- 给每个程序创建单元测试（第 42 条：使用单元测试）。
- 使用断言（assertion）来验证你的假设和代码是否能够正确运行（第 43 条：使用断言）。
- 尽量编写可维护的代码，即可读、稳定、易于分析和修改的代码（第 46 条：简化可疑代码，第 48 条：提高可疑代码的可读性和结构性）。
- 避免构建中存在的非确定性源（第 52 条：配置确定性构建和执行）。

最后，无论是团队还是个人管理软件的开发和运行，需要遵循以下几点。

- 记录问题，并使用合适的系统进行检验（第 1 条：使用问题追踪系统来处理所有问题）。
- 分流并给所有问题划分优先级（第 8 条：专注于最重要的问题）。
- 在维护良好的修订控制系统中记录软件的所有更改（第 26 条：使用修订控制系统记录成因和调试历史）。
- 采用渐进方式部署软件，以便对新老版本进行对比（第 5 条：找出已验证系统和失败系统间的差异）。
- 力求所用工具和部署环境的多样性（第 7 条：为软件构建和执行环境拓展多样性）。
- 定期更新工具和库（第 14 条：考虑更新软件）。
- 购买所用的第三方库的源代码（第 15 条：参阅第三方源代码，深入理解第三方应用），同时，购买更高级的工具来诊断难以察觉的错误（第 51 条：使用静态程序分析，第 59 条：使用动态程序分析，第 62 条：运用专门工具解开死锁和竞态条件，第 64 条：调查源于争用的可扩展性问题，第 65 条：使用性能计数器共享 定位错误）。
- 对任何硬件界面和嵌入式系统的调试提供专业工具包（第 16 条：使用专业的监测和测试设备）。
- 使开发人员能够远程调试软件（第 18 条：从桌面开启对难以移动的系统的调试工作）。
- 为高要求的排查任务提供性能充足的 CPU 和磁盘资源（第 19 条：自动调试任务）。

- 在代码检查和指导方面鼓励开发人员的协作（第 39 条：与同事检查并推理代码）。
- 鼓励推动使用测试驱动开发（第 42 条：使用单元测试）。
- 纳入软件构建的性能概况、静态分析和动态分析，同时保持快速、精益、平均的构建和测试周期（第 11 条：最小化从修改到结果的换向时间，第 51 条：使用静态程序分析，第 53 条：配置使用调试库和检测，第 57 条：廓线系统和运作流程，第 59 条：使用动态程序分析工具）。

关于术语的一些说明

本书中，在 ISO-24765-2010（系统和软件工程——词汇）定义范围内使用 *fault* 一词："计算机程序中一个不正确的步骤、进程或者数据定义"。这常被称为 *defect*。在日常语言中，就是通常所说的 *bug*。同样地，我所使用的术语 *failure* 是根据同一标准："系统或系统组件不能在规定限度内实现应用功能"。故障的表现有程序崩溃、锁死，或给出错误的结果。因此，在有 *fault* 的情况下，就会出现 *failure*。令人困惑的是，有时 *fault* 和 *defect* 也会用来指代 *failure*，ISO 标准已经意识到了这个问题。本书中，我会遵守这一区别。然而，为了避免把内容变成法律条文那样，在上下文很清楚的情况下，我通常用 *problem* 一词来指代 *fault*（即代码中的问题）或 *failure*（即可复现的问题）。

如今，UNIX 操作系统的 shell、库和工具可以在许多平台上使用。本书使用 *UNIX* 来指代任何遵循 UNIX 原则和 API 的系统，其中包括苹果的 *Mac OS X*、*GNU/Linux* 的各种发行版（例如，Arch Linux、CentOS、Debian、Fedora、openSUSE、红帽企业版 Linux、Slackware 和 Ubuntu）、UNIX 操作系统的各个分支（如 *AIX*、*HP-UX*、*Solaris*）、各种 BSD 衍生（如 *FreeBSD*、*OpenBSD*、*NetBSD*），以及在 *Windows* 上运行的 *Cygwin*。

与之类似，在编写 C++、Java 或 Python 程序时，使用了一种较为现代的语言版本。我在本书中已经尽力规避了那些过于偏颇或过于前沿的例子。

书中所述的"你的代码（your code）"和"你的软件（your software）"指代你所调试的代码和你正在处理的软件。这样说更简便，而且也暗含了所有者的含义，这在开发工作中是非常重要的。

使用 *routine* 一词来指代代码调用单元，如成员函数、方法、函数、过程和子程序。

使用 *Visual Studio* 和 *Windows* 来指代相应的微软产品。

使用版本控制系统（*revision control system* 和 *version control system*）来指代用于软件配置管理的一些工具，如 Git。

排版与其他惯例

- 代码是用打字机字体（typewriter font）来书写的，关键点设置为粗体（**bold**），条款和工具名称用斜体设置（*italics*）。
- 交互会话清单使用灰度来区分提示、用户输入和产生的输出结果。

```
$ echo hello, world
Hello world
```

- UNIX 命令行选项类似 `--this`，或者使用单字符缩写，如 `-t`。与之对应的 Windows 工具选项则是 `/this`。
- 关键快捷键以 Shift-F11 键这种形式进行设定。
- 文件路径的形式为：/etc/motd。
- 网络超链接的设置形式为 *like this*。如果你在纸质媒体上阅读，你可以在“Web Resources(网络资源)”附录中找到 URL 链接。如果你在电子媒体上阅读，你知道该如何做。
- 菜单导航显示如下：*Debug–New Breakpoint–Break at Function*（调试 – 新建断点 – 函数断点）。
- 为了简洁起见，C++ 代码中省略了 `std::`。
- 在讲到 GUI 工具时，指的是在写作本书时可获取到的最新版本的功能。如果你使用的版本不同，请查阅相关的菜单或窗口，或查询如何访问此功能的文档。有趣的是，命令行工具的界面几十年来一直保持稳定，而图形用户界面的每个新版本都会把功能模块移来移去。可以从这一观察得出的很多结论都作为练习留给了读者。

何处获取代码和勘误

本书例子中出现的代码和正文勘误可通过本书网站 www.spinellis.gr/debugging 获取。

请在 informit.com 上注册你所购买的 *Effective Debugging*，这样可以更便捷地获得下载资源、更新和修订。注册步骤：首先登录 informit.com/register，然后登录或创建账户。接着，输入 ISBN 编码(9780134394794)并单击提交。在完成以上步骤之后，你就可以找到“Registered Products（注册产品）”下提供的所有增值内容了。

致谢

首先，我要感谢 Addison-Wesley 出版社负责本书的编辑 Trina Fletcher MacDonald 和负责这套丛书的编辑 Scott Meyers，感谢他们对本书出版给予的专业指导和管理。我还要感谢本书的技术评审 Dimitris Andreadis、Kevlin Henney、John Pagonis 和 George Thiruvathukal，他们提供了数百项一流的主意、评论和建议来完善本书。我要特别感谢的是本书的文字编辑 Stephanie Geels，她有一双犀利的眼睛和如轻羽般的笔触。感谢她高质量的工作，我才能享受这令人生畏的过程。我还要向 Melissa Panagos 出色高效的过程管理致谢，向 Julie Nahil 对整个生产过程的监督致谢，向 LaTeX 魔术师 Lori Hughes 对本书的排版致谢，向 Sheri Replin 的编辑建议致谢，向 Olivia Basegio 对本书技术审核委员会的管理致谢，向 Chuti Prasertsith 绝妙的图书封面设计致谢，向 Stephane Nakib 的市场营销建议致谢。也向 Alfredo Benso、Georgios Gousios 和 Panagiotis Louridas 对本书的早期指导表示感谢。

以下四项扩展自我发表在 ***IEEE Software*** 的 Tools of the Trade 专栏的相关文章。

- “第 5 条：找出已验证系统和失败系统间的差异”——Differential Debugging，第 30 卷，第 5 期，2013 年，第 19~21 页。
- “第 22 条：使用 UNIX 命令行工具分析调试数据”——Working with Unix Tools，第 22 卷，第 6 期，2005 年，第 9~11 页。
- “第 58 条：追踪代码的执行”——I Spy，第 24 卷，第 2 期，2007 年，第 16、17 页。
- “第 66 条：考虑用更高层抽象进行重写”——Faking It，第 28 卷，第 5 期，2011 年，第 95、96 页。

除此以外。

- “第 63 条：隔离与移除非确定性”，源自于 Martin Fowler 的文章 Eradicating Non-Determinism in Tests（2011 年 4 月 14 日）和 TestDouble（2006 年 1 月 17 日）。
- “第 48 条：提高可疑代码的可读性和结构性”中大部分的重构建议，来源于 Martin Fowler 的著作 ***Refactoring***（Addison-Wesley 出版社，1999 年）。
- “第 60 条：使用后期调试方法分析死锁”，其灵感源自于 Bryan Cantrill 和 Jeff Bonwick 在 2008 年 10 月的 ***ACM Queue*** 上发表的 Real World Concurrency。
- “第 66 条：考虑用更高层抽象进行重写”，其中的 Java 代码由 Tagir Valeev 提供。

雅典经济与商业大学的许多同仁向这一项目提供了很多无私的帮助，有的帮助甚至是不为我所知的。他们是：Damianos Chatziantoniou、Georgios Doukidis、Konstantine Gatsios、George Giaglis、Emmanouil Giakoumakis、Dimitris Gritzalis、George Lekakos、Panagiotis Louridas、Katerina Paramatari、Nancy Pouloudi、Angeliki Poulymenakou、Georgios Siomkos、Spyros Spyrou 及 Christos Tarantilis。

调试是一项技艺，要通过学习才能获得。借此机会，我要向在过去四十余年中那些一直忍受我的程序错误的同事们表示感谢，他们向我提供了很多有帮助的反馈报告，对我的代码进行了审阅和检查，并且教给我如何避免、搜寻和测试故障。按我的工作和合作项目的时间顺序来看，我要特别感谢以下各位。

- Google 的 Ads SRE FE 团队：Mark Bean、Carl Crous、Alexandru-Nicolae Dimitriu、Fede Heinz、Lex Holt、Thomas Hunger、Thomas Koeppe、Jonathan Lange、David Leadbeater、Anthony Lenton、Sven Marnach、Lino Mastrodomenico、Trevor Mattson-Hamilton、Philip Mulcahy、Wolfram Pfeiffer、Martin Stjernholm、Stuart Taylor、Stephen Thorne、Steven Thurgood、Nicola Worthington。
- CQS：Theodoros Evgeniou、Vaggelis Kapartzianis、Nick Nassuphis。
- 雅典经济与商业大学管理科学及技术系，现在和之前的研究项目以及实验室的合作伙伴：Achilleas Anagnostopoulos、Stefanos Androutsellis-Theotokis、Konstantinos Chorianopoulos、Marios Fragkoulis、Vaggelis Giannikas、Georgios Gousios、Stavros Grigorakakis、Vassilios Karakoidas、Maria Kechagia、Christos Lazaris、Dimitris Mitropoulos、Christos Oikonomou、Tushar Sharma、Sofoklis Stouraitis、Konstantinos Stroggylos、Vaso Tangalaki、Stavros Trihias、Vasileios Vlachos、Giorgos Zouganelis。
- 在希腊财政部信息系统秘书处期间：Costas Balatos、Leonidas Bogiatzis、Paraskevi Chatzimitakou、Christos Coborozos、Yannis Dimas、Dimitris Dimitriadis、Areti Drakaki、Nikolaos Drosos、Krystallia Drystella、Maria Eleftheriadou、Stamatis Ezovalis、Katerina Frantzeskaki、Voula Hamilou、Anna Hondroudaki、Yannis Ioannidis、Christos K. K. Loverdos、Ifigeneia Kalampokidou、Nikos Kalatzis、Lazaros Kaplanoglou、Aggelos Karvounis、Sofia Katri、Xristos Kazis、Dionysis Kefallinos、Isaac Kokkinidis、Georgios Kotsakis、Giorgos Koundourakis、Panagiotis Kranidiotis、Yannis Kyriakopoulos、Odyseas Kyriakopoylos、Georgios Laskaridis、Panagiotis Lazaridis、Nana Leisou、Ioanna Livadioti、Aggeliki Lykoudi、Asimina Manta、Maria Maravelaki、Chara Mavridou、Sofia Mavropoulou、Michail Michalopoulos、Pantelis Nasikas、Thodoros Pagtzis、Angeliki Panayiotaki、Christos Papadoulis、Vasilis Papafotinos、Ioannis Perakis、Kanto Petri、

Andreas Pipis、Nicos Psarrakis、Marianthi Psoma、Odyseas Pyrovolakis、Tasos Sagris、Apostolos Schizas、Sophie Sehperides、Marinos Sigalas、George Stamoulis、Antonis Strikis、Andreas Svolos、Charis Theocharis、Adrianos Trigas、Dimitris Tsakiris、Niki Tsouma、Maria Tzafalia、Vasiliki Tzovla、Dimitris Vafiadis、Achilleas Vemos、Ioannis Vlachos、Giannis Zervas、Thanasis Zervopoulos。

- FreeBSD 项目：John Baldwin、Wilko Bulte、Martin Cracauser、Pawel Jakub Dawidek、Ceri Davies、Brooks Davis、Ruslan Ermilov、Bruce Evans、Brian Fundakowski Feldman、Pedro Giffuni、John-Mark Gurney、Carl Johan Gustavsson、Konrad Jankowski、Poul-Henning Kamp、Kris Kennaway、Giorgos Keramidas、Boris Kovalenko、Max Laier、Nate Lawson、Sam Leffler、Alexander Leidinger、Xin Li、Scott Long、M. Warner Losh、Bruce A. Mah、David Malone、Mark Murray、Simon L. Nielsen、David O'Brien、Johann 'Myrkraverk' Oskarsson、Colin Percival、Alfred Perlstein、Wes Peters、Tom Rhodes、Luigi Rizzo、Larry Rosenman、Jens Schweikhardt、Ken Smith、Dag-Erling Smørgrav、Murray Stokely、Marius Strobl、Ivan Voras、Robert Watson、Peter Wemm、Garrett Wollman。
- LH Software 和 SENA 公司：Katerina Aravantinou、Michalis Belivanakis、Polina Biraki、Dimitris Charamidopoulos、Lili Charamidopoulou、Angelos Charitsis、Giorgos Chatzimichalis、Nikos Christopoulos、Christina Dara、Dejan Dimitrijevic、Fania Dorkofyki、Nikos Doukas、Lefteris Georgalas、Sotiris Gerodianos、Vasilis Giannakos、Christos Gkologiannis、Anthi Kalyvioti、Ersi Karanasou、Antonis Konomos、Isidoros Kouvelas、George Kyriazis、Marina Liapati、Spyros Livieratos、Sofia Livieratou、Panagiotis Louridas、Mairi Mandali、Andreas Massouras、Michalis Mastorantonakis、Natalia Miliou、Spyros Molfetas、Katerina Moutogianni、Dimitris Nellas、Giannis Ntontos、Christos Oikonomou、Nikos Panousis、Vasilis Paparizos、Tasos Papas、Alexandros Pappas、Kantia Printezi、Marios Salteris、Argyro Stamati、Takis Theofanopoulos、Dimitris Tolis、Froso Topali、Takis Tragakis、Savvas Triantafyllou、Periklis Tsahageas、Nikos Tsagkaris、Apostolis Tsigkros、Giorgos Tzamalis、Giannis Vlachogiannis。
- 欧洲计算工业研究中心（ECRC）：Mireille Ducassé、Anna-Maria Emde、Alexander Herold、Paul Martin、Dave Morton。
- 伦敦帝国理工学院的计算机科学系：Vasilis Capoyleas、Mark Dawson、Sophia Drossopoulou、Kostis Dryllerakis、Dave Edmondson、Susan Eisenbach、Filippos Frangulis Anastasios Hadjicocolis、Paul Kelly、Stephen J. Lacey、Phil Male、Lee M. J. McLoughlin、Stuart

McRobert、Mixalis Melachrinidis、Jan-Simon Pendry、Mark Taylor、Periklis Tsahageas、Duncan White。

- 加州伯克利大学的计算机科学研究组（CSRG）：Keith Bostic。
- Pouliadis & Associates 公司：Alexis Anastasiou、Constantine Dokolas、Noel Koutlis、Dimitrios Krassopoulos、George Kyriazis、Giannis Marakis、Athanasios Pouliadis。
- 在零零散散的各种会议和场合中：Yiorgos Adamopoulos、Dimitris Andreadis、Yannis Corovesis、Alexander Couloumbis、John Ioannidis、Dimitrios Kalogeras、Panagiotis Kanavos、Theodoros Karounos、Faidon Liampotis、Elias Papavassilopoulos、Vassilis Prevelakis、Stelios Sartzetakis、Achilles Voliotis、Alexios Zavras。

最后，要感谢我的家人，多年来大度地忍受我在家中不断地调试系统以支持本书的写作，有时我甚至挤占了休假和周末的时间。特别要感谢 Dionysis 制作了图 5.2，以及 Eliza 和 Eleana 一起帮忙挑选封面。

作者简介

Diomidis Spinellis 是雅典经济与商业大学科技管理学院教授。他的研究领域包括软件工程、IT 安全以及云系统工程。他的著作《代码阅读》（*Code Reading: The Open Source Perspective*）以及《代码质量》（*Code Quality: The Open Source Perspective*）双双荣获 Jolt（软件开发生产力）大奖并被广泛传译。Spinellis 博士曾在多个学术期刊以及期刊会议论文集中发表了 200 多篇技术论文，被引用次数高达 2500 多次。十年来，他作为 *IEEE Software* 编委会的一员，为 Tools of the Trade 专栏定期撰稿。他为 OS X 以及 BSD UNIX 贡献了很多代码，并且是 *UMLGraph*、*CScout*，以及其他软件开源包、库和工具的开发者。他拥有伦敦帝国理工学院软件工程专业的硕士学位及计算机科学专业的博士学位。他是 ACM 以及 IEEE 的高级会员，并且自 2015 年以来一直担任 *IEEE Software* 的主编。

目录