<<代码质量>>

图书基本信息

前言

编程和其他事情一样，错误即重生。

--Alan J. Perlis　　我希望自己能在前言中这样写道：你手边的这本书是精心策划而成的出版结果，其始于《代码阅读》，而终于《代码质量》。

然而，这么做是对事实的一种歪曲，也是对我们这些工程师所喜爱的有序世界的一种媚从。

其实真相是《代码质量》大体上源自一系列的偶然事件。

签下出版《代码阅读》一书的合同时，我手头已有了一个纲要和一些完成的章节。

基于这些完成章节的长度和所耗精力，我天真错误地估计了全书长度和成书日程。

若你靠编写软件为生，可能已猜到，在书稿应完成之时，我仅仅完成了大纲章节的一半多一点，并已用完所有分拨篇幅。

为体面脱身，我建议编辑将已完成的部分（除去关于可移植性的一章）作为《代码阅读》的第1卷出版，而将剩下的工作作为第2卷。

我们达成了一致，于是《代码阅读》[Spi03a]出版。

结果该书好评如潮，并获得了2004年度软件开发杂志生产效率大奖（The 2004 Software Development Magazine Productivity Awards），还被译为其他6种语言。

在《代码阅读》一书中，我以取自实际工作中的开源项目为例，尝试覆盖软件开发人员可能面对的大多数与代码相关的概念，包括程序结构、数据类型、数据结构、控制流、项目组织、编码标准、文档及架构。

在第2卷中，我先前计划涉及接口与面向应用的代码，包括国际化与可移植性问题、常用库与操作系统要素、底层代码、领域特定语言与声明性语言、脚本语言及混合语言系统。

但当程序员们拿到《代码阅读》一书后，我从他们那里获得了一些有益的看法。

他们给我的反馈表明许多读者都在焦急地等待着下一卷，不过他们期望的并不是一个对于设备驱动程序的详细剖析（我之前为接下来的一卷保留的一章）。

2003年7月，时任编辑Mike Hendrickson建议我撰写一本名为Secure Code Reading的书。

虽然作为一个科学家，IT安全是个让我感兴趣的领域，但是我懒于追逐写作安全方面书籍的时尚，故而仅写了一个与安全有关的章节。

在有了一个与可移植性相关的章节和一个与安全相关的章节后，我突然找到了新书的书名和主题--《代码质量》，其应将重点放于阅读和编写软件代码及软件代码的质量属性上（这些也时常被称为非功能属性）。

借助阅读软件系统的代码，我们可以发现非功能属性与产品的非功能需求相关。

该需求不与系统提供的特定功能直接相关，而是与更广泛的重要系统属性有一定的关系。

常见的与系统属性相关的非功能属性为：可靠性、可移植性、易用性、互用性、适应性、相依性及可维护性。

另外两个与系统的效率有关的值得注意的非功能属性是：系统与时间约束有关的性能和系统空间需求。

借助阅读代码来了解系统非功能属性是开发者必不可少的一项技能，这里面有两个原因。

一是未满足非功能需求所引发的结果可能是危险的，甚至是灾难性的。

一个系统若仅某些功能需求出现错误（大多数软件系统都含有这样的错误），则其还能以降级模式运行，用户也会被告知避免使用其中的一些功能。

但若非功能需求出现错误则较为严重：不安全的Web服务器或不可靠的防抱死系统（ABS）比无法使用更糟糕。

二是非功能需求有时难于验证。

我们写不出这样一个测试用例使得它能够验证系统的可靠性或系统是否存在安全漏洞。

基于以上两点，我们在应对非功能需求及相关软件属性时，需利用我们所能利用的一切手段。

将代码与非功能属性联系起来的能力应是每一位软件工程师都需具备的有力武器。

除去视角不同外，《代码质量》沿用了《代码阅读》的成功秘诀，即重点关注已有代码的阅读；

仅分析取自现有开源系统的实例；给出所有实例的来源；采用注释来剖析代码；提供有意义的练习来强化读者的批判能力和技巧；在页边空白处指出编码习惯及陷阱；以箴言录的形式总结每章的建议；在“延伸阅读”部分，给之前的实践以理论的拓展；采用了UML（Unified Modeling Language）来绘制所有简图。
其中最为精妙的元素就是我自行强加的规则--避免使用“玩具”示例（toy example），所有的用例都须取自现有开源项目。
为践行该规则，我常常花费数小时来寻找一个合适的例子，它应当既能表明我欲展示的概念，又要易于理解，并且还需足够短小以便可放至一本书中。
个人认为这样的活动既有助于智力同时也可使作品更为严谨。
通常在寻找某方面的缺点时，我会碰到其他值得讨论的内容。
有时我对于某理论概念示例的寻找没有什么结果，那么在此情形下，我就有理由认为在实践中这些概念并未重要到必须将其囊括于本书的地步。
　　《代码质量》和《代码阅读》背后的原理与动机是一脉相承的--阅读代码可能是计算机专业人员最常见的活动之一，不过其却较少被当做一门课程来教授或被正式当做一种学习程序设计和编码的方法来使用。
所幸开源软件的盛行已向人们提供了大量可供自由阅读和学习的代码。
基于开源软件的初级读本是提高个人编程能力的有价值的工具。
所以我希望这两本书能够引发人们在计算机教育课程中加入代码阅读的课程、活动及练习的热情。
如此一来，在不久的将来，就像是从优秀的文学作品中学习写作一般，我们的学生就可从现有的开源系统中学习编程了。
　　内容及补充材料　　我决定在《代码质量》中沿用同《代码阅读》中一样的系统和分发版本作为开源代码样例。
原因是我认为两卷间必须存在一定的延续性，这样方可允许读者观察到同样的源代码如何既能在《代码阅读》中被用于认识其所涵盖的功能、架构及设计等功能特性，又能被用于《代码质量》中所涉及的非功能特性。
　　本书所用的代码来自于那些大多数仅有历史意义的代码快照。
然而有了它们，我就可以借机展示那些在较新版本中已被发现并更正的安全漏洞、同步问题、可移植性问题、API调用的误用及其他bug。
这些许久之前的代码可能表明，其作者目前或已晋升至管理岗位以致不喜于阅读此类书籍，或视力退化至无法看清书中字体。
我可借助这些变化来自由品评相关代码，而不用担心险恶的报复。
虽如此，我还是可能被指责诽谤那些怀有推进开源运动信念的作者所贡献之代码，我对此表示理解，同时也认同我们应当改进这些代码而非仅仅对其评论。
若我的评论使某些源代码的作者感到冒犯，请允许我预先在这里真诚的道歉。
不过作为对自己言行的辩护，我个人认为在大多数实例中，我的评论并非针对特定的代码片段而是借助其来指明开发者在实践中应避免的行为。
通常情况下，被我作为反例来使用的代码都是易受此类伤害的对象，原因在于它们被编写时，技术及其他方面的限制使得其也算是一种合理的行为，抑或我的一些特定的批评其实脱离了代码所处上下文。
无论何种情形，我都希望这些评论能得到你的善意一笑，同时我也坦率承认，我本人的代码也包含有类似的，甚至可能更糟的错误。
　　选择本书示例所用系统时，出于实用的原因，我希望这些代码能够适于作为教学载体。
故而我所寻找的内容涵盖代码质量、结构、设计、功用、流行度及无版权问题。
我尽可能平衡语言的选择，积极寻找合适的Java和C++代码。
然而，当类似概念可被多种语言演示时，我选择采用C作为“最小公分母”。
所以，本书中引用的61%的代码为C代码，这其中包括适用所有语言的实例及系统级编程（多用C语言）实例。

另外19%的实例引用了Java代码。
选择Java代码是为了展示与面向对象概念及相关API有关的元素。
这些概念多也适用于C#，并且许多也适用于C++（4%的实例引用了C++代码）。
　　此外，在本书中我也更为强调UNIX中的API和工具而非Windows中的类似元素。
原因依然涉及“最小公分母”逻辑--许多UNIX中的工具和API在Windows中也有对应元素，而反之则不然。
大量与UNIX相兼容的系统，如GNU/Linux与BSD的一些变体，都是免费提供的，并且常常还向使用者提供可引导的CD-ROM，如此一来任何人都可轻松体验这样的系统。
最后，UNIX中的API和工具，从我所引示例的细节方面来看，在过去30余年来一直保持着惊人的稳定，这就为我们讨论和展示通用规则提供了一个坚实的根基。
虽然如此，我还是在一些地方引用了Windows API及命令来讨论非UNIX平台上的情况。
但是请别让这些引用欺骗了你--我并不认为本书涉及Windows平台的编程问题要比涉及UNIX平台的相关问题完整和广泛。
　　除去在所有实例中均采用开源软件外，本书可能被（狭隘地）指为忽略了大量的流行时尚，如Java、C#、Windows、Linux及以解决现有问题为导向的写作风格。
其实我很看重前述这些元素：我这里29%的机器运行着Linux，我教授一门Java课程，我已在Windows平台上写了大量程序，并且我的书架上至少有10本书充满了带序号的段落来提供解决很多问题的具体建议。
然而，我还是相信在当今这个充满变化的世界，理解说教背后的原理至关重要。
正如你在下面的章节中将会看到的，一旦我们将重点关注于原理，则：　　· 底层技术的选择常常会被虚化。
　　· 我们学到的东西就拥有更广的应用性和更长的时效。
　　· 具体的建议会自行浮现（见每一章末的“那些值得记住的建议”）。
　　最重要的一点在于，是否理解编程讲义背后的原理才是一个受尊重的软件工程师同一个卑微的码农之间的区别。
　　鸣谢　　许多人都慷慨贡献了他们的建议、评注及时间来助我完成本书的写作。
首先，我要感谢本系列的编辑Scott Myers，他担起了类似于读者的角色，巧妙地为本书指引了方向，系统地阐明了本书可在哪些地方变得更加易读和精悍。
Hal Fulton、Hang Lau与Gabor Liptak阅读了本书的样章并提出了许多有用的评注和意见。
Chris Carpenter与Robert L. Glass也审阅了本书的样章及之后的整个草稿，他们的智慧和经验让我受益匪浅。
十分感谢Konstantinos Aboudolas、Damianos Chatziantoniou、Giorgos Gousios、Vassilios Karakoidas、Paul King、Spyros Oikonomopoulos、Colin Percival、Vassilis Prevelakis、Vassilis Vlachos、Giorgos Zervas，特别是Panagiotis Louridas，他们非正式地审阅了本书早期草稿的部分章节，并提供了用以改进的具体评注和建议。
此外，Stephanos Androutsellis-Theotokis、Lefteris Angelis、Davide P. Cervone、Giorgos Giaglis、Stavros Grigorakakis、Fred Grott、Chris F. Kemerer、Spyros Kokolakis、Alexandros Kouloumbis、Isidor Kouvelas、Tim Littlefair、Apostolis Malatras、Nancy Pouloudi、Angeliki Poulymenakou、Yiannis Samoladas、Giorgos Sarkos、Dag-Erling Sm·rgrav、Ioannis Stamelos、Dave Thomas、Yar Tikhiy、Greg Wilson、Takuya Yamashita、Alexios Zavras、Giorgos Zouganelis也提供了有价值的建议，我时常突然跑到他们那里请教一些专业领域中艰涩的问题。
我还想感谢雅典经济与贸易大学管理科学与技术系的同事们对于我工作的支持。
我也感谢下面这三人在本书写作过程中一些必不可少的方面给予我的指导：Mireille Ducassé（技术写作--1990）、John Ioannidis（代码风格--1983）和Jan-Simon Pendry（时间与空间性能--1988）。
　　Addison-Wesley出版社的编辑Peter Gordon熟练地指导了本书的创作，处理了很多棘手的问题。
Kim Boedigheimer以出众的效率应对了日常事务。
7小时的时差常常允许我们在一天内以20个小时一轮换的方式来开展本书的工作。

在本书的创作过程中，Elizabeth Ryan如同一个专业指挥家一般，高效地协调和团结着我们这个全球化多学科的写作团队。

其他作者将本书的文字编辑Evelyn Pyle描述为如同鹰一般目光犀利。

我同意该说法并且还要说一句：她的工作真是十分出色，她找出了很多我永远也想象不出的手稿错误，同时她也比较注意细节和一致性，并据此对相关文字做了修改，仅有少数顶尖程序员才能与之相比。

另外两位具有类似编程技艺的人员也为本书的完成做出了自己的贡献：Clovis L. Tondo依靠对书中工具的深入理解和书中代码的惊人尊重完成了本书的排版工作，Sean Davey则依靠自身卓越的能力确保本书风格的一致。

本书绝大多数所引实例来自于已有的开源项目。

使用源自真实世界的代码允许我能展示人们在实践中可能遇到的代码类型，而非被简化过的“玩具”程序。

所以我想感谢所有对本书中所引开源程序有贡献的开发者，感谢他们向编程社区贡献其工作成果。

若书中代码的作者姓名被列于相应的源代码文件中，则其也会出现于本书的附录中。

<<代码质量>>

内容概要

Jolt大奖素有“软件业之奥斯卡”的美称，本丛书精选自Jolt历届获奖图书，以植根于开发实践中的独到工程思想与杰出方法论为主要甄选方向。
Diomidis
Spinellis首部著作《代码阅读》（Code
Reading）旨在阐明程序员应如何理解与修改代码，与此不同的是，本书重点讨论代码的非功能特性，深入讲述代码如何满足重要的非功能性需求，如可靠性、安全性、可移植性和可维护性，以及时间效率和空间效率。
本书从Apache
Web应用服务器、BSD UNIX操作系统和HSQLDB
Java数据库等开源项目中攫取数百个小例子，并以实例为基准点，辅以理论分析，从实用的角度讲述每个专业软件开发人员能立即运用的概念和技术。

本书荣获2007年Jolt大奖，适用于不同知识层次的软件工作、程序开发和研究人员。

作者简介

自1985年开始，本书作者Diomidis Spinellis在开发大量开创性的，并受到极高评价的商业和开源项目的过程中，一直在钻研、发展本书中所提及的各项技术，期间他编写和维护的代码行数超过25万行。
他在英国伦敦帝国理工学院获得了软件工程方向的硕士学位及计算机科学博士学位。
目前，他是希腊雅典经济与商业大学管理科学与技术系的教授。
他曾撰写过多部畅销世界的计算机技术图书，包括《架构之美》、《代码质量》和《代码阅读》等。

<<代码质量>>

书籍目录

章节摘录

版权页： 插图： 上述与日志记录器进程之间的通信仅仅是高开销的本地IPC操作的一个例子。
其他IPC开销的例子包括管道线中的过滤器进程之间的通信、与本地运行的RDBMS的交互及通过一个本地X Window系统服务进行的I/O操作。
（最后一个例子适用于所有的X客户端GUI程序。
）值得注意的是，在某些情况下，所举的例子中某些数据的复制是可以消除的。
例如，在FreeBSD系统里，当一个发送进程通过一个足够大的缓冲区（PIPE MINDIRECT——8192字节长）把数据写入到管道里时，写入缓冲区会完全映射到内核的内存空间中，而接收进程能够从发送进程内存空间中直接复制这些数据。
在5.4.2节中讨论文件映射操作时将会分析更多的例子。
消除网络栈中不同层之间的数据副本也是网络技术研究人员最喜欢的一种消遣。
最后，看一个远程的进程间通信的例子，例如联系一个远程的DNS服务器来获得一个主机的地址。
每次当工作站的一个用户在不同的主机上访问一个web页面时，类似的交互都会发生。
表4—2中列出的最后一组数字就对应这样的操作。
注意，远程IPC的时间开销要比本地IPC的时间开销（已经很大了）大三个数量级。
通过研究图4—8中描述的相应的交互有助于我们理解这个开销。
这幅图描述的是当使用ping命令向一个远程DNS服务器查询一个主机的地址时发生的系统调用。
系统调用的初始序列——socket、connect及sendto——与我们之前讨论的本地IPC中的初始序列相同。

名人推荐

如果《代码质量》和《代码阅读》得到了应有的关注，我认为，在提高代码专业化水平方面，它们将会给业界带来诸多更加显著的改变，与过往20年以来任何滚滚袭来的浪潮相比，这种改变都会更加迅猛。

——Dr.Dobb's Journal杂志社 Gregory V.Wilson

<<代码质量>>

版权说明