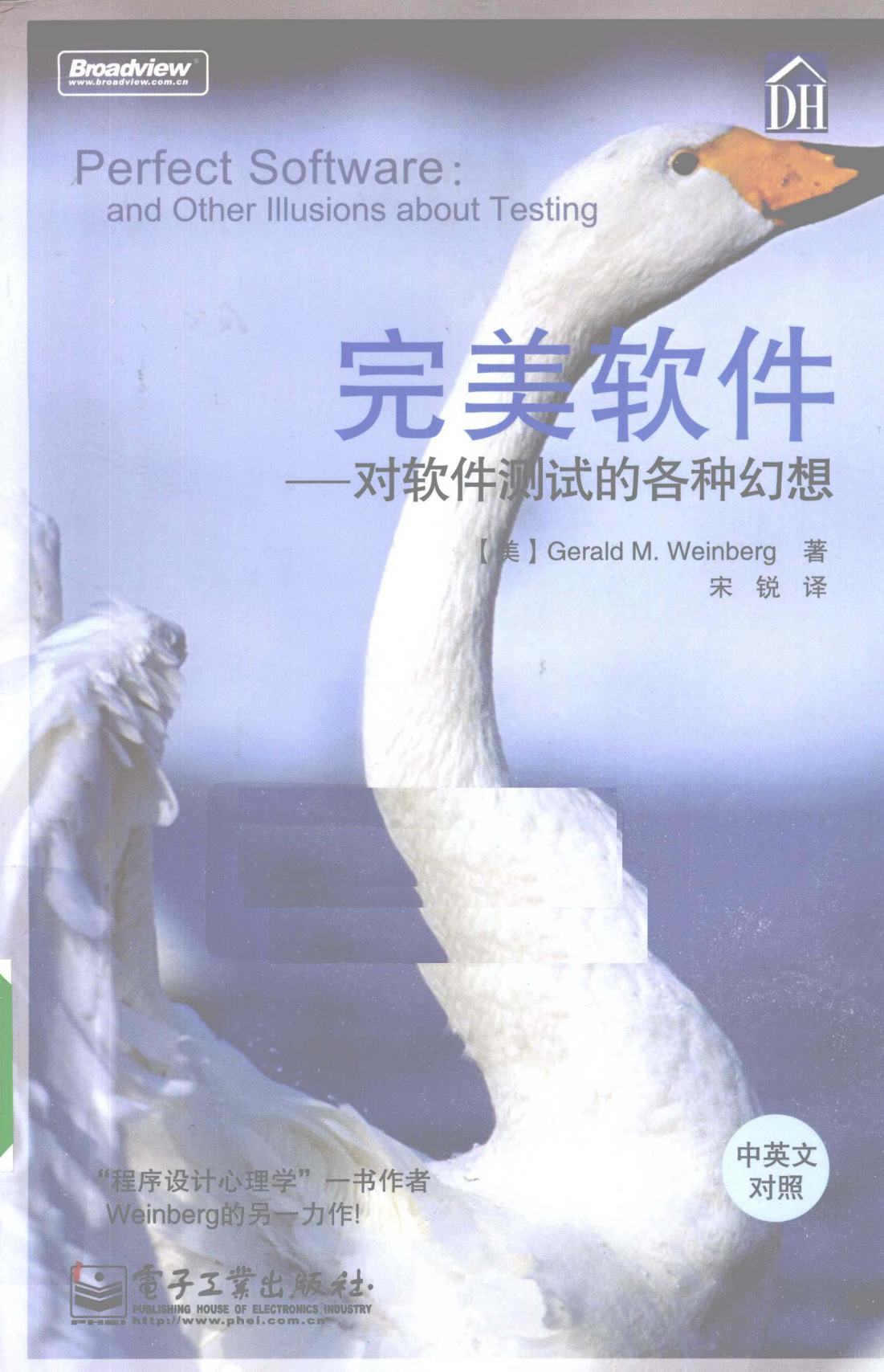
Broadview
www.broadview.com.cn

DH

Perfect Software: and Other Illusions about Testing

完美软件
——对软件测试的各种幻想

【美】Gerald M. Weinberg 著
宋 锐 译



中英文对照

“程序设计心理学”一书作者
Weinberg的另一力作!

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Perfect Software:
and Other Illusions about Testing

完美软件
——对软件测试的各种幻想

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

内 容 简 介

本书是从事软件行业五十余年的 Gernald M. Weinberg 针对软件测试所写的新作。他在软件项目的管理、设计、开发和测试方面都具有极其丰富的经验，对于与软件开发有关人员的心理尤其有深入的研究。在本书中，他重点讨论了与软件测试有关的各种心理问题及其表现与应对方法。作者首先阐述软件测试之所以如此困难的原因——人的思维不是完美的，而软件测试的最终目的就是发现对改善软件产品和软件开发过程有益的信息，故软件测试是一个信息获取的过程。接着，作者利用丰富的经历和大量的实例，展现了在软件测试中可能会出现的各种与人的心理有关的现象、误区、欺诈，以及容易犯下的常见错误等等。本书的重点不是告诉大家要做什么或者说如何做，而更多的是让读者明白在与软件测试相关的活动中会出现某些特定现象的原因。理解这些与人的心理有关的现象有助于与软件开发有关的所有人之间更好地就软件测试的目的和实现过程进行沟通，从而实现具有更高品质的软件。

版权贸易合同登记号　图字：01-2009-0870

图书在版编目（CIP）数据

完美软件：对软件测试的各种幻想 /（美）温伯格（Weinberg,G.M.）著；宋锐译. —北京：电子工业出版社，2009.12
书名原文：Perfect Software—and Other Illusions about Testing
ISBN 978-7-121-09931-1

Ⅰ. 完… Ⅱ. ①温… ②宋… Ⅲ. 软件－测试－研究 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2009）第 213219 号

策划编辑：郭　立
责任编辑：郭　立
特约编辑：顾慧芳
印　　刷：北京机工印刷厂
装　　订：三河市鹏成印业有限公司
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：720×1000　1/16　印张：23.5　字数：288 千字
印　　次：2009 年 12 月第 1 次印刷
定　　价：55.00 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。

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。

服务热线：（010）88258888。

译者序

软件测试的目的是什么？

我想大多数人对于这个问题的回答会是“保证交付高质量的软件”。也许从最终目的上来说是这样的，但是单纯只是软件测试本身是无法为软件的质量提供足够保证的。软件的质量是从需求分析甚至是刚刚产生软件相关概念的时候就开始产生，受到整个开发过程影响的一个性质，而测试只是用来将这一性质数值化、表面化的过程。是不是拥有良好的测试过程就能够保证交付高质量的软件呢？显然是不够的。要保证软件具有较高的品质，需要管理、开发、测试等多个部门的共同努力，尤其是管理部门如何看待测试，如何利用测试获得的信息是至关重要的。

当我们按照一个预定的过程对软件进行测试，却未能获得预期结果的时候，往往会说：“这次测试失败了。”或者是“这个测试没有通过。”而实际上正确的说法应该是：“软件在这个测试中失败了。”或者是“软件没能通过这个测试。”

在许多人看来，这种说法上的差异也许是微不足道的，或者只是文字游戏。但是事实上，它反映了大家对于软件测试的心理误区。人们往往并不能很清楚地区分测试与除错、开发过程甚至是软件本身的关系。而这一误区正是导致对测试产生不切实际的期望，夸大或者忽视测试的作用的根源。

如果你从事测试已经有很长时间了，就很可能听到过这样一些问题：

“你为什么没有发现那个问题？”

“我们为什么要在可以自动化测试的情况下雇佣人来进行测试？”

“一定要让它在下周可用。”

而产生这些问题的原因，往往就在于人们对软件测试的误解。本书

的主要目的，就是要破除这样的一些误解，还软件测试以本来面目。这本书的预期读者非常广泛，涉及与软件开发有关的所有人，包括但不限于软件开发人员、测试人员、客户、项目经理以及高层管理人员，等等。

本书的作者 Gernald M. Weinberg 在软件行业浸淫了五十余年，几乎开发过所有类型的软件，出版了四十余本技术书籍，发表过数百篇技术文章。他对软件项目的管理、设计、开发和测试具有极其丰富的经验，对于与软件开发有关人员的心理尤其有深入的研究。他的《The Psychology of Computer Programming》（中译本名《程序开发心理学》）开创了“以人为本”的研究方法，它以其对程序员们的智力、技巧，团队和问题求解能力等方面独特的视角和敏锐的观察经受住了时间的考验，具有深远影响。

而在《完美软件》这本书中，Weinberg 采用的是纯散文和故事化的风格，而没有使用花哨的指标和图表。他深入浅出地讲述了与软件测试有关的各种误解与错觉，可以让行政人员、经理或者开发人员对测试有足够的了解，以便他们理解测试所要面对的挑战，进而对测试设置恰当的期望并就这些期望进行清晰的交流。

本书回答的问题包括：

- 为什么在看起来测试只会耽搁时间的时候还要进行测试？
- 为什么无法一开始就构建正确的软件，从而不需要测试？
- 需要对所有可能性都进行测试吗？
- 为什么不对所有可能性都进行测试？
- 是什么原因导致测试如此困难？
- 为什么测试需要这么长的时间？
- 是否有可能构建完美的软件？
- 为什么我们不能接受一些缺陷？

本书的入手点更多的是出于心理学而不是计算机技术。通过对心理学的研究，我们会发现人类本身就不是完美的，自然也不应该期望软件是完美的。由于除了某些最简单的情况以外，软件的可能路径实际上都是无限的，因此对软件进行穷举测试通常是不可能的。由于人类容易出错，而且程序中产生缺陷的可能途径也非常多，所以没有任何缺陷的（完美的）程序是一个无法实现的目标。虽然这不是什么好消息，但是事实

就是如此。

虽然软件不可能是完美的，但是受乐观主义和必要性的推动，人们每年仍然会创造数百万行代码，而这些程序中的相当一部分将会被发布到客户群中。因此，负责任的开发人员别无选择，只能通过建立测试过程来发现软件中最重要的和最有可能出现的那些缺陷。

正如 Weinberg 指出的那样，有些人确实选择只进行很少的测试，有时是依赖于诸如“如果我们运气好的话，就没有人会遇到那个缺陷”的想法。其他一些人只是将该缺陷重新定义成一个特性就继续别的工作。这些做法既不合乎道德，也可能导致法律问题。

Weinberg 以非常诙谐而含蓄的风格介绍了他遇到过的众多例子，揭示了与软件测试有关的众多常见误解与谬论，还揭露了在测试中常见的各种欺诈方式。通过阅读本书，应该认识到软件测试是一个与人的心理活动密切相关的过程；测试工作的核心是收集信息，这些信息是关于软件产品、开发过程以及测试过程本身的；要达到“保证软件质量的目的”，不仅仅要进行良好的测试，更重要的是要对测试获得的信息进行合理的利用。

本书是给涉及软件开发的所有人的一本指南，告诉他们：对于软件而言，完美是一个不现实的目标。因此，本书对软件开发团队中的所有人都是必备的。

宋　锐

2009 年 7 月于长沙

Acknowledgments

A book is a product, a software product, in a way. Like any viable software product, a book should be tested prior to release to the public to minimize the risk of its not being useful to its readers. To my team of book testers—James Bach, Michael Bolton, Fiona Charles, Elisabeth Hendrickson, Pradeep Soundararajan, and Dani Weinberg, who together detected dozens of bugs in early drafts of this manuscript—I give wholehearted thanks. I made the bugs originally and it was my job to eliminate or fix them, so if any remain, it's my responsibility, not theirs.

To my clients and students from all over the world, I give thanks for their having contributed the stories and examples that add flesh to the bones on which this book is built. All the examples are authentic and all the stories are true, though most have been disguised in some way to protect confidentiality.

Should you, my readers, find some bug that I missed or some story or example that fits, I hope you will bring it to my attention. I thank you in advance for joining the test team.

致　谢

一本书是一个产品。从某种意义上来说，也是一个软件产品。和所有可行的软件产品一样，一本书在向公众发布之前也应该经过测试，从而让它可能对读者没有帮助的风险最小化。我真心实意地感谢我这本书的测试团队——James Bach、Michael Bolton、Fiona Charles、Elisabeth Hendrickson、Pradeep Soundararajan 和 Dani Weinberg。他们从本书的早期草稿中找出了数十处错误。这些错误原本就是由我制造的，所以消除或修复这些错误也是我的工作。如果书中还有错误的话，那也应该由我来承担责任，而不是由他们承担。

我同样感谢我遍及世界各地的客户和学生们，因为他们提供了许多的故事和例子，为我这本书的骨架增添了血肉。所有这些例子和故事都是真实可靠的，虽然大部分都进行了一定程度的掩饰以保护隐私。

如果读者发现了某些我未发现的错误，或者也有适合本书主题的故事或例子，我希望你能提醒我加以注意。我提前感谢你加入我的测试团队。

前　言

五十多年以前，当我刚开始从事计算机行业的时候，计算机是极其稀少而珍贵的。当我发现自己坐在一间巨大的房间前的控制台面前时，还只是一个十几岁的少年。那个房间里塞满了硬件，大约拥有当时世界上整个计算能力的百分之十。我只要伸出手指按一下，就可以关掉这样的计算能力——有时候这确实就是我的工作。

只要愿意排队，你就可以用大约两倍于我月薪的钱来租用这样的计算能力一个小时。今天，计算机已经非常便宜，实际上我已经在捐赠一些计算速度比那些早期型号快五万倍、内存也要大五万倍的计算机。捐赠的这些机器非常紧凑而轻便，我可以很轻松地把它放进公文包中。而它的计算能力却连当今世界计算能力的十亿分之一都不到。

显然，我目睹了计算机技术的长足发展，不过这一发展对我的影响是潜移默化的。我实际上并未注意到这些变化，直到几年前听到一则有关卡纳维拉尔角取消一次火箭发射的新闻报道。在播音员结束新闻报道时，其中一人说："太空总署（NASA）说这是一个计算机软件错误。"

另一名播音员回应道："像计算机这么简单而常见的东西也会出这样的错，难道不奇怪吗？"

第一个人回答道："是很奇怪，难道你认为他们不会对软件进行测试？"

* * *

Yes, I think they would test their software. In fact, I'm sure they did. Evidently, though, the announcer thought testing must inevitably lead to a perfect product.

I couldn't stop thinking about this dialogue and the broadcaster's unrealistic expectations. At first, I simply shrugged and told myself, "Such is the ignorance of the general public about software testing," but it was too late to shrug it off. My consciousness was now raised. I began to see how even the managers at my software-building clients display the same ignorance. *Especially* the managers. Software testing was driving them crazy.

I'm an empathic person, and when people around me suffer, I suffer with them. I could see that the software managers were suffering, along with their employees and customers. I could also now see that almost all of them were suffering not so much because testing was complicated or time-consuming or laborious; they were suffering because of their unreasonable expectations and fallacious models about software testing.

Finally, I decided to do what I always seem to do when I find ignorance causing pain: I wrote about the problem and probable solutions. What I wrote eventually became this book.

* * *

When I wrote *The Psychology of Computer Programming* many years ago, I intended it to help people who want to understand programming. [1] Quite a few people have told me it helped, so perhaps, in part, the success of that book inspired me to write this book: to help people who want to understand testing.

My intended audience is broad: I envision this book in the hands of professional testers, developers, consumers, analysts, designers, programmers, all of their managers, and all of their coworkers.

Most *professional testers* will know most of what's in this book, but I hope that by reading on, they will see new ways to communicate what they know—to *their* managers, developers, coworkers, and customers.

I'd like to help both *developers* and *testers* understand what their *managers* face when they confront software testing issues.

不错，我认为他们会对软件进行测试。事实上，我确信他们进行了测试。不过，播音员显然认为测试必然应该产生完美的、没有任何缺陷的产品。

我无法停止思考这段对话以及播音员不切实际的期望。一开始，我只是耸耸肩膀，对自己说："这就是普通大众对软件测试的无知。"不过现在要忽视这一问题已经太晚了。我的警觉已经提高了。我开始发现甚至是我那些生产软件的客户公司的经理们也表现出了同样的无知。*尤其*是那些经理们，软件测试简直要让他们发狂。

我是一个容易认同别人感情的人。当我周围的人痛苦时，我同样会感到痛苦。我理解软件经理们正在感受痛苦，他们的员工和客户同样如此。我还知道他们感到痛苦的原因，几乎所有人都并不完全是因为软件测试非常复杂或者耗时耗力，而更多的是由于他们对软件测试的不合理期望和那些靠不住的测试模型。

"最后，我决定还是采取我解决无知导致的问题时最喜欢采用的方法：写下这个问题以及可能的解决方案。我写下的内容最终形成了本书。"

在我多年以前写作《*The Psychology of Computer Programming*》一书时，我希望它能够帮助那些想了解编程的人[1]。有不少人告诉我这书确实为他们提供了帮助，所以，在一定程度上也许是那本书的成功促使我着手写作这本书：为了帮助那些想了解测试的人。

这本书的预期读者非常广泛：我预想这本书会拿在专业测试人员、开发人员、客户、分析师、设计师、程序员，以及他们的所有经理和同事手中。

大部分专业测试人员都知道这本书中的大部分内容，但我还是希望他们可以通过阅读本书发现一些新的途径，来和他们的经理、开发人员、同事和客户交流他们知道的那些信息。

我希望能够帮助开发人员和测试人员了解他们的经理在遇到软件测试事宜时会面对哪些情况。

I'd like to help *customers*—the people who buy the software—to be better-informed consumers.

And, since everyone is affected by software today—and hurt by poorly tested software—I'd like to help *everyone* to care more about testing.

Because I hope to reach a broad audience, I've chosen to use plain language wherever possible, avoiding highly technical terms and overly detailed analyses. (For those who are interested in learning more technical detail about software testing, I point to some particularly solid books in the Additional Reading section at the end of this book.) To keep my focus on helping people who want to understand testing, I've organized the book around the questions that puzzle the most people:

Why do we have to bother testing when it just seems to slow us down?

Why can't people just build software right in the first place, so it doesn't need testing?

Do we have to test everything?

Why not just test everything?

What is it that makes testing so hard?

Why does testing take so long?

Is perfect software even possible?

Why can't we just accept a few bugs?

April 2008 G.M.W.
Albuquerque, New Mexico

我希望能够帮助客户，也就是那些买软件的人，了解更多的信息。

最后，既然现在每个人都会受到软件的影响，并且受到未经良好测试的软件的伤害，我希望能够帮助所有人更加注意测试。

由于希望本书能够获得广泛的受众，所以我选择尽可能使用平直的文字，避免那些高度技术化的术语和过于详细的分析。（对于那些有兴趣学习有关软件测试的更多技术细节的人，我在本书结尾的其他阅读材料部分指出了一些非常有用的参考书籍。）为了集中精力帮助那些希望了解软件测试的人，我根据下面这些会让大多数人感到困惑的问题来组织本书：

为什么在看起来测试只会耽搁时间的时候我们还要进行测试？

为什么大家不能一开始就构建正确的软件，从而不需要测试？

我们需要对所有的可能都进行测试吗？

为什么不对所有的可能都进行测试？

是什么原因导致测试如此困难？

为什么测试需要这么长的时间？

是否有可能构建完美的软件？

为什么我们就是不能接受一些缺陷？

G. M. W.

2008 年 4 月于新墨西哥州阿尔伯克基

1

Why Do We Bother Testing?

> *". . . the realization came over me with full force that a good part of the remainder of my life was going to be spent in finding errors in my own programs."*
>
> —William Aspray and Martin Campbell-Kelly, *Computer: A History of the Information Machine* (New York: Westview Press, 2004), p. 167.

Suppose you have a great idea for a software program that will compute the ideal investment strategy. You plan to use the software to turn your meager life savings into millions. You write the software, and you're sure you've done it perfectly. Now what would you do?

> *Would you test the program before investing every penny you've saved based on its investment-strategy advice?*
>
> *Would you test it if someone else had written the program?*

If you answered the first question, "No, I wouldn't bother to test it," don't read any further. This book is not for people who think that they can write a perfect program and be perfectly sure that it is perfect. If you also answered no to the second question, you must believe that human thinking can be perfect. Answering no to either or both of these questions means, to me, that you are in a category of people who will not enjoy this book because you won't understand why it's necessary. Put it down right now. Spare yourself further pain.

目　录