

P e r f e c t S o f t w a r e

温伯格技术思想 三部曲

颠覆完美软件

软件测试必须知道的几件事

[美] Gerald M. Weinberg 著
宋锐 译

a n d O t h e r I l l u s i o n s a b o u t T e s t i n g



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

P e r f e c t S o f t w a r e :

温伯格技术思想  三部曲

颠覆完美软件

软件测试必须知道的几件事

[美] Gerald M. Weinberg 著

宋锐 译

a n d O t h e r I I

T e s t i n g

软件从业者思想启蒙的巨著

技术人生的必读经典

电子工业出版社

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 Gerald 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，尤其对软件开发人员的心理有深入的研究。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。本书的重点不是告诉大家要做什么或者如何做，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。

Original edition copyright ©2008 by Gerald M. Weinberg. All rights reserved. Translation published by arrangement with Dorset House Publishing Co.,Inc.(www.dorsethouse.com) through the Chinese Connection Agency, a division of The Yao Enterprises, LLC.

本书中文简体版专有出版权由 The Yao Enterprises, LLC.代理 Dorset House Publishing Co.,Inc.授予电子工业出版社，未经许可，不得以任何方式复制或抄袭本书的任何部分。

版权贸易合同登记号：图字·01-2013-4962

图书在版编目 (CIP) 数据

颠覆完美软件：软件测试必须知道的几件事 / (美) 温伯格 (Weinberg, G.M.) 著；宋锐译。—北京：电子工业出版社，2015.7

(温伯格技术思想三部曲)

书名原文：Perfect software and other illusions about testing

ISBN 978-7-121-25861-9

I. ①颠… II. ①温… ②宋… III. ①软件—测试IV. ①TP311.5

中国版本图书馆 CIP 数据核字(2015)第 074361 号

策划编辑：刘 皎

责任编辑：徐津平

印 刷：三河市双峰印刷装订有限公司

装 订：三河市双峰印刷装订有限公司

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：11 字数：195 千字

版 次：2015 年 7 月第 1 版

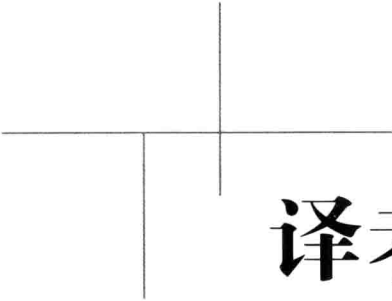
印 次：2015 年 7 月第 1 次印刷

定 价：49.00 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。事实上，它反映了大家对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。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、项目经理及高层管理人员，等等。

本书的作者 Gerald 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《颠覆完美软件：软件测试必须知道的几件事》这本书中，Weinberg 采用的是纯散文和故事化的风格，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临的挑战，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。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。

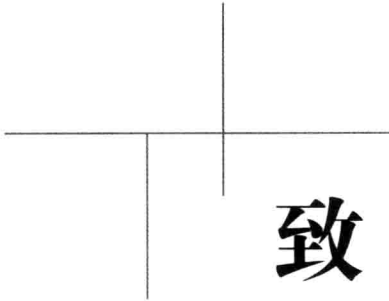
虽然软件不可能是完美的，但是受乐观主义和必要性的推动，人们每年仍然会创造数百万行代码，而这些程序中的相当一部分将会被发布到客户群中。因此，负责的开发人员别无选择，只能通过建立测试过程来发现软件中最重要的和最有可能出现的那些缺陷。

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，我们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

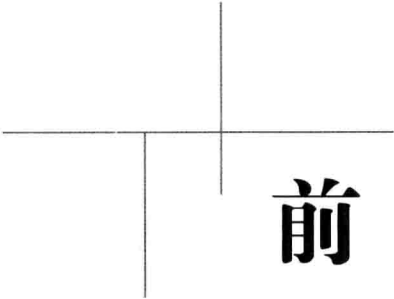


致 谢

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，从而让它对读者没有帮助的风险最小化。我真心实意地感谢本书的测试团队——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，因为他们提供了许多故事和例子，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。

那时，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，其中一个人说：“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？”

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了。

我是一个容易情绪化的人。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：写下这个问题及可能的解决方案。我写下的内容最终形成了本书。”

多年前，在我写作 *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。

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，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，而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，所以我尽可能使用通俗的文字，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时我们还要进行测试？

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？

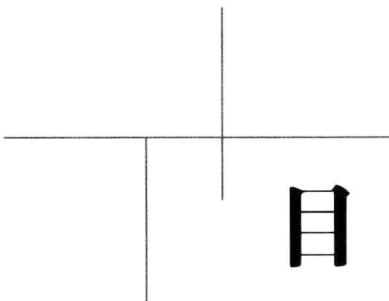
为什么测试需要这么长的时间？

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



目 录

1	进行测试的原因	1
1.1	人类不是完美的思考者	2
1.2	我们要做出有关软件的决定	2
1.2.1	日记条目 1	2
1.2.2	日记条目 2	3
1.2.3	日记条目 3	3
1.2.4	日记条目 4	3
1.2.5	日记条目 5	4
1.2.6	日记条目 6	4
1.3	决定可能是有风险的	5
1.4	测试可以提供降低风险的信息	6
1.5	小结	8
1.6	常见错误	8
2	测试无法做的事	10
2.1	信息未必有助于降低风险	11
2.2	也许我们不会使用那些花钱得到的信息	12

2.3	决定是感性的而不是理性的	13
2.4	不良的测试也许比不测试更糟	14
2.5	产品可能尚未准备好接受测试	14
2.6	小结	15
2.7	常见错误	15

3

不对所有可能性进行测试的原因

18

3.1	可能进行测试的数目是无限的	19
3.2	测试最多只是采样	20
3.3	信息的成本可能超过无知的成本	21
3.4	也许我们可以用较少的测试获取更多的信息	22
3.5	测试自助餐	22
3.6	小结	23
3.7	常见错误	23

4

测试和除错的区别

25

4.1	测试会随着机构的成长发生变化	28
4.2	以时间限制试探法作为管理法则，但根据需要进行调整	30
4.3	小结	31
4.4	常见缺陷	31

5

元测试

33

5.1	我们有说明书，但是找不到了	34
5.2	我们的错误太多了，导致缺陷数据库无法高效运转	35
5.3	我们没找到任何缺陷，实际上我们并没有真正地找	35
5.4	我们修改记录让缺陷看起来没那么严重	36
5.5	这不是我的组件中的问题，所以我不记录	36

5.6	我不知道在测试错误的应用程序	37
5.7	我们不测试最差的组件，因为花的时间太长	37
5.8	我们已经发现了这么多缺陷，不会再多了	38
5.9	我们的测试证明程序是正确的	38
5.10	我们运行了很多测试用例，根本就看不过来	38
5.11	如果我们的软件在有三名用户时工作良好，显然它在有一百 名用户时也不会有问题	39
5.12	我们不希望测试人员知道我们将忽略他们提供的信息	39
5.13	我没有报告缺陷，所以开发人员不会对我发脾气	40
5.14	我们不需要测试它，因为开发人员非常有水平	40
5.15	接着说元信息	41
5.16	小结	41
5.17	常见错误	41

6 信息免疫 44

6.1	我们在生存规则受到威胁时会感到害怕	45
6.2	我们压抑无法接受的事物	46
6.3	我们让不可接受的事物合理化	47
6.4	我们将自己的负面品质投射给其他人	48
6.5	我们转移指责从而免除自己的责任	49
6.6	我们对自己的不足进行过度补偿	51
6.7	我们在觉得失去控制时开始强迫自己	51
6.8	小结	52
6.9	常见错误	52

7 如何应对防卫反应 54

7.1	确定恐惧	55
-----	------------	----

7.2	使用危机思维	55
7.3	实践，实践，再实践	56
7.4	对自己进行测试	57
7.5	小结	58
7.6	常见错误	58

8

良好测试的要素

59

8.1	永远无法确切地知道	59
8.2	只能根据事实来评估良好性	61
8.3	你可能希望故意插入一些缺陷	62
8.4	对良好性的估算总是统计性的	62
8.5	可以对非差性进行估算	63
8.6	小结	64
8.7	常见错误	64

9

有关测试的主要误区

66

9.1	指责误区	66
9.2	穷举测试误区	67
9.3	“测试产生质量”误区	68
9.4	分解误区	69
9.5	合成误区	70
9.6	“所有测试都相同”误区	71
9.7	“随便哪个笨蛋都可以测试”误区	72
9.8	小结	73
9.9	常见错误	73

10 测试不仅仅是敲键盘 75

10.1	毫无目的地敲击键盘是不是测试	76
10.2	白手套测试	77
10.3	狗食测试	78
10.4	对测试人员也要进行测试	80
10.5	可能在没有意识到的情况下进行测试	80
10.6	演示不是测试	81
10.7	小结	82
10.8	常见错误	82

11 信息摄取 84

11.1	使用萨提亚交互模型来解析沟通	84
11.1.1	摄取 (Intake)	85
11.1.2	确定含义 (Meaning)	85
11.1.3	确定重要性 (Significance)	86
11.1.4	做出反应 (Response)	86
11.2	人们听取信息时是有选择性的	87
11.3	数据来源会影响到摄取	87
11.4	时机也会导致差异	88
11.5	人们会出现信息过载	88
11.6	减少测试的数量也许可以传递更多的信息	89
11.7	寻找测试之外的信息摄取	90
11.8	不要混淆理解和摄取	90
11.9	使用数据质疑来过滤理解	91
11.10	小结	91
11.11	常见错误	91

12

确定含义

93

12.1	案例 1：四个缺陷，五种含义	94
12.2	案例 2：四个缺陷，七种含义	95
12.3	案例 3：四个缺陷，自行确定含义	96
12.4	进行解释之前先弄清期望的是什么	96
12.5	不知道期望时的做法	98
12.6	使用已经获得的信息	98
12.7	使用间接信息	99
12.8	使用未获得的信息	99
12.9	同样的话可能具有不同的含义	100
12.10	“相同”可能并不一样	101
12.11	某些时候不精确会更好	101
12.12	小结	102
12.13	常见错误	102

13

确定重要性

104

13.1	不同人会给同样的信息赋予不同的重要性	105
13.2	公共的重要性也许和个人的不一样	106
13.3	重要性依赖于上下文环境	107
13.4	不能总是根据金钱来确定重要性	108
13.5	不要采用过细的尺度	110
13.6	首先解决重要问题	110
13.7	听从自己的情绪反应	111
13.8	小结	112
13.9	常见错误	113

14 做出反应 115

- 14.1 是运气不好还是管理不善 115
- 14.2 项目最后会赶进度的原因 116
- 14.3 接近项目结束时应如何反应 117
- 14.4 对测试所需时间的估算与现实差距很大的原因 118
 - 14.4.1 好天气估算 119
 - 14.4.2 不切实际的过程模型 119
 - 14.4.3 低质的过程数据 119
 - 14.4.4 没有过程数据 121
- 14.5 确定是否已经错过了可以有所改变的时刻 122
- 14.6 小结 122
- 14.7 常见错误 122

15 避免软件测试变得更困难 124

- 15.1 情况变得更糟的原因 124
- 15.2 让系统尽可能小 126
- 15.3 让“系统”模型是可扩展的 126
- 15.4 增量构建有清晰接口的分立组件 127
- 15.5 减少进入产品的缺陷数目 127
- 15.6 小结 128
- 15.7 常见错误 128

16 不使用机器进行测试 130

- 16.1 用机器进行测试总是不够的 130
- 16.2 首先对最差的部分进行评审可以让人了解缺陷的严重性 135
- 16.3 事实并不总是能令人信服的 136