《探索式软件测试》

图书基本信息

书名：《探索式软件测试》

13位ISBN编号：9787302223849

10位ISBN编号：730222384X

出版时间：2010 年4月

出版社：清华大学出版社

作者：James A. Whittaker

页数：230

译者：方敏;张胜,钟颂东;郭艳春

《探索式软件测试》

前言

“ 用户购买功能的同时也在忍受缺陷。 ” ——史考特 · 沃兹沃思(Scott Wadsworth) 任何一个使用过电脑的人都知道软件故障。从最初的第一个程序到最新的现代应用程序，软件从来没有完美过。 将来这一切也多半不会改变。不只是软件开发错综复杂和开发人员容易犯错的天性，还有硬件、操作系统、运行时环境、驱动程序、平台、数据库等不断变化，这些加在一起使软件开发任务成为全人类最让人称奇的专业技能之一。 不过，仅仅令人称奇是不够的，正如本书第1章指出的那样，这个世界还要求软件拥有高质量。 显然，关心质量不只是软件测试人员的事。我们应该用正确的方式构建软件，将可靠性、安全性和高性能等作为系统设计的一部分来考虑，而不是到开发末期才想起它们。然而，说到理解软件缺陷的本质，测试人员总是站在前沿。如果没有测试人员在测试的最前沿发挥他们的洞察力、技术以及应变措施，使这样的可能性变为现实，软件质量的全面解决方案差不多算是“ 镜中花，水中月 ”。 谈论软件质量的方法有很多，感兴趣的听众也有很多。本书是为软件测试人员而写的，写的是一种我认为比其他任何缺陷都重要的特殊缺陷：即逃过所有各种检测手段而最终存在于发布产品中的缺陷。 任何一个软件公司发布的产品都有缺陷。缺陷是怎么引入的？为什么没有在代码审核、单元测试、静态分析或其他面向开发人员的活动中把它们找出来？为什么自动化测试没有找出它们？那些缺陷有些什么特质使其能逃过手工测试？ 什么是找出产品缺陷的最好方法？ 本书针对的正是最后一个问题。在第2章“ 手工测试 ”中，我提出了一个观点：因为用户是在使用软件过程中找到这些缺陷的，所以我们的测试人员也应该通过使用软件来找到它们。无论使用自动化测试和单元测试，还是其他一些手段，都难以接触到这些缺陷。无论测试人员怎么实现自动化测试，即使全部都自动化，这些缺陷还是会处处作怪，并在产品中屡屡重现从而伤害最终用户。 问题在于很多现代化手工测试实践都缺乏目的性，随机性强且重复性强。有些人可能还会加上一条：手工测试无聊透顶。本书试图为手工测试流程提供一些指导、技术和规划。 在第3章“ 局部探索式测试法 ”中，针对测试人员在运行任何一个测试用例时都需要做出很多细微的战术层面决定，我给出了详尽的指导建议。测试人员必须决定对于某个特定的输入字段应该使用什么输入值，或者给应用程序使用的文件提供什么数据。在测试过程中，必须做出许多这样的小决定。在缺乏指导的情况下，这些决定常常是未经分析且不是最优化的。在向一个文本框内输入一个数时，选择整数4难道就胜过整数400么？应该用长度为32字节的字符串还是长度为256字节的字符串？选择一个而不选另一个是有一定道理的，这一切都取决于处理该输入的软件的具体情况。鉴于测试人员每天都要做出数百次这样的小决定，在这里提供有效的指导建议显得至关重要。 在第4章“ 全局探索式测试法 ”中，针对测试人员在编制测试计划和测试用例设计时需要考虑哪些广泛的战略性问题，我也给出了一些指导建议。这些技术都基于“ 漫游测试 ”(tour)概念，如同一个导游带领旅游团队参观大都市中一系列著名景点一样，这种漫游测试法指出的路线可以指导测试人员如何探索软件的方方面面。这里的探索并不一定是随机的或者漫无目的的。本书所记录的方法已经成为微软和谷歌的许多测试人员日常工作的一部分。诸如“ 地标测试法 ”(landmark tour)和“ 极限测试法 ”(intellectual ’ s tour)等词汇已经列入了手工测试人员的标准词汇表中。测试技术以前确实被称作“ 漫游 ”，但是用整个旅游业来隐喻软件测试，并在测试实际发布的应用程序时，大规模使用这些隐喻的名称，还属于本书的一个创举。 全局探索式测试法对于制定完整的测试策略给出了指导建议。例如，如何创建一组特性覆盖率(feature coverage)较高的测试用例？如何确定是否要在一个单独的测试用例中使用多个特性？如何创建一个完整的测试用例套件(test case suite)，从而使软件尽可能地满负荷工作以便能找到更多重要的缺陷？这些都是设计测试用例和保证测试套件质量时必须解决的重大问题。 在第5章“ 混合探索测试技术 ”中，通过把探索式测试和传统的脚本或基于场景的测试技术相结合，进一步扩展了漫游的概念。我们将讨论如何修改各种端到端场景(end-to-end scenario)、测试脚本(test script)或用户故事(user story)，来创造更多的变化情况，以激发传统静态测试技术查找缺陷的潜力。 在第6章“ 探索式测试的实际应用 ”中，来自微软不同产品组的五位客串作者提供了他们使用漫游技术后得到的经验报告。这些作者和他们的团队在真实的开发环境中，把漫游方法应用在真实的软件上。他们记录了各自是如何使用漫游、修改漫游甚至创建自己的漫游的。这些内容来自于使用漫游法测试重要的关键软件产品的测试人员，属于真正的第一手资料。 最后，我用两章内容总结前面各章所讨论的内容。在第7章“ 漫游测试的棘手问题 ”中，描述了我认为的测试中最困难的几个问题，以及如何将那些具有高度针对性的探索式测试方法融入一个更广泛的解决方案中。在第8章“ 软件测试的未来 ”中，我更进一步讨论在未来几年中，诸如虚拟化、可视化甚至电视游戏之类的技术将如何改变测试

的面貌。附录包括我对测试职业生涯的看法，收集了我以前一些深受读者喜爱的文章(加入了一些新的注解)，其中一些文章已经无法在其他地方看到了。 写这本书对我来说是一种享受，我希望你阅读本书也是一种享受。

《探索式软件测试》

内容概要

《探索式软件测试》任命软件测试人员，OA专家、开发人员、程序经理和架构师阅读，对他们的工作具有重要的启发作用。探索式软件测试作为一种富有创新精神和现实意义的测试方法，引起越来越多软件测试人员、质量保证人员和项目经理的高度重视。《探索式软件测试》作者结合自己二十年的经验，从多个角度结合富的实例阐述了探索式软件测试的使用技巧、提示和相关技术。全书共8章，3个附录，为手工测试流程提供了重要的指导，技术和规划。

《探索式软件测试》

作者简介

詹姆斯·惠特克(James Whittaker)的全部职业生涯都致力于软件测试，在该学科的许多方面都留下了他的印记。他是基于模型的测试领域的先驱，他在田纳西大学的博士学位论文是该主题的标准参考资料。他在错误注入(error injection)方面的工作，创造了备受欢迎的运行时错误注入工具Holodeck。他是软件安全和渗透测试(penetration testing)的早期创导者。作为教师和演讲者，他也为人们称道，他曾在国际会议上赢得过多个最佳论文和最佳演讲奖。

《探索式软件测试》

书籍目录

《探索式软件测试》

章节摘录

插图：已经有很多文章阐述了漫无目的的生活会带来哪些坏处，但盲目的测试和现代测试实践中的无目标性，导致了这样一个问题。软件测试不是简单地拿起来就做的事情。它要求有计划、有准备、有策略和有多变的战术，这是成功进行软件测试的前提。但是，太多的软件企业因偏爱“想干就干”而忽略适当的准备。测试工作非常重要，所以决不能如此随便地对待它。我在佛罗里达理工学院担任教授的时候，我教过软件测试课程。有一个学期，注册这门课的学生多得超乎我的想象。我决定做一个试验，想把少数随意选择这门功课的学生吓走。在开学的第一天，我在计算机房上课，我指导学生们确定要测试的应用程序，两人为一个测试小组。我没有给学生们完成这个测试任务提供进一步的指导。作为鼓励，我答应他们，如果我对他们采用的技术留下深刻的印象，他们就会被留在我的班上。如果他们不能满足要求，我就认为他们自动放弃（我不是有意要赶他们走，但是这种要求达到了我想要的效果）。我在实验室里走来走去，无形中对学生们产生了压力，有时我会与一组学生攀谈，要求他们解释如何找到软件错误。每当我提出这样的问题，就会得到类似的答案，如“不太清楚，我们只希望它会出错。”最后，一些聪明的学生意识到这种回答并不奏效，他们在随后的回答中渐渐地开始包含一些策略。事实上，我清楚地记得有两个学生说“我们将要对所有的文本输入框输入较长的字符串，希望能够找到一些不检查字符串长度的地方。”①我当即接受他们成为我的第一组学生，并鼓励他们继续做下去。太好了！虽然这不是最好的或者最重要的策略，但它至少是一种策略，有助于减少测试的无目标性。软件测试人员运行测试程序时，经常缺乏策略或没有指定明确的目标。在他们进行手工测试时，他们漫无目的地测试应用程序。在他们实现测试自动化时，仅仅由于他们熟悉怎样编写测试自动化，就去这么做了，而不考虑这样的自动化是否能发现有价值的缺陷，是否经得起时间的考验，是否值得付出维护费用。

后记

人非圣贤，孰能无过。一语道破人类容易犯错的天性，也为人们提供了堂而皇之逃避责任的借口。但随着科学技术的发展，软件在金融、医学、航天、汽车等行业全方位的渗透，导致我们越来越难借助于这样的托词，原谅自己的失误，因为一个微不足道的疏忽或者失误，放过软件中存在的漏洞、缺陷和隐错，就会导致无可挽回的损失甚至灾难性的后果。上世纪末在美国，因为两套软件采用的度量单位不一致（一个公制单位N／s，一个英制单位b／s，后者是前者的4倍），导致美国国家航空航天局（NASA）几亿美元的损失，使火星气候轨道飞行器在最后的关键时刻坠入火星，而不是顺利进入轨道按照既定设计绕火星运行。此类事例在计算机一统天下的今天并不罕见，同时也为我们敲响了警钟，软件测试到了不得不高度重视的程度。反观软件测试行业，可用的测试方法与10年前却没有多大差别。要想提高软件质量，测试先行，测试创新，已经到了刻不容缓的地步。探索式测试（Explory tory Testing，也称探索性测试）是一种软件测试方法，最先是CemKaner在1983年提出的。这是一种强调个人自由与责任的测试方法，让独立测试人员可以借由不断的学习来改善测试的规划与测试的执行，而在测试的过程中也会同时改善测试案例达到相辅相成的效果。在Nortel和微软的很多项目中，都采用了这一新颖、有趣和富有创意的测试方法。

媒体关注与评论

“好书！Whittaker讲述的概念有创意、巧妙、令人印象深刻。他真正懂得如何鼓励工程师们以不同的角度考虑软件测试。” ——谷歌公司测试工程总监Patrick Copeland“James把美妙的手工测试方法学提升到了极致。漫游不仅概念正确，而且实际应用得很好，我们已经在所有测试人员的内部培训课程中分享了漫游的概念。如果你想把手工测试过程带到21世纪，请阅读这本书吧。” ——微软公司卓越测试总监Alan Page“1990年，我开始与James在IBM一起共事。他早在当时就鼓励测试人员和开发人员大胆创新。通过这本书，他把对软件质量的热情提升到了全新的高度。请阅读这本书吧，见证你自己成为一个更好的测试工程师。James是这方面的权威，这个星球上的软件测试工程师和开发工程师，无论他们是真正关心软件质量，或者只是想在自己的日常工作中增加更多的乐趣，都应该阅读这本书。” ——Cisco Systems公司高级工程主管Kaushal K. Agrawal“James Whittaker是测试领域中一位真正有远见的人士。Utest和我们的QA专业全球社区经常向James咨询以获得他的灵感、他对测试未来趋势的解读和他的各种测试理念。现在，他终于为大家写出了这本书，我们这个行业会由于这本书而变得更有智慧。” ——uTest公司CEO和合伙人Doron Reuveni“只有像James Whittaker这样的人才会把旅游和软件测试以小说的形式结合起来，也只有James才能把它做到这么天衣无缝。漫游方法不仅提供了令人难忘的有效思考模式，还把适当的结构和组织与广泛的探索和创造结合在一起。bug们，小心啦！” ——谷歌公司Alberto Savoia“James是软件测试主题最出色的演讲者之一，读他的书就像聆听他的演讲。如果你希望增加测试知识并成为一个更优秀的测试人员，这本书就是为你而写的。” ——TCL集团公司主席和合伙人Stewart Noakes“我从事探索性测试已经有段时间了，James的漫游测试法给我所使用的各种方法起了名字，定义了测试重点，更重要的是给了我实际的指导。这本书将会使探索式测试的教学和应用变得更方便。” ——iMeta Technologies公司高级测试顾问Rob Lambert“我为这项工作感到非常激动，它概念新潮却又合情合理。连我这样的普通读者都能轻松地理解和使用，而不用首先去学习一些华而不实的过时理论。在阅读本书时，我也从不需要借助字典。测试领域长久以来就一直期盼着这样的创新之作，我由衷地认为本书在这方面走在行业最前沿。” ——Netjets公司QA部门经理Linda Wilkinson

《探索式软件测试》

编辑推荐

如何发现和修复被常规软件测试忽略的关键软件缺陷？在《探索式软件测试》中，享誉业界的软件测试专家Ja rrlesWhittaker揭示了当下最严重、隐藏最深的软件错误的真正诱因，并介绍了如何利用功能强大的探索式测试技术来找到并纠正这些错误。先后就职于谷歌、微软和其他顶尖软件公司的james Whittaker，在软件测试的前沿阵地拥有近20年的丰富经验，他为传统的手工测试引入了可重复、规范、可传授和特别高效的新过程。Whittaker定义了针对单个测试人员的简单技术和针对大规模测试团队的复杂技术。他还引入了一个混合策略，将探索式概念引入传统脚本测试。在《探索式软件测试》中，可以体会到如何在恰当的时机使用这些方法，如何成功地充分应用这些方法。简洁、诙谐和可行，《探索式软件测试》引入的这些技术已经经过上市软件的测试人员广泛应用，人们在实际测试过程中深受这些方法的启发，成功实现了预期目标。《探索式软件测试》是为测试人员、QA专家、开发人员、程序经理和架构师所写的。《探索式软件测试》涉及以下重要问题：为什么自动化测试无法消除所有缺陷，如何才能让这些缺陷无处遁形？哪些技术可帮助我不断发现和消除致命错误？如何更高效地进行手工测试，增加些许轻松和愉悦的感觉？对于每个项目，如何确定最高效的高级测试策略？在我无法进行全部测试时，哪些输入是必须测试的？哪些测试用例能提供最理想的特性覆盖率？在结合使用探索测试和传统脚本或场景测试时，如何才能获得理想效果？如何体现来自开发过程的反馈意见，代码更改吗？

《探索式软件测试》

精彩短评

1、看了这本书，我才把我用了很久的测试方法上升成了理论。唯一感觉别扭的就是里面的话有点繁琐。
2、力荐。相当好的一本软件测试书。入门必读。读过直接就可以上手工作了。
3、这本书是写得很好，但是书的印刷质量太差了，竟然有十几页是空白的。我一直对亚马逊的质量都很满意的，第一次给亚马逊差评的，这次却大的失望了～
4、写得挺有意思的一本书，包含对软件测试未来的一些无边的设想。
5、正文不是很长，让我对快速过完一遍很有信心。
正在读，挺受启发的。
6、我的第一本具有重大指导意义和启发的软件测试书籍，再也不是看一堆屁用的理论。作者教我们以一种找茬的身份来进行测试，并且给出各种实际例子。二刷是必须的。
7、很不错的书，对于测试的思想很有帮助
8、工作中碰到瓶颈了，开始学习。
9、文章写的通俗易懂~
10、挺好的 没什么缺点
11、探索式测试
12、书本质量很高，内容也很通俗，主要交了探索式测试法，不同于常规的测试方法
13、一直不知道该拿起什么样的书来对自己的专业技能进行强化一下，直到看到这本书。从不知道原来自己做的工作一直是属于这样的测试方法，系统的学习来使自己的工作更有效~
14、只是粗略地介绍了探索性测试技术，至于如何应用在实际项目中，书中的干货显然不多，毕竟这个并不是可以三言两语可以描述出来的东西。很多年过去了，在业界也没有一个很成功探索性测试例子，充其量也是一个包装得更科学的随机测试，而作者自己也承认，这仅仅是探索，而不是测试，不要抱以太多期望，也是醉了。总的来说还是蛮有用的，附录感觉更难体现测试技术的价值，读完还算蛮有收获。
15、想具体学习一下探索式测试的理论基础，看看还是不错的，第四章是精华。
16、作者经验丰富，探索式测试对于提高测试效率和质量，充分发挥测试者的智慧是很好的一种思想
17、看了3次算是对里面的内容有了一个比较仔细的理解、同时也写了几篇关于探索式软件测试的文章
http://www.hiadmin.org/testing/exploratory-type/探索式软件测试的四个类型
http://www.hiadmin.org/testing/exploratory-testing/探索式软件测试:基本概念介绍
http://www.hiadmin.org/testing/exploratory-software-testing/什么是探索式软件测试？
18、粗略介绍，和无法和实际项目结合实践。
19、很喜欢，简单易懂，还不错
20、后来发现自己竟然基本没有读过与工作相关的书。
21、很不错的图书，可以买来看看。
22、第一次买的书，没怎么看，书送人了。又买了一本，浏览了一倍，这本书感觉没什么层次感，博客堆积，不够深入。探索测试，更适合测试专家进行测试方法，普通的测试人员不适合。
23、故事形式。说了很多隐喻。
24、JW大师的作品，好玩，实用~ 若单元测试有了，手工测试可以玩许多深入的事情。
25、很有启发性
26、唯一一本权威
27、讲的东西大概是一种基于目的而不是基于模块的测试，根据粗略的测试脚本，根据特定的目的，现有测试后有脚本，然后列举了一系列的方向，感觉讲的很玄学，例子有点不好理解。
28、探索式软件测试
29、作者很强，翻译的也很好
30、怎么说呢~有几个方法在没看书前做实习QE时自己总结出来过~有开阔了不少新视野~就是理论太重，话说的有点太专业，偏繁琐。。。不过是本好书，短期内吃不下，只能慢慢随用随消化了。。。
31、传的神乎其神
32、不错的关于探索式测试的系统性介绍的书籍。对于提升软件测试的思路和技能有直接的帮助——

当然，前提是你得去实践。
33、内容应该还不错，但读起来很枯燥乏味！看着看着就想睡觉（可能是俺的水平差吧），能明白，但看不下去，组织得比较乱，不吸引人。
34、很好的总结，经验之谈，容易读懂，把我们经常会用的一些零散的思维捡起来了
35、看其中四五六三章即可，各种测试方法说的太简单了
36、看完之后大致知道软件测试是干嘛的了，就是找bug，提交bug。虽然我不得不承认软件测试很重要，想要做好软件的话软件测试尤其重要，但是我还是觉得软件测试没什么意思。就是一天测到晚把软件测死为止。属于重复劳动，没有创造性。不过说到底，除了艺术，谁又不是重复劳动呢，悲剧啊。
37、有些测试的思想在工作中一直在用，另些扩展自己的思路
38、很喜欢作者的叙事风格，像读小说一样，把软件测试说的很透彻。其实书里的很多场景我们从业人员都遇到过，只是没有总结过。书后面的也比较有意思，畅想一下软件测试的未来，对自己的职业有信心了。推荐购买！
39、好书,以前看过一点<How to Break Software>.有一些重复之处,不过,James A. Whittaker不愧是测试界的大师.
40、写给手工测试人员的，理论不错，但是写的有点啰嗦，干货不算很多。而且有点故意往理论上靠的感觉。 后面博客摘录部分感觉价值不太大。 14.08中
41、测试界大牛JW的书，里面体现的思路给人相见恨晚的感觉。探索式测试的策略、手工和自动化的对比、测试未来的发展，各方面都很贴合今天互联网行业的现状甚至很超前，而这是一本08年的书！再次感叹大多数野蛮生长的小作坊被国外大公司抛下了多少个时代
42、还没细看内容，质量很好
43、探索式测试，值得探索
44、灰常好的书，内容精要，知识概况精炼。
45、推崇手工测试的专家，整理分析阐述了探索式测试的理论方法实践和相关技术。自动化测试的特点和优势，发挥不同测试类型的优势，以及测试的未来展望。还有对于如何培养测试人员，测试工作发展的建议都有干货十足。软件测试领域的经典之作，值得学习和不时的研读。
46、相当不错，力荐。作者以轻快、客观的口吻严肃地谈了：如何去做软件测试。方法受益
47、还没看,不知道
48、换书用~
49、中国的软件测试发展还是在一个探索摸索阶段，看到此书的写成时间很惊讶，居然那早了一二十年的时间，同时也打开了一个脑洞，原来还可以这样来定义测试。
50、还行..
51、非常好的一本书. 思维又开阔了很多, 同时, 以前自己的经验和知识一下子被他人总结的如此的整齐, 难以置信. 原来测试的方式方法还可以如此般的描述并形成理论.
52、漫游测试模型下的探索式测试。本人的主要收获在未来测试、测试职业发展、未来测试工具等方面的理念，以及偶尔论及的只言片语上。

《探索式软件测试》

精彩书评

1、英文书评“好书！Whittaker讲述的概念有创意、巧妙、令人印象深刻。他真正懂得如何鼓励工程师们以不同的角度考虑软件测试。”——谷歌公司测试工程总监Patrick Copeland“James把美妙的手工测试方法学提升到了极致。漫游不仅概念正确，而且实际应用得很好，我们已经在所有测试人员的内部培训课程中分享了漫游的概念。如果你想把手工测试过程带到21世纪，请阅读这本书吧。”——微软公司卓越测试总监Alan Page“1990年，我开始与James在IBM一起共事。他早在当时就开始鼓励测试人员和开发人员大胆创新。通过这本书，他把对软件质量的热情提升到了全新的高度。请阅读这本书吧，见证你自己成为一个更好的测试工程师。James是这方面的权威，这个星球上的软件测试工程师和开发工程师，无论他们是真正关心软件质量，或者只是想在自己的日常工作中增加更多的乐趣，都应该阅读这本书。”——Cisco Systems公司高级工程主管Kaushal K. Agrawal“James Whittaker是测试领域中一位真正有远见的人士。uTest以及我们的QA专业全球社区经常向James咨询以获得他的灵感、他对测试未来趋势的解读和他的各种测试理念。现在他终于为大家写出了这本书，业界会由于这本书而变得更有智慧。”——uTest公司CEO和合伙人Doron Reuveni“只有像James Whittaker这样的人才会把旅游和软件测试以小说的形式结合起来，也只有James才能把它做到这么天衣无缝。漫游方法不仅提供了令人难忘的有效思考模式，还把适当的结构和组织与广泛的探索和创造结合在一起。bug们，小心啦！”——谷歌公司Alberto Savoia“James是有关软件测试的最好的演讲者之一，读他的书就像聆听他的演讲。如果你希望增加测试知识并成为一个更好的测试人员，这本书就是为你而写的。”——TCL集团公司主席和合伙人Stewart Noakes“我从事探索性测试已经有段时间了，James的漫游测试法给我所使用的各种方法起了名字，定义了测试重点，更重要的是给了我实际的指导。这本书将会使探索性测试的教学和应用变得更方便。”——iMeta Technologies公司高级测试顾问Rob Lambert“我为这项工作感到非常激动，它概念新潮却又合情合理。连我这样的普通读者都能轻松地理解和使用，而不用首先去学习一些华而不实的过时理论。在阅读本书时，我也从不需要借助字典。测试领域长久以来就一直期盼能出现这样的创新之作，我由衷认为本书在这方面走在行业最前沿。”——Netjets公司QA部门经理Linda Wilkinson

2、发现自己一直没写书评。总体来讲，这本书，内容很棒很详尽，是本值得看的好书。但严格来讲，个人认为书中内容很难讲就是探索式软件测试（简称为ET，Exploratory Testing）。业内广泛认为ET这个词是由Cem Kaner最早提出的，根据他的说法：ET是一种软件测试的风格（style），强调测试人员的自由权利和责任心，通过同时进行测试相关学习、测试设计、测试执行和测试结果解析这四种相互支持的活动，不断地优化其自身工作的价值。- 参考一：http://en.wikipedia.org/wiki/Exploratory_testing- 参考二：http://www.kaner.com/pdfs/QAIExploring.pdf （第36页）本书的作者，将探索一座城市比喻为测试一个系统，基于此来解释探索式软件测试。例如地标建筑，可以看做是该系统、该软件的主打功能，或者宣传重点；而风景名胜，则可以看做是该系统过去备受好评的那些消费者最爱功能；而地图，我们则可以看做是一份官方的需求文档；当然，还有很多可供查阅的民间游记，这则颇像是工作中的现行测试计划、测试用例或过往测试执行的测试报告等等。从此角度出发来看的话，那么本书的重点都侧重在讲，在开始探索之前，应该如何规划、计划这趟旅程。而这其中，未知的部分，并不多。Michael Bolton曾撰写过一篇著名的文章“Testing vs. Checking”，文中提出观点认为testing和checking是不同的，checking是为了确认已知事物的表现是否吻合预期，而测试则是为了找到新信息。- 参考：http://www.developsense.com/blog/2009/08/testing-vs-checking/- 详细：Checking is something that we do with the motivation of confirming existing beliefs. Checking is a process of confirmation, verification, and validation.- 详细：Testing is something that we do with the motivation of finding new information. Testing is a process of exploration, discovery, investigation, and learning.参照如上的这些观点来看的话，我认为这本《探索式软件测试》可以成为一本非常好的测试设计指南，但是，我很难认可它是一本真正在讲解“探索式软件测试”的书，因为它并未将其重心放在如何同步地进行学习、设计、执行和解析这四件事，而是如何规划自己的测试活动。==========徐毅：独立敏捷顾问，经验丰富的国内知名敏捷及精益教练，专注于敏捷软件开发、Scrum、敏捷转型、敏捷测试、测试自动化、robotframework等。

3、看了3次算是对里面的内容有了一个比较仔细的理解、同时也写了几篇关于探索式软件测试的文章http://www.hiadmin.org/testing/exploratory-type/探索式软件测试的四个类型http://www.hiadmin.org/testing/exploratory-testing/探索式软件测试:基本概念介绍

《探索式软件测试》

http://www.hiadmin.org/testing/exploratory-software-testing/什么是探索式软件测试
4、之所以打了4星，是跟我对本书的理解程度有关的。看第一遍，仍然问题一堆，说不出学会了什么。看第二遍，很多问题才得到了解答，但也只能说是大概理解了一些思想。也许以后看第三遍，并且在实际工作中有了应用后会有更大收获吧。作者的探索式测试方法是漫游测试（touring test），把对软件的测试过程比喻成在一个城市的旅游，城市中大大小小的景点就是你要发现的缺陷，不同的游览策略将指导你去往不同类型的景点。这也就是本书的核心思想：测试工作的划分，应该根据测试意图（即你想发现哪一类缺陷）而不是根据应用程序的结构关系。拿在北京旅游做例子，游览策略应该是这样的：地标建筑、古迹名胜、风景名胜、历史文化……而不是这样：海淀区、朝阳区、宣武区……书中的大多数文字，也就是在讲解如何才能根据测试意图来执行测试。具体的测试思路书中讲了很多，一定有一些方式是平时很少考虑到的。第六章有很多ET在微软实践的例子，需要好好体会。除了ET之外，作者关于测试工作本身和tester的很多思考和建议也非常值得细读品位
。----------摘抄一些重要文字和我喜欢的一些内容本书写的是一种我认为比其他任何缺陷都重要的特殊缺陷：即逃过所有各种检测手段而最终存在于发布产品中的缺陷。很多现代手工测试实践都缺乏目的性，随机性强且重复性强。使用探索式测试并不是说不写文档。测试结果、测试实例和测试文档都会在运行测试时创建。如果一个测试用例很可能马上就失效，当初就根本没有必要去编写它。探索式测试的缺点在于测试人员有可能在测试中没有重点，从而漫无目的地尝试各种情况来试图发现软件缺陷，这会浪费大量的时间。这里要强调指导方法的重要性。探索式测试如果没有一个好的指导方法，就好像游客新到一座城市，然后盲目彷徨想碰巧找到景点一样。从测试策略的角度来说，明确到底要测什么和怎么测试同样重要。探索式测试试图把制订计划、进行测试、重新制订计划等多个过程有机的结合起来，每次只前进一小步，但这每一步都是由软件过去和当前的运行状况、软件在测试时表现出来的各种行为和软件运行时留下的种种蛛丝马迹来即时确定的。探索式测试有下面几个目标：理解应用程序如何工作，它的接口看起来怎样，它实现了哪些功能。强迫软件展示其全部能力。找到缺陷。应该根据测试意图而不是根据被测应用程序的结构关系来划分。静态场景测试和探索式测试并不冲突。场景可以代表探索式测试的一个绝佳起点，探索可以给场景加入宝贵的变化，否则场景将很有限。成功的漫游测试会刻意揭示某一类型的缺陷。附录：无论你曾有过什么样的实践经验，只有在你必须教授某一学科时，你才能真正掌握它
。

《探索式软件测试》

章节试读

1、《探索式软件测试》的笔记-第1页

探索式软件测试书摘（James A.Whittaker）
http://blogs.msdn.com/b/james_whittaker/archive/2008/12/09/google-v-microsoft-and-the-dev-test-ratio-debate.aspx

探索式测试（exploratory testing）是一种自由的软件测试风格，强调测试人员同时开展测试学习、测试设计、测试执行和测试结果评估等活动，以持续优化测试工作。
软件测试风格rather than 具体的软件测试技术

探索式测试的分类：
自由式的ET
基于场景的ET
基于策略的ET
基于反馈的ET

软件缺陷的根源：程序员引入的根源和运行环境导致的缺陷

软件测试的决策有5部分：输入，状态，代码路径，用户数据，执行环境
输入：什么是输入？合法与非法输入？开发人员定义错误处理程序的三种方式（输入过滤器、输入检查、使用异常）

状态：什么是软件状态？用户的输入（不同的输入，不同的输入顺序）导致软件状态的改变，一定要注意观察状态的改变。

漫游测试（与场景测试相对）
我们将软件特性分成了：商业区、历史区、旅游区、娱乐区、旅馆区、破旧区
商业区：用户所要使用的软件特性和功能。
历史区：历史的版本遗留的代码
旅游区：有些特性和功能对新用户非常有吸引力，然而老用户不经常使用的部分

测试方法：
出租车测试法（出租车禁区测试法）联系打的理论，坐公交理论，类似旅行计划的制定
取消测试法
破坏测试法
遍历测试法
超模测试法？针对用户界面的优秀漫游测试法
极限测试法
深巷测试法

漫游与测试中的棘手问题
漫无目的
重复性
暂时性
单调性
健忘性

《探索式软件测试》

手工测试主要有两种类型：
1.基于脚本的手工测试
2.探索式测试特别适合于敏捷开发（agile development）

Problem：
1.哪种类型的代码比较适合使用自动化测试，哪种类型的代码比较适合手工测试？从理论的角度解释问题
2.在自动化测试中，哪种类型的软件缺陷比较容易被发现？又哪种类型的缺陷不容易被发现，for example.
3.ET的缺点？优点？如何进行ET工作？
4.你自己的测试方法和哲学是什么？脑力风暴
5.测什么？如何测？测试的分类？测试的策略选择？
6.测试人员如何记录应用程序的哪些部分已经被测过？列举至少四条标准作为测试人员衡量测试完整性的基础。
7.测试的最终目标是找到软件的缺陷，但同时也应该让测试更高效，测试周期更短。
8.我想知道你们公司是如何评估测试人员的？
9.虚拟化技术的软件测试中的应用？
10.如何阻止优秀的测试人员转而投向开发工作？

软件测试的真正价值并不是体现在代码中找出了多少缺陷，而是发现设计和编程人员解决问题方法上的局限、思路中的狭隘以及技能方面的不足。

手工测试人员善于成为问题领域的专家，善于分析业务逻辑错误。自动化测试擅长低级别的细节。自动化测试可以检测到崩溃、挂起、不正确的返回值、错误代码、突发异常、内存使用情况等。选择什么样的测试取决于希望找到什么样的软件缺陷。大部分时间里，
手工测试在寻找业务逻辑错误上优于自动化测试；而自动化测试在寻找基础结构性软件缺陷上胜过手工测试。

什么时候能让软件测试就像玩电子游戏一样，充满乐趣呢！

==========
1. 简要说明什么是ET

就是在完全不熟悉项目业务需求的基础上，采用边
学产品知识，边测试，通过一些手段来操作产品，使其暴漏出一些隐含的问题。其测试执行思路与测试设计思路是同时进行的。一个很明显的Freestyle ET方式。

2. ET 测试的范围

由于大部分项目存在一些共性，ET 测试的范围一般是主要的功能的实现，再加上主要的功能中隐含的一些潜在的风险，例如超长输入引出的系统错误等。具体可参见ET实践流程。

3. 为何要做ET

至于做 ET实践的原因多方面：
---------目前项目测试人员的功能测试手段太单一
---------目前第3轮测试发现的bug率以及投资回报率很低 淘宝网-探索式测试白皮书
---------为了质疑目前测试部三轮测试的流程规范
---------国外已经有了比较成熟的ET理论和实践经验
---------创新并实践前段时间ET的理论学习

4. 什么时候开始做ET

根据 ET 测试的方式和目的以及时间安排，可看出 ET 并不是为了发现主要功能的流程问题。所以特别需要在相对稳定的系统上做 ET，这里有两个好处：一是由于 ET 测试人员没有项目测试人员对需求了解深入，对于主要功能的流程问题没有项目测试人员发现那么及时以及深入。二是在稳定的系统上做ET，有益于发现项目测试人员的盲点，以及发挥测试的极限测试手段，同时也有益于ET测试产出的效果。所以在 XX1 项目 ET 实践过程中，是在第二轮测试的最后一天开始 ET。一般是在安全测试通过后。因为安全测试的 bug 修复后会引发比较多的页面 bug，此在一定程度上会影响 ET发现较严重的bug数量。

5. 怎样做 ET

ET 过程中使用到的一个非常清晰的任务列表，指出了要测试什么，怎么测试（强调策略，不是详细测试步骤），要寻找什么样的bug，有哪些风险，要去检查什么文档等。

根据国外 ET 实践理念，采用 Session 来进行测试范围的确定(具体请看 ET 的管理)，下面是简单的一些说明：
第一步: 大概花1-2 个小时时间看PRD和原型(了解目的和产品背景)。
第二步: 大概花1-2 个小时时间确定下有哪些主要的功能模块和贡献性的功能模块。
第三步： 与项目组测试人员沟通哪个功能模块发现bug最多，哪个功能模块发现bug最少，哪个模块存在风险比较大。
第四步：根据前几步情况和参加ET的时间段来确定有多少个Session，并指出每个Session大概花多长时间。一般是1.5-2 个小时。就淘宝而言，一个 Session大概是2-3个 UC 的情况。
第五步：制定ET 测试计划，包含所有Session的名称和测试时间以及缓冲情况。
第六步：根据 ET 测试计划，边学习产品需求，边测试。发现问题立马记录问题描述。最后发送 ET 测试报告。
第七步：与项目组测试人员沟通ET的效果以及该产品存在的风险，从用户易用性角度给该产品总体评价，同时跟踪确认bug的fix情况。

6. 做 ET时注意什么

ET
测试人员需要以最少的时间去了解这个产品的某个需求，而不是不要花很长时间去了解某个复杂业务的具体实现过程。

关注细节的部分，多使用一些极限测试的手段，比如超长字符，非法字符，异步编辑等。被某个细节block时，及时与开发人员沟通。 发现一个疑似问题，立马记录其问题描述。全神贯注的进行边学习

产品，边测试。

7. ET 产出了什么
8. ET 发现了什么样的bug
对于 ET 测试发现的bug类型做一些个人分析，相信很多人对于这个比较感兴趣：
第一：一般情况下，ET测试的目的不是为了发现正常流程下的主要功能bug，特别是 Freestyle ET 方式，其实践的时间点在第二轮测试结束后 淘宝网-探索式测试白皮书
第二：ET 测试过程中，会变化测试手段去测试，更多的是用户测试和极限测试和交叉测试，也就会发现很多这些手段产生的bug
第三：ET测试过程中，使用的一些测试手段决定了发现的bug类型，常用的测试手段有：边界值测试，极限测试，用户测试，菜单浏览，域测试，组合测试。
第四：ET测试过程中，需要在学习业务的同时，不断的变化上述的测试手段去测试(关键点)。
第五：从产出上来看，使用了极少时间去进行 ET 测试产生的 bug 率是比较高的(平均每个小时产生3.5个 bug)。对于项目组测试人员来说，其单位时间内产生的bug率应该是比较低的。因为其投入包括熟悉需求；测试设计，3 轮测试执行等。具体官方数据个人不是很清楚。可参照 XX1 项目某测试人员的数据：投入时间267 个小时，发现124 个 bug；则平均每个小时产生0.47 个 bug。(注意：该数据不是直接判断标准，只做参考)

2、《探索式软件测试》的笔记-第74页

我们测的路径越多越好。

讲述用户故事
描述需求
演示产品功能
演示集成场景
描述设置和安装
描述警告和出错情形

3、《探索式软件测试》的笔记-第24页

测试用例划分优先级，优先保证基本功能可用。工作也是一样，每天计划好8小时的时间，划分优先级，每天充电

4、《探索式软件测试》的笔记-第28页

正常输入和异常输入，很多人忘记关注逆向测试，很多开发不喜欢写错误处理，想办法让系统崩溃也是很重要的测试

5、《探索式软件测试》的笔记-第28页

所有的软件，它们可能运行于完全不太的环境中，但是它们都会执行四个基本任务，即接受输入、产生输出、存储数据和进行运算

6、《探索式软件测试》的笔记-第50页

各个区域：商业区，历史区，旅游区，娱乐区，旅馆区，破旧区

商业区测试类型：
指南测试法 要求测试人员通过阅读用户手册并严格遵照手册的建议执行操作。

卖点测试法 找到那些最能卖钱的特性进行测试

地标测试法 先选择地标，然后在软件中执行类似于在森林中地标间跳跃的动作

极限测试法 向软件提出很多难以回答的问题。如何使软件发挥到最大程度？哪个特性会使软件运行到其设计极限？哪些输入和数据会耗费软件最多的运算能力？哪些输入可能欺骗它的错误检测例程？如果软件用于产生某些特定输出时，使用哪些输入和内部数据可以不断挑战软件的这种能力？

快递测试法 专注于数据，确认哪些被存储起来的输入数据并“ 跟随 ”它们走遍软件

深夜测试法

遍历测试法

历史区测试类型：
恶邻测试法 发现和报告越来越多的缺陷后，就可以把缺陷数目同产品特性联系起来，从而可以跟踪究竟在哪些地方会出现产品缺陷。

博物馆测试法 老代码

上一版测试法

娱乐区测试类型：
配角测试法 专注于某些特定的特性，虽然不是用户使用的主要特性，却容易被忽视。

深巷测试法 最不可能被用到的或是那些最不吸引用户的特性

通宵测试法

旅游区测试类型：
收藏家测试法 收集软件的输出

长路径测试法 在到达目的地之前尽量多地在应用程序中穿行，选择长路径而不是短的，把那个埋在应用程序最深处的界面作为测试目标

超模测试法 重点在于测试界面

测一送一测试法 运行多个拷贝

苏格兰酒吧测试法 特别适用于大规模的复杂应用程序，在这些应用程序中的有些地方，测试人员需要事先知道如何去找到它们

旅馆区测试类型：
取消测试法

懒汉测试法 尽量少的做实际工作，接受所有默认值

破旧区测试类型：
破坏者： 利用各种机会破坏

反叛测试法 逆向测试法 输入最不可能的数据
歹徒测试法 输入一些不应该出现的数据
错序测试法 以错误的顺序做事情

强迫症测试法 重复反复做

7、《探索式软件测试》的笔记-第30页

输入筛选器，需要验证：非法的进不来；合法的都在里面；能否绕过筛选器

8、《探索式软件测试》的笔记-第17页

使用探索式测试并不是说不写文档。测试结果、测试实例和测试文档都会在运行测试时创建。这和普通测试在测试计划里预先编写好截然不同。用于记录探索式测试结果的最佳工具就是那些截屏软件和记录击键的软件
探索性测试的缺点在于测试人员有可能在测试中没有重点，从而漫无目的地尝试各种情况来试图发现软件缺陷，这会浪费大量时间。从测试策略的角度来说，明确到底要测什么和怎么测试同样重要

9、《探索式软件测试》的笔记-第162页

算法、计算复杂度、图论、数据结构背景知识都是必不可少的技术。

10、《探索式软件测试》的笔记-第29页

开发人员有三种基本方式定义错误处理程序（error handler），它们分别是输入筛选器（input filter）、输入检查（input check）和使用异常（exception）

11、《探索式软件测试》的笔记-第21页

目前看完了第二章，了解了什么是局部探索式和全局探索式测试的定义。以前不知道什么是探索式测试，看完了前两章后回想了下，感觉像是为以前做过的一些测试工作找到了理论依据。
当时我做测试的思路是按照定好的测试计划来做测试，在case目的不变的基础上可以进行自由发挥，因此发现了一些不易被发现的bug，当时颇感欣喜。现在发现，这样做，确实有点贴近书上描述的思想，但是并没有书中说的那么详尽具体，更不及书中规范。所以，还得接着学习。
希望这本书后面的内容能带给我更多收获。
（不知道为啥总感觉这本书或多或少有点啰嗦，我更开门见山直奔主题的感觉~~#）

12、《探索式软件测试》的笔记-第28页

如何测试用户输入
合法输入和非法输入
输入筛选器
输入检查

异常处理代码
常规输入还是非常规输入
默认输入或用户提供的输入
使用输出来指导输入选择

如何测试软件状态

代码路径

用户数据

运行环境

13、《探索式软件测试》的笔记-第41页

输入 状态 路径 数据 运行环境

《探索式软件测试》

版权说明