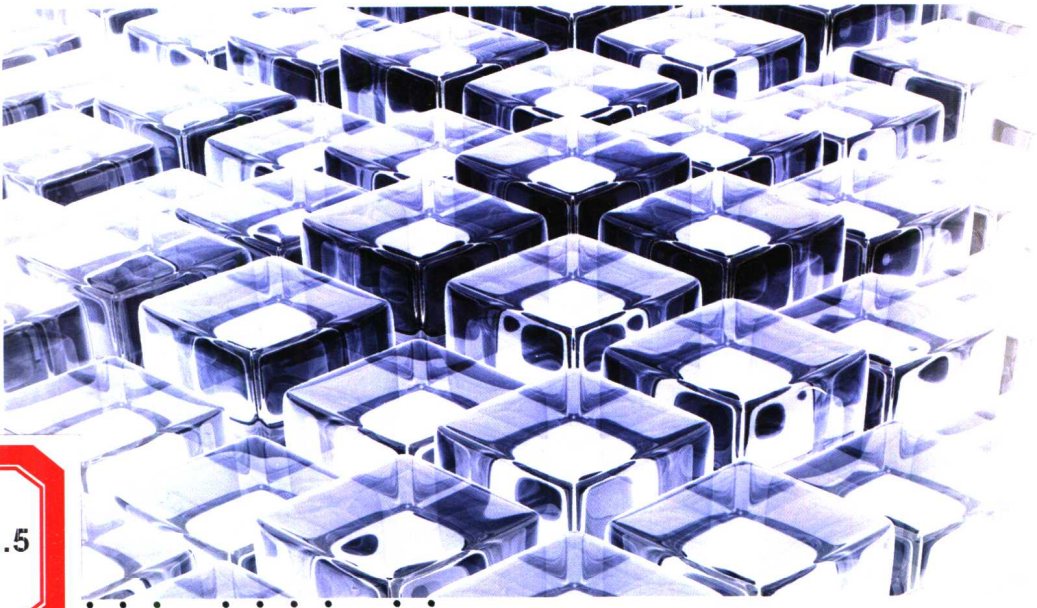
软件测试与Junit实践

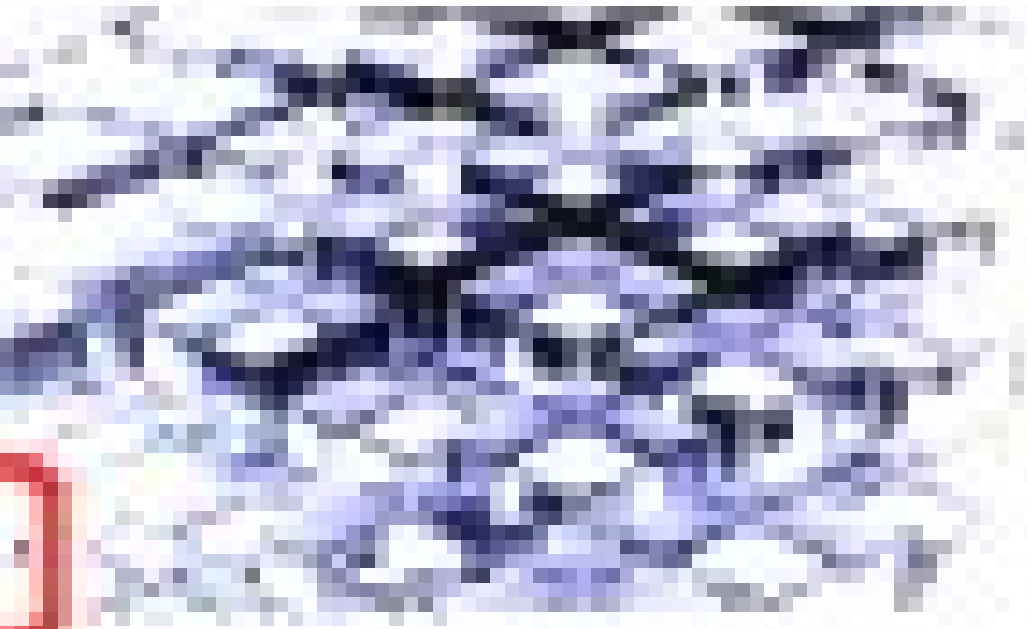
王东刚 编著



人民邮电出版社
POSTS & TELECOM PRESS

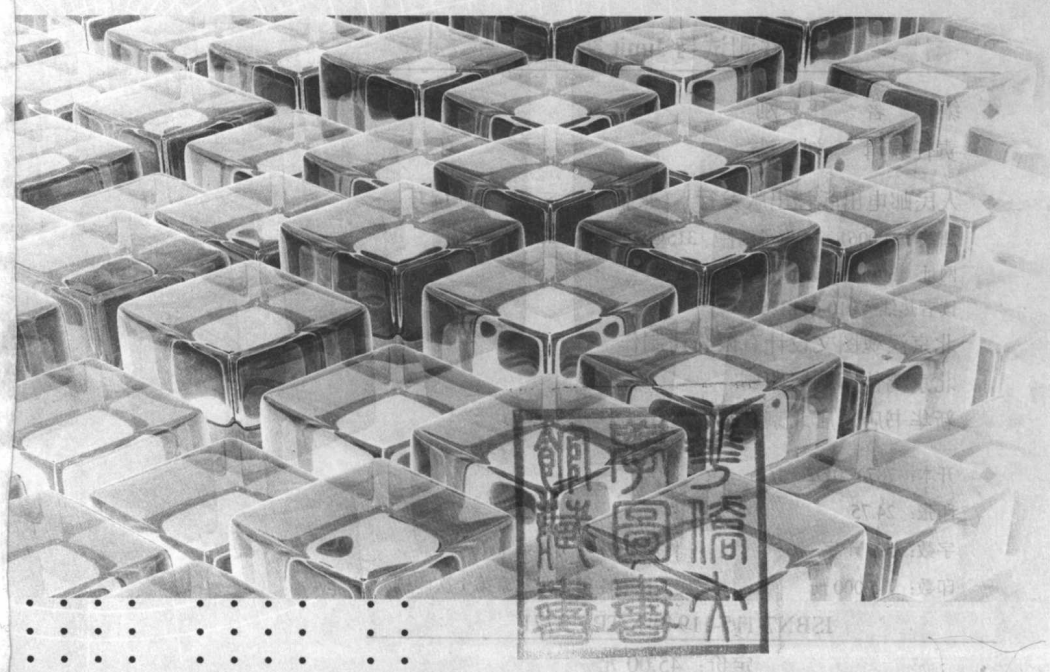
软件测试与Junit实践

[redacted]



软件测试与Junit实践

王东刚 编著



人民邮电出版社

图书在版编目（CIP）数据

软件测试与 Junit 实践 / 王东刚编著. —北京：人民邮电出版社，2004.1

ISBN 7-115-11949-X

I. 软… II. 王… III. 软件—测试 IV. TP311.5

中国版本图书馆 CIP 数据核字（2003）第 106276 号

内容提要

本书是为软件测试领域相关人员编写的。前 3 章重在培养读者的软件测试意识。第 4 章是一个特殊的测试流程介绍。第 5 章详细阐述了类测试思想。第 6~9 章偏重于测试技术应用，着重介绍一款 Xunit 工具，并且加入了实际例子进一步告诉读者如何更好地应用。第 10 章介绍了测试后 BUG 的处理手段和相关处理流程。在附录中还有作者的感悟和与测试相关的书籍的评价。

本书的特点在于“学以致用”，从测试的实际应用出发，结合大量的测试例子，使读者在测试层面上有更大的提高。

本书适合于实现人员、设计人员、测试人员以及测试管理者参考使用。

软件测试与 Junit 实践

◆ 编　　著　王东刚
责任编辑　刘　浩

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67132692
北京汉魂图文设计有限公司制作
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：24.75
字数：597 千字　　2004 年 1 月第 1 版
印数：1-5 000 册　　2004 年 1 月北京第 1 次印刷

ISBN7-115-11949-X/TP • 3771

定价：45.00 元

写在前面的话

“可悲，那是因为我要面对太多错误的事。冰冷，那是因为我无法顾及别人的情绪。孤独，那是因为我住在一个边界之地。身后是无法穿透的黑暗，我守望着或者不停撕裂着错误的隔膜，阻挡无知的人继续迈向界后的深渊。

—— 创世纪寓言”

最近和公司同事 nickle 做了一次彻夜长谈，他在公司专职软件设计，使用 UML 语言绘制出软件模型然后转交给程序员进行编码。当我听他说曾经专职做过一年软件测试时感到非常吃惊，下意识地追问他为什么放弃测试工作改行为软件设计了，nickle 是这样回答我的。

“软件测试这个行业在国内太新了，可以说什么都没有。找些相关的资料都比较困难，挂上 Google 搜索的全是英文的，如果找到了中文的你会发现它已经失去了实际应用价值”。他继续说：“比如你可以制定很多的规范来强制别人对你的输出(产生很多的标准输出工件)，可你有细致的、能控制的测试后续流程吗？假设你有细致的、能控制的测试后续流程，可你的管理层能给你充分的资源来支持你的工作吗？假设你的管理层全心全意地支持你开展工作，可是相关人员能否正确理解你的意思，做好配合呢？就算上面的条件你都具备了，可你是否有勇气对你测试后的项目质量负全责？”

想到自己，也曾经在很深的夜里睡不着，是因为无法理解软件测试的真谛而在空旷的街道上满无目地行走。也曾经和一些同事激烈争吵过，是因为自己制订的一些标准输出不被别人重视。也曾经为自己的工作胆战心惊过，是因为自己负责测试的项目出现了严重的质量纰漏，更不用说忍受过多少次责骂和贬低了。可是在这一行中，还有多少人因为前途迷茫和无助而退出呢？正为“世间本来没有路，走的人多了就成了路”，所以我鼓励所有的现在正在从事软件测试工作的人不要因为别人的偏见、指责和不理解而放弃这项工作。

我跟别的同行交流听到过这样的话“你连编码都不会只能去做软件测试了”、“我为什么要请这么多的测试人员，一个就可以了”、“软件测试不就是把软件用一遍这么简单吗，有什么困难的？”、“我刚毕业，没有什么实际经验所以就做测试了”、“公司如果要裁员，先把测试部的人裁掉吧”等。我的同行 james 对我说：“软件测试已经成为了软件工程中最低级、最简单、最不需要知识积累的代名词。”在很多场合，软件开发者和项目管理者都将测试工作看成一件可有可无的事情。下面是我和一个程序员在工作上发生的冲突：

我：seancles，我认为你应该暂停一下你手中的开发工作。你写的某些代码出错频繁，已经严重妨碍我们继续对它测试下去了，我建议你好好检查一下你自己输出的代码。

Seancles：我没时间顾及这么多的事情，项目时间太紧张了。再说，你们做测试工作的这么空闲，不就是刷新一下页面、点击几个按钮这么简单吗？这样吧，你们帮我修改代码吧，也算是给你们增加一些工作量。

我：测试并不包括修改程序，我还是建议你好好检查一下你书写的代码。我认为有些问题在你这里是可以避免的。

Seancles：行了，行了。我现在很忙没有时间，如果你对我有什么意见可以跟我的项目经理反映……

你是不是也碰到过这样的事情？我认为这是一个意识问题，我们需要改变人们对测试工作的态度，需要改变人们对测试工作的看法。能够让人们从开始就尊重软件测试工作，理解软件测试是在帮助他们，确保他们开发出来的产品是正确的、安全可靠的，这才是当前软件测试最需要解决的问题。

人总是狂热地追求把非理性的、不可控制的现实行为用逻辑的方法体现出来，在这两种状态转换的时候就会出现很多可预见和不可预见的问题。从这里你就大概可以知道问题出现的根源了，你甚至可以这样说“因为人的存在，这个世界才出现了很多可视和不可视的错误”。曾经有个程序员很快写了这样的一段代码给我，说如此简单的代码肯定不会存在什么错误了。

```
public class MyTest{
public static main(String[ ] arge){
System.out.prinnln("Hello World!");
}
}
```

代码确实很简单，但是我看了一下马上给他指出了很多存在的问题。第一个错误，代码中没有看到任何注释性语言，虽然代码简单但还是应该添加注释语言。第二个错误，源文件中没有使用 package 关键字，可能导致 JVM 无法正常编译源代码。第三个错误，没有指明 System.out.println()方法引用自哪个 Java 类，虽然 JVM 在编译的时候会自动添加一些 Java 基类，但我还是怀疑 JVM 无法正常编译源代码。第四个错误，书写方式没有符合 Java 编码规范，所有的类文件开始字母都应该是大写的。第五个错误，书写格式混乱，这样的格式不适合后阶段维护人员的工作。第六个错误……可以这样说“再简单的事物都存在着错误，仔细检查总会发现很多存在的问题”。

有时公司的管理层或者其他部门也不是很明确我们所做的工作，因为客户的指责而把这种愤怒迁于我们，产品经理 tyro 就和我有过这样的一段对话：

Tyro_tan：为什么经过你们测试的软件到了客户那里还是出了不少问题？客户现在非常不满意，请给我一个合理的解释！

我：没错，我们是很认真地测试过这个项目了，但是我们不可能将测试覆盖到项目所有的方方面面。

Tyro_tan：这是什么意思？

我：测试的范围和详细程度受制于资源约束，在这里资源可以是时间、投入的人力等，让我们的测试覆盖整个项目这不实际。我们只能使用有限的资源去测试，出现上面的情况是正常的。

Tyro_tan：难道公司没有给你这些资源吗？我知道你有不少的人手呀。

我：这个项目从开始到发布只给了我们一个星期的测试时间，但是根据我们对项目的量化，要在这样的时间内做好测试最起码需要 15～17 个人手，而我没有。

Tyro_tan：你跟我说这些没用，现在软件出了问题你快点派人处理……

我：但我还是需要这么多人力呀？

Tyro_tan：我和你一起跟公司管理阶层反映，尽快招聘一些人来做这些事情。这个项目已经来不及了。

事实上，我们可以从上面的对话中找出许多的错误的根源，这里包含了多种不成熟的反馈。

（1）不成熟的预测：项目在开始之前就要详细描述项目执行和发布后可能遇到的风险，显然我们没有建立良好的风险回避机制。一旦出现了严重的突发事件，我们往往被迫选择这种临时而不负责任的短小会晤。

（2）不成熟的度量：各个控制层面在项目的开始阶段竟然不能做到对各阶段工作量的有效度量，更别说对资源投入的曲线递增计划。我们从来都是羡慕维尔奇在通用公司的成功，可我们从来不去仔细研究这种成功背后的大量实际数据和维尔奇的度量公式。

（3）不成熟的客户：也许我们没有告诉客户一个现实，即使软件已经递交给他们，可是测试工作并没有因此而结束。这同样说明了我们对客户前期、中期培训工作毫无效果，可能公司本身就没有客户培训这种意识。就算项目完全失败了，他们还是一味地责怪自己的团队出现的漏洞足可以开过一辆坦克车。

（4）不成熟的团队：现在软件工程讲究的是团队协作，如果是一个工作状态优秀并且完全投入的团队，就有可能在恶劣的环境中创造出美丽的巴比伦花园。团队是什么，就是既能保存团员个性又能充分体现团队共性的一个战斗体，天生就是为了战胜艰难而存在的。当然，团队和伪团队（小组）是有明显区分的，团队为一件事情的后果承担共同责任，而伪团队则是某一个人为全部的事情负责。

（5）不成熟的交流：A 团队不知道 B 团队究竟在做什么，他们很恼火 B 团队不能很好地配合他们的工作。同样，B 团队也很恼火 A 团队混乱的工作方向，索性坐下来看一段时间再说。这就是交流和相互扶持的绝对失败，A、B 团队在一起的作用已经和他们建立时的初意完全相反了。

（6）不成熟的认识：很多情况下我们总是做事后诸葛亮，以为在已经损坏的项目中继续投入人力也许会完全挽回所有损失。这就是战争，大量的幼稚人群被胡涂的管理层派往战场，可他们会惊奇地发现，事情并非好转而是变得越来越糟糕。

（7）不成熟的我：我不能很好地面对现实，我并没有为自己的错误而羞愧地检讨自己，而是盲目地跟从管理层的意思，眼睁睁地看着事情变得越来越糟糕。就是有太多这样的我，才造成了现在混乱无序的情况。

即使我们做好了所有的事，也不代表我们就作对了所有的事，还需要知道：任何严格的测试都不可能发现软件中存在的所有隐患。做测试之前就需要正确认识这个问题，有谁能看到自己的背后呢？当然这并不是说我们测试无效，最多是说我们认识不够。所以在测试的事后需要采用更多的发现手段，这里的发现手段就是不同的测试方法。没有一种测试方法可以发现所有问题，然而多种测试方法组合就能发现更多的问题，如图 1 所示。

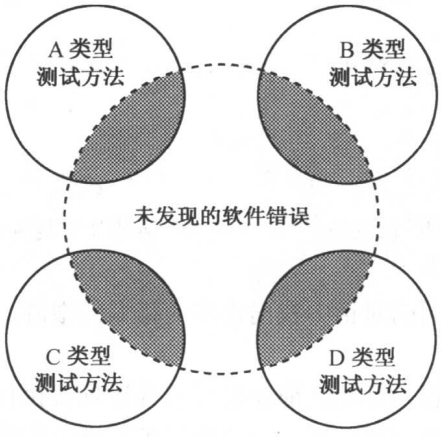


图 1　不同的测试方法和发现问题的作用域

不管任何测试方法都有其针对性，类似于图 1 所示：A、B、C、D 四个圆环（代表 4 种不同测试类型）和一个虚圆环（代表软件中存在的缺陷和错误）交汇，交汇区域代表已经被发现的软件缺陷和错误，也可以称为 4 种测试类型的有效域。虚圆环的空白代表未发现的软件错误（一个未知的 X 错误域），4 个圆环的空白则表示不同测试类型的失效域（对测试方法的使用不当或者存在陌生操作等原因而造成）。很遗憾，X 错误域根本无法度量，就像我们无法预知黑洞的背面一样。但我们必须正视它的存在，尽量提高各种测试类型的有效性，减少 X 错误。

从事测试工作这么久，我一直想把测试这种抽象的活动用图形表现出来，如图 2 所示。

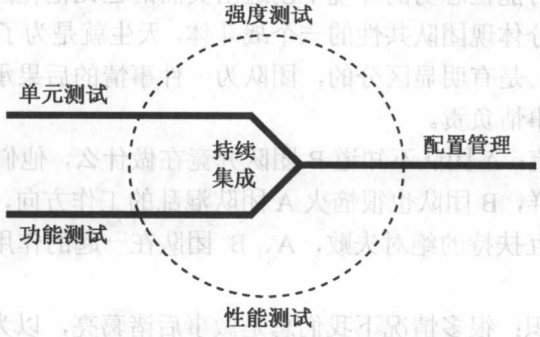


图 2　测试活动的抽象图形表现

图 2 看上去像个“二极管”电路，其中的单元测试保证程序的内在高品质，功能测试保证程序的外部功能正确。在不断的持续集成的动作中进行强度、性能方面的测试，保证程序中所包含的部件，以及各个部件之间的高稳定性，达到最佳优化状态。最后的配置管理在测试动作后及时更新软件的版本，推动整个产品（或者项目）向良性的道路发展。

刚进入软件测试行业的人总是混淆质量保证和测试，其实这两种行为是不同的。质量保证是全局的，而软件测试隶属于质量保证，是一种积极的实际数据采集活动。换一句话说，测试无法改善软件的质量，它不过是一种事后的监管手段。天下没有不漏水的桶，发现漏水的桶是测试的事情，尽量让生产出来的桶不漏水则是质量保证要做的。

许多新入行的人员总是不停地向别人索取文档（比如测试流程和各种阶段产生的规约），

入了行的人员则每天抱怨自己辛辛苦苦做的东西不能很好地用在实际工作中去。测试是一门思想，别人的思想在你这里运用可能是一种灾难，更何况测试是一个整体，需全盘运用。实施测试则是一种策略，不能开始的时候就把所有的事情都做好，需要采用渐进的、迭代的手段去实现，慢慢向别人渗透你的思想。你可以采用如下的改进模型，如图 3 所示。

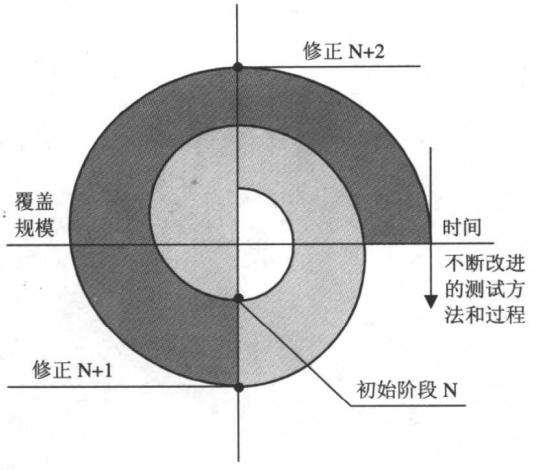


图 3　测试不断改进模型

初始阶段我们采用一些非常简单的测试方法和模糊的测试过程，也许那个时候的工作输出很糟糕，但不要气馁，应该坚持改进。N + 1 次修正后，我们清晰了测试过程。N + 2 次后的修正，我们强化了集成测试和系统测试的部分手段。N + 3 次后的修正，我们加入了严格的单元测试，并且重新修正了整个测试过程。持续 N + I 次改进，我们的工作就会渐渐接近完美。走第一步总是最困难的，但却是最值得纪念的，请走出属于你的第一步吧。

本书第 4 章的敏捷测试过程奉献给所有在测试前沿奋斗的同行们，这个测试过程能让你在资源严重受限的环境中最大限度地保证测试执行质量。其实创造这个测试过程是违心的，一方面它体现了我们低劣的质量管理意识，另一方面它体现了我们当前脆弱的项目管理体系。这里尽可能地把这个过程细化，让你明白每段的实际作用。如果你能把它融入到自己的测试体系中，并且能把使用后的效果反馈给我那就太好了。

书里还包含着我在这个行业的一些经验，当然也有我到今天还无法解答的疑问。我认为每本书的作者都应该积极地和读者们去讨论观点，而不是单纯灌输。写书是为了阐述自己在这个领域里的一些观点和结论，是否正确，则需要大家来评判。我虽然在这本书里使用大量的实际案例，但是我还是要说，这是一本思想书，说明我的测试思想，以及为什么要这样做。你能看出了很多疑问，并产生新的观点，而且渐渐形成自己的测试思想，这才是我写这本书的最终目的。

作者

目　录