第一部分　写作线索

本部分为全书制定的写作线索。第1章讨论最近几年软件行业的变化，以及这些变化对软件质量提出的要求。本书将这些要求转换为十一个基本原理，以此作为本书其他各章的中心。第2章定义验证、确认、质量保证和质量控制等关键术语，然后详细讨论各种生存周期模型，以及这些模型对验证和确认活动的意义。本部分还要介绍关于进入和退出准则的概念，这对于理解书中后面关于测试的不同阶段是非常必要的。

第1章 测试原理

1.1 生产软件中的测试背景

我们今天使用的几乎一切东西都包含软件。在软件发展的早期，软件用户的数量与大公司相比还是很少的。现在，一个典型的工作场所（或家里），差不多每个人都在使用计算机及软件。管理人员使用生产率很高的办公软件（代替以前的打字机）。会计师及财务人员使用电子表格软件和其他财务软件包，比使用计算器（甚至手工）要快得多。公司和家里的每个人都用电子邮件和互联网进行娱乐、教育、通信和交互，获取任何想要的信息。另外，“技术”人员使用程序设计语言、建模工具、仿真工具和数据库管理系统完成以前主要靠手工完成的任务。

除了上帝，我们都要测试

上面只是说明软件的使用对于用户来说“很明显”的几个例子。但是，软件的无处不在和广泛普及远不止以上这些例子所揭示的那样。现在的软件就像20世纪初的电一样普及。我们在办公室和家庭所使用的几乎每一台设备都嵌入大量的软件，例如手机、电视、手表和冰箱以及厨房的每一件电器都有嵌入式软件。

另一个值得注意的现象是软件在任务关键场合的使用，在这些场合出现失效是根本不能接受的。对于心脏起搏器软件，决不能提议“请关机并重启系统”！我们离不开的几乎所有服务中都有软件。银行、航空管制、汽车等，驱动它们的软件都是绝对不能失效的。这些软件系统必须每时每刻、永久、可靠、可预见地运行。

这些无所不在、广泛使用和关键之处都对软件的开发和部署提出了一定的要求。

首先，开发软件产品或提供服务的公司必须尽全力减少、最好消除每件所交付的软件产品或服务中的缺陷。用户越来越不能容忍劣质的软件产品。从软件开发公司的角度看，发布有缺陷的软件产品在经济上也不是可行的。比如，在电视机发运给成千上万的用户后，其中的嵌入式软件被发现有一个缺陷。怎么可能发送“补丁”给这些用户，要求他们“安装补丁”？因此，唯一的解决方案是在产品交付用户之前就一次做好。

第二，缺陷不可能隐藏很久。当用户数量有限，并且使用产品的方式是可预期的（高度受限的），软件产品中存在的缺陷可能发现不了，并隐藏很长时间。但是，随着用户数量的增加，缺陷不被发现的几率越来越小。如果产品中存在缺陷，总有一天有人会发现它。

第三，一个软件产品或服务的使用特性正在变得越来越不可预知。如果软件是为一个特定公司的特定功能（例如工资管理软件包）专门开发的，那么产品的使用特性是可以预期的。例如，用户只能执行定制软件所提供的特定功能。此外，软件的开发者了解用户、了解用户的业务活动及操作。另一方面，请考虑驻留在互联网上的一般应用程序。开发者没办法控制用户的操作。用户可能运行没有测试过的功能；可能使用的是不合适的硬件或软件环境；可能没有受过完整培训就直接以不正确的方式或没有目的地操作软件。尽管存在所有这些“误操作”，软件应该仍能正确运行。

最后，对每一个缺陷的结果和影响都要进行分析，尤其是关键任务应用程序。软件发布时可以接受的是99.9%的缺陷修复率，只遗留0.1%的缺陷。看起来对于发布的软件这是一个不错的统计数字。可是，如果我们在关键任务应用软件中留下了0.1%的缺陷，那么统计数据将如下所示。

- 每周会有10 000个不合格的外科手术。
- 每天会有3架飞机坠毁。
- 每周有5个小时没有电。

确实，上述数据对于任何个人、组织或政府都是不可接受的。我们提出一种补救措施，例如“万一着火，请穿这件衣服”，或者在文档中写明失效，例如“假如飞机着陆有误，你可能只会损失部分器官”，这些对于关键任务应用程序来说都是不可接受的。

本书关注软件测试。在传统意义上，测试被狭义地定义为测试程序代码。我们更愿意采用一种更广义的测试定义：它包含所有生产优质产品的相关活动。生产软件产品必须有几个阶段（例如需求获取、设计和编码），除此之外还有测试（传统意义的测试）。虽然测试肯定是促成产品高质量的因素之一（也是一个阶段），但是测试本身却不能提高产品的质量。对于一个好的产品，测试与其他阶段恰当的交互是必不可少的。图1-1描述了这些交互及其产生的影响。

如果其他阶段的质量较低，测试的有效性也很低（图1-1左下角的情况），这种情况是不能持久的。因为这样的产品将很快被淘汰。其他阶段质量很低而试图通过加强测试来弥补（图1-1的左上角）可能增加每个人的压力，尤其是产品将要发布时发现了缺陷。类似地，盲目地相信其他阶段的高质量从而进行较少的测试（图1-1的右下角），当最后一刻发现缺陷时将导致不可预见的危险情形。理想的状态是包括测试在内的所有阶段的高质量（图1-1的右上角）。这种情况下，用户会感到高质量带来的好处，这将激发团队更好地工作，公司获得成功。

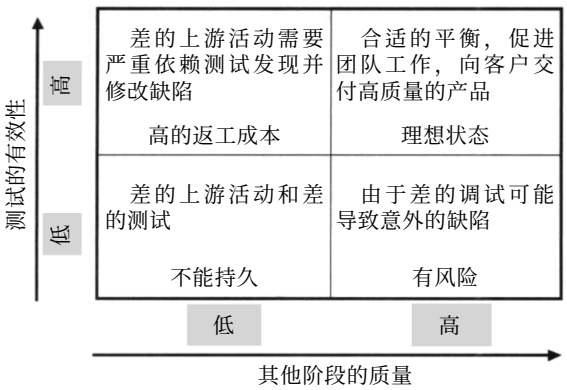


图1-1 测试的有效性与其他阶段质量的关系

1.2 本章介绍

本章讨论测试的一些基本原理。我们认为这些原理对于理解测试目的，给用户提供高质量的产品来说是十分重要的。这些原理同时构成本书其余部分的基础。本章是本书的核心。

测试的基本原理有：

1. 测试的目标是在用户发现缺陷之前找到它们。
2. 穷尽的测试是不可能的，程序测试只能说明缺陷的存在，决不能说明没有缺陷。
3. 测试贯穿于全部软件生存周期，并且不是周期结束前的最后一个活动。
4. 理解测试背后的原因。
5. 首先测试测试用例。
6. 测试用例需要逐步完善、不断修订。
7. 缺陷成群集中出现，因此测试应该关注这些缺陷群。

8. 测试包括缺陷预防。

9. 测试是缺陷预防和缺陷检测之间的精心平衡。

10. 智能的和经过良好策划的自动化是实现测试效益的关键。

11. 测试需要具有天分、具有自信和信任团队的非常投入的人才。

我们将在随后的章节中阐述这些原理。我们在适当的时候将用一个信息技术领域以外的小故事阐明这些原理，以帮助读者理解。

1.3 不完善的车

汽车销售员：“这辆车是完好的，你只需要刷漆即可。”

不管开发什么软件，最终都要满足客户的需要。除此之外的一切都是次要的。测试是确保产品满足客户需要的一种手段。

我们要给“客户”下个广泛的定义。客户不仅指外部客户，也包括内部客户。例如，如果产品是用公司内部不同小组开发的组件组成的，那么这些不同组件的用户就应该被认为是客户，即使他们来自同一个公司。这种客户视角可以提高包括测试在内的所有活动的质量。

我们把内部客户的概念作进一步拓展：开发团队认为测试团队是其内部客户。这样可以保证产品的构建不仅针对使用需求，而且针对测试需求。这种观念可提高产品的“可测试性”，还可改善开发团队和测试团队之间的交互。

销售代表/工程师：“这部车有最好的变速器和制动器，它从静止加速到每小时80英里不到20秒！”

顾客：“噢，这有可能是真的，但是不幸的是当我踩刹车的时候车速却变得更快了！”

我们希望读者不要最后把这两个观点，即客户观点和质量观点，看作是附加的观点，而应该贯穿到本书从头到尾讨论的所有活动和组件中。

如果我们的工作是给客户提供一部完好的车（并且不要求用户刷漆），我们要确保车如用户所期望的那样工作，没有任何（大的）毛病，那么就要保证我们自己找到并改正车的所有缺陷。这就是测试的根本目标。我们在做任何测试工作时都必须记住这一点。

测试应该在用户没有发现缺陷前找到它们。

1.4 Dijkstra定律

假设一段接受六个字符密码的程序，并保证第一个字符必须是数字，其余字符是字符数字型的。如果我们的目标是穷尽测试，那么需要测试多少个输入数据的组合呢？

第一个字符可能有10种方式（数字0～9）。第二到第六个字符每一个字符都可能有62种方式（数字0～9，小写字母a～z，大写字母A～Z）。这就意味着总共有$10\times(62^5)$或9 161 328 320个有效组合要测试。假设每一个组合用10秒测试，测试这些所有有效组合将用2905年！

因此，在2905年后，我们可以得出结论：所有的有效输入都是可以接受的。但是这不是故事的结束，当我们输入无效数据，程序又会发生什么呢？继续上面的例子，如果假定有10个标点符号，那么我们将用44 176年测试所有有效和无效的输入组合。

所有这些仅是验证一个字段，并穷尽测试。显然，穷尽测试一个现实的程序是永远也做不到的。

上述这些都意味着只能选择测试用例的一个子集来测试。为了增加测试的有效性，应该选择最有可能发现错误的子集。本书第4章和第3章将讨论一些等价类划分、边界值分析、代码路径分析等技术，以帮助确定测试用例子集，从而增加发现缺陷的可能性。

然而，不管选择哪些测试用例子集，都不能100%地保证没有遗留缺陷。另一方面，引用一句老话，除了死亡和税收外没有任何事情是确定的，但是我们还是活着，并通过明智地把握不确定性来做其他事情。

> 测试只能证明缺陷的存在，但绝不能证明缺陷不存在。

1.5 及时测试

产品中的缺陷可能来自任何阶段。当获得最初需求时就可能存在错误。如果错误的或不完整的需求构成设计和开发产品的基础，那么最终产品的功能就不可能正确。类似地，当产品设计即构成产品开发（即编码）的基础是有缺陷的，那么缺陷设计产生的代码将不符合需求。因此，一个必要的条件是软件开发（需求、设计、编码等）的任何阶段都能发现并修正本阶段的缺陷，不能让缺陷传递到下个阶段。

我们看缺陷传递的潜在成本。如果在需求获取阶段所获取的需求是有错误的，而这个错误直到产品交付给用户时都没有发现，那么公司会遭受以下额外的损失：

- 基于错误的需求做出错误的设计；
- 在编码阶段将错误的设计传递到错误的代码中；
- 测试保证产品符合（错误）的需求；
- 发布的产品带着错误的功能。

在图1-2中，需求中的缺陷用深灰色方框表示。从图1-2可以看出，这些深灰色方框一直传递到随后的三个阶段，即设计、编码和测试中。

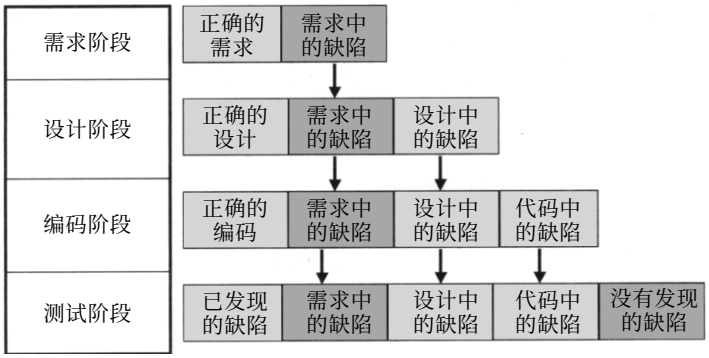


图1-2 早期阶段产生的缺陷如何使成本增加

当测试阶段完成后有瑕疵的产品送达用户手中，用户可能遭受停产带来的商业损失。反映到软件生产组织将是信誉的丧失。除了信誉损失，软件开发组织为了纠正问题，将不得不

重做列表中所有的步骤。

类似地，如果在设计阶段存在缺陷（尽管需求获取是正确的，彩图图1-2中用黄色表示），那么以后各阶段（编码、测试等）的成本会成倍增加。但是，这个成本比第一种情况在需求获取阶段出问题的成本要低。这是因为设计错误只能传递到编码和测试阶段。类似地，编码的缺陷只能传递到测试阶段。同样，可以预期编码阶段的缺陷影响较少（相对需求缺陷和设计缺陷，彩图图1-2中用绿色表示），导致的费用比前几个阶段要少。从上述讨论可以推断出，缺陷导致的费用随着检测出缺陷时间的拖延而增加。

构造产品的成本及产品中的缺陷数量随着缺陷数量传递到下一阶段而急剧增加。

因此，在缺陷注入（缺陷被引入）和检测缺陷（遇到缺陷并改正缺陷）之间相隔的时间越短，不必要的费用就越少。因此，尽可能早地发现缺陷是非常重要的。行业数据已经再次肯定了这些发现。虽然关于缺陷检测延迟造成的成本并没有公认的结论，但是需求阶段引入的缺陷如果最终出现在发布的产品中，其导致的费用将是在需求阶段内发现并改正缺陷的上千倍，请参考图1-3。

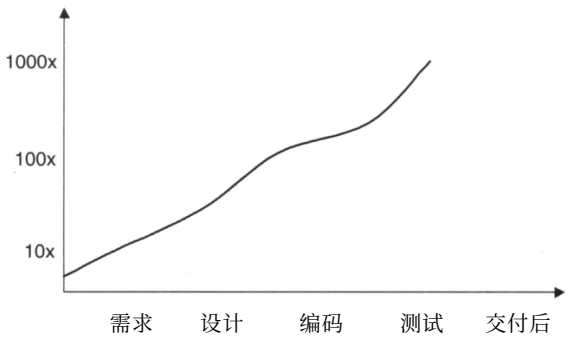


图1-3 缺陷对软件成本的综合作用

1.6 圣人和猫

一个圣人坐着思考。一只猫走来走去打断了他的沉思。于是，他叫门徒在他沉思时用篮子把猫扣住。此后，每次圣人思考都要把猫扣住。这变成了一种日常规程，一代代地传了很多年。一天，寺院里没有猫了。门徒们惊惶失措到处寻找，说：“我们需要猫。有了猫，我们才能用篮子把它扣起来，这样圣人才能思考！”

测试首先要明白测什么，正确的结果是什么和为什么要执行这些测试。如果不明白为什么就执行测试，最后就会执行不合适的测试，不能检查产品所应该做的。事实上，有时为确保测试成功而修改产品，甚至产品被改得不能满足用户的需要！

理解为什么要测试某些特定功能就引出不同的测试类型，将在第二部分介绍。我们做白盒测试，检查代码中的各种路径，以确保代码的正确运行。知道给定测试用例会执行哪些代码，就能对其进行必要的修改，以确保覆盖合适的路径。知道产品的外部功能，就可以设计黑盒测试。集成测试用于确保不同组件能够组合在一起。国际化测试确保产品能够在世界各地多种语言环境中正常运行。回归测试用于确保变更与设计要求的一样，没有任何意外的副作用。

“为什么测试”的重要性同“测试什么”和“怎样测试”一样。

1.7 首先测试测试用例

一个耳科医生要检查他的病人，告诉她：“我想测一下你能听多远。我从不同的距离问你的名字，你来回答。请转过身准备回答。”病人明白了需要做什么。

医生：（从30英尺外）：你叫什么名字？

……

医生：（从20英尺外）：你叫什么名字？

……

医生：（从10英尺外）：你叫什么名字？

病人：第三次了，我再说一边，我的名字叫Sheela！

从上述例子看出，很明显医生而不是病人的听力有问题。不知道医生会不会认为病人不能在20 英尺和30 英尺听见问话而为病人开了治疗处方。

测试用例就像程序和文档一样，也是人类生产的产品。我们也不能确保测试用例是完美的。重要的是要在测试之前确保测试用例本身没有问题。确保测试用例本身已通过检测的一种方法是，对于一个特定的测试用例，把输入和预期的输出文档化，请专家确认这些文档的有效性，也可以通过一些外部方法反向检查测试用例。例如，给定一个已知的输入值，并且单独跟踪程序或进程的执行路径，就可以手工确定预期的输出。通过比较这“已知正确的结果”和产品产生的结果，可以增加测试用例和被测产品的可信水平。第3章将讨论评审和审查实践，以及测试策划，第15章将讨论测试测试用例的方法。

首先测试测试用例——有缺陷的测试用例比有缺陷的产品更危险！

1.8 杀虫剂悖论

每年，各种各样的害虫袭击田野和农作物。农业专家们要找到正确的对抗方法，用新改良的配方设计出杀虫剂。有趣的是，害虫适应了新的杀虫剂，产生了免疫力，使新杀虫剂失效。随后的几年里，老的杀虫剂只能用来杀死没有免疫力的害虫，同时还必须引入一些新的改良配方，同更顽强的新变异害虫作斗争。新旧杀虫剂的结合有时阻碍了旧杀虫剂效能的发挥。随着时间的流逝，旧的杀虫剂变得毫无用处。于是，害虫和杀虫剂之间不停地战斗，看最终谁占上风。有时杀虫剂赢，但是，有时害虫又可以成功地战胜最新的杀虫剂。这场斗争的结果是大自然和杀虫剂的不断发展进化。

缺陷就像害虫，测试就是设计正确的杀虫剂捕获并杀死害虫，测试用例就是杀虫剂。像害虫一样，缺陷发展免疫力抵抗测试用例。当我们写新的测试用例发现产品的缺陷时，其他“隐藏”在下面的缺陷就暴露出来。

有两种方式可以解释产品如何发展它的“免疫力”对付测试用例。一种解释是，最初的测试用例深入到代码一定程度，由于发现缺陷而停止进一步进行。一旦这些缺陷修复，测试继续进行，直到进入新的

测试就像杀虫剂——必须不停地改变其构成，以对付新的害虫（缺陷）。

以前没有处理的代码从而发现新的缺陷。这种“白盒”或代码方法可以解释为什么新的测试用例可以发现新的缺陷。

关于免疫力的第二个解释是，当用户（测试人员）开始使用（执行）产品时，最初的缺陷妨碍他们使用全部的外部功能。随着测试用例开始运行，发现缺陷，修复问题，用户得以继续探索新的没有使用过的功能，并发现新的缺陷。这种“黑盒”观点采用功能方法解释为什么“测试越多缺陷越多”的现象。

另一个解释这个问题的方法是，不是缺陷发展了免疫力，而是测试用例走得更深，更进一步诊断问题并且最终“杀死缺陷”。遗憾的是，由于软件很复杂，并且在多个组件之间交互，因此最终杀死的事情很少。测试后缺陷依旧存在，困扰着用户，造成数不清的破坏。

这种不断地修订要运行的测试用例，以识别新缺陷的需要，促使我们研究测试策划并且执行不同类型的测试，尤其是回归测试。回归测试承认新的修复（杀虫剂）可导致新的“副作用”同时引起一些旧缺陷出现。设计并执行回归测试的核心问题是设计正确的测试用例，同在先前测试中获得免疫力的缺陷作斗争。本书第8章将讨论回归测试。

1.9 护航舰队与破布

我们每个人都经历过交通堵塞。通常在交通堵塞时，我们可以看到护航舰队一样的效果。严重堵塞的道路向前延伸，车辆看起来像加入一支护航舰队，跟随着缓缓地向前航行（即驾驶），直到遇到下一个护航舰队。

测试只能发现存在于聚类中的一部分缺陷，修复一个缺陷可能给该聚类中引入另一个缺陷。

缺陷在程序中通常也会呈现出护航舰队现象，缺陷集中出现。Glenford Myers在关于软件测试的开创性著作[MYER-79]中提出，程序中某部分错误存在的可能性与此部分已经发现错误的数量成正比，参见图1-4。

听起来这可能不合理，但在逻辑上是可以证明的。通常一个缺陷的修复会引入一些不安定因素，因此有必要再修复。所有这些修复的副作用就是导致缺陷在产品的某些部分成群出现。

从测试策划的角度看，这就意味着如果我们发现在软件的某一部分存在缺陷，那么应该花更多而不是更少的时间测试这部分程序。这将增加测试的投入回报，因为测试的目的就是发现缺陷。这还意味着无论何时产品有任何变化，受到影响的部分就是容易出错的区域，这部分就需要测试。我们将在第8章讨论这方面的内容。

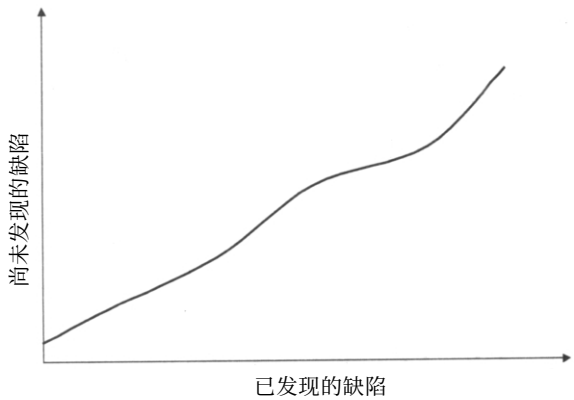


图1-4 尚未发现的缺陷随已发现缺陷数量的增加而增加

修复缺陷围绕特定几行代码进行。围绕同一段代码的修复会带来副作用。这会导致程序螺旋式变更，都是对特定部分的代码进行的。当我们审视这些修复了成群缺陷的代码，就像

看一块破抹布！修补衬衣的一个破洞可能损坏另一个地方。一劳永逸的解决方法就是扔掉衬衣买件新的。这就相当于重新进行体系结构设计和重写代码。

1.10 桥上的警察

城市中有座桥。只要人们从桥上走过就会掉下去。为了解决这个问题，市长任命了一个强壮的警察站在桥下面挽救掉下的人。问题在一定程度上得以缓解，但仍不能彻底解决。

这个警察退休后，一个新警察接替了他的工作。在开始的几天里，新警察没有站在桥底接住掉落的人，他同一个工程师一起修好了桥上的洞，这是早先那个警察没有注意到的。从那时起，没有人再坠桥，新警察也没有接住过一个人。（这使他现在的工作变得多余，他继续做其他对自己和人们都有益的事情……）

测试人员有可能能够很好地了解用户遇到的问题。就像上面故事所讲的第二个警察，他知道人们掉下来和人们为什么会掉下来。他们（宁愿冒着错过接住掉落人们的风险）不是简单地接住他们，而是调查导致掉落的根本原因并采取预防措施。测试人员本身可能难以采取预防措施。就像第二个警察需要工程师帮他堵上漏洞，测试人员也必须同开发工程师一起找到缺陷的根源并解决。测试人员不应认为解决问题与己无关，要像第二个警察一样，利用自己的缺陷检测经验，转换到缺陷预防，这既可以使自己的职业工作更丰富，也会使公司受益。

预防比治疗更有效——你可以把眼光放得更长远。

缺陷预防是测试人员工作的一部分。如果能够在缺陷预防和缺陷检测活动之间找到平衡，那么测试人员的职业工作就会变得丰富并会得到回报。这种职业发展道路包含在第13章将要介绍的一种三阶段模型中。下面我们讨论缺陷预防和缺陷检测之间的平衡问题。

1.11 钟摆的终结

任何软件公司最终的目的都是要确保用户能够得到没有多少缺陷的产品。达到这个目的有两个途径：一个是关注缺陷检测并改正缺陷，另一个是关注缺陷预防。这也称为质量控制关注和质量保证关注。

传统上测试被认为是一种质量控制活动，强调缺陷检测和更正。我们更愿意采用广义的测试观点，认为测试是面向测试预防的。例如，一方面第3章将讨论的白盒测试属于静态测试，包括桌面检查、代码走查、代码评审和审查。即使这些在传统上被认为是“质量保证”活动，整个测试活动的策划要以向用户交付高质量的产品为宗旨，这是难以做到的，除非全面地考虑通过质量保证能做什么，以及通过质量控制（即传统意义上的测试）能做什么。

质量保证通常都与过程模型例如CMM、CMMI、ISO 9001等关联，而质量控制通常与测试（测试是本书要讨论的主要内容）有关。这就导致了质量保证和质量控制之间的不自然分类。遗憾的是，公司都把这两种功能看作是互斥的，即“二选一”。我们甚至听到过“只要有好的过程管理，测试就是多余的”，或“过程是辅助的，测试可以发现一切”的说法。这就好像位于钟摆两端的两种思想流派：一个来源于缺陷预防（质量保证）关注，另一个来源于缺陷检测（质量控制）关注。经常会看到公司像钟摆一样从一个极端到另一个极端（如图1-5所示）。

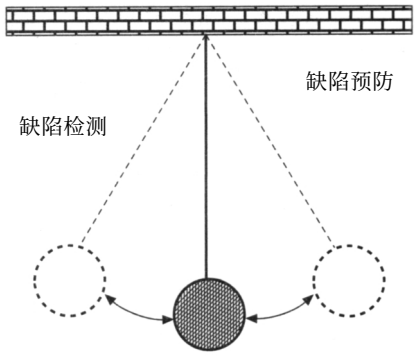


图1-5 质量控制和质量保证是提高质量的两种方法

我们认为，缺陷预防和缺陷检测不应看作是互斥的，也就是钟摆的两个极端，而应看作是两种相辅相成的活动，需要恰当地组合。图1-6给出了一种缺陷预防 - 缺陷检测网格，把这两种功能看作两个维度，两种活动正确的组合就对应于选取网格的的正确象限。

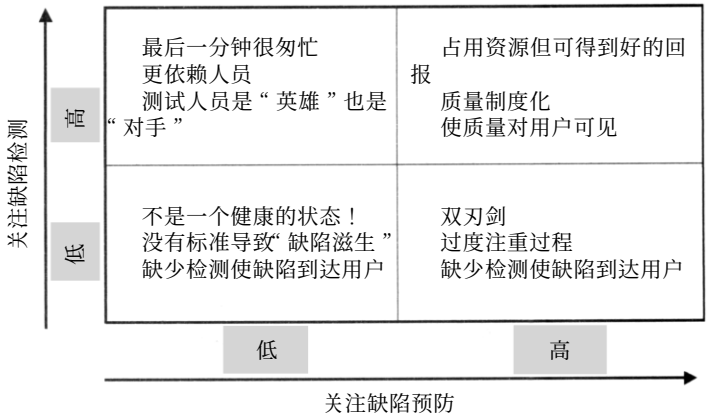


图1-6 缺陷检测关注和缺陷预防关注之间的关系

如果不太关注缺陷预防，就不会强调适合的标准、评审和过程。这种行为是缺陷理想的“温床”。保证产品质量的大部分工作都要靠测试及缺陷检测团队完成。如果也不太关注缺陷检测（图1-6的左下象限表示），公司就处于一种不好的状态。没有测试和缺陷检测活动就不能及时“杀死”缺陷，也就导致缺陷会达到用户。显而易见这不是一种健康的状态。

即使加强对缺陷检测的关注，但是仍然不注意缺陷预防（图1-6的左上象限），测试也会变成一种高度紧张、压力极大的工作。在产品发布前的最后一刻，还在检测大多数缺陷。测试人员于是成为及时找出所有缺陷从而扭转败局的超级英雄，同时他们也可能成为开发人员的敌人，因为他们总是找出开发人员所做工作的问题。这个象限比前一个好一些，但是很难维持，因为最后一刻玩的心跳会使人们筋疲力尽。

三个中国医生是兄弟。小弟是世界上非常有名的外科医生，他能发现人身上的肿瘤并切除。二哥能够在疾病的早期发现并开药治好它。他只在他所居住的城市非常有名。大哥不被外人所认识，但是两个弟弟总是听他的建议，因为他能够告诉他们在疾病没有出现时如何预防。大哥不是最出名的，但无疑却是最有效的。

预防疾病比治疗更有效。预防缺陷的人往往没有被关注。他们在公司中从未被称为英雄。灭火的人是大家看得见的人，不一定是一开始就确保火灾不发生的人。无论如何都不应该打击缺陷预防人们的积极性。

正如我们在上一节看到的，缺陷预防和缺陷检测不是互斥的，需要适当平衡才能提高产品质量。缺陷预防可改善产品生产过程的质量，而缺陷检测则可以捕获并改正过程中残留的缺陷。因此，缺陷预防就是关注过程，而缺陷检测就是关注产品。缺陷检测作为一种额外检查，可以放大缺陷预防的作用。

缺陷预防的提高在于能够建立评审机制，遵循先进的标准，完成工作遵循文档化的过程。从一开始就主动地关注把一切做好，会使测试工作（也就是缺陷检测）能够增加更多的价值，并能够在缺陷到达用户之前捕获残留的缺陷。质量就是始终如一地关注将缺陷预防与缺陷检测制度化。因此，一个公司必须分配足够的资源，以维持高水平的缺陷预防和缺陷检测活动（如图1-6所示的右上象限）。

缺陷预防和缺陷检测不能认为是互斥的，而应该是互为补充的。

但是，一个公司应该仔细避免过多依赖缺陷预防，而减少对缺陷检测的关注（如图1-6所示的右下象限）。过多关注缺陷预防，轻视缺陷检测，在所发布产品的质量管理层中会产生一种不舒服的感觉，因为内部发现的缺陷会很少。这种感觉会导致引入新过程，以改善缺陷检测的有效性。而太多的过程和缺陷预防最后可能被当作官僚形式，对于不同的情况没有灵活性和适应性。虽然过程带来了纪律约束，并降低了对个别人员的依赖，但是，如果没有实现其精神实质，过程也会成为双刃剑，会伤害到人们的积极性。如果公司既强调缺陷预防也强调缺陷检（图1-6中的右上象限），表面上看起来成本很高，但通过对内制度质量提升，对外使用户能够看到这种变化，这种投入必定带来丰厚的回报。

公司应当为缺陷检测和缺陷预防适当地定位，也就是选择图1-6中的适合象限。对缺陷预防和缺陷检测强调的尺度要随着产品类型、发布日期的临近程度和可用资源的不同而调整。综合考虑不同因素，有意识地平衡缺陷预防和缺陷检测，将使公司生产出更高质量的产品。对于一个公司来讲，要避免过于强调一方而忽视另一方面，这一点很重要，这在下一节详细讨论。

1.12 黑衣人

从以上讨论可以看出，测试要求人员有多项才能。从事测试工作的人员应该以客户为焦点，从客户角度理解隐含的要求。他们应具有很高的分析技能，以选择合适的测试用例子集，同时能正确应对杀虫剂悖论。他们应该提前考虑到缺陷预防，同时又能够识别并矫正出现的错误。最后，他们必须能够完成自动化工作（下一节将要阐述）。

为“测试”而自豪，就会处理好“其他一切”！

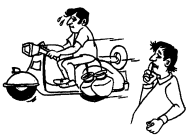
尽管所有这些都对所需的技术和人与人之间交流的技巧提出挑战，但测试仍然不是一种很受欢迎的工作。De Marco和Lister在他们的《人件》[DEMA-1987]一书中描述了一个有趣的实验。在测试团队中有目的地插入一些人，这些人“没有影响开发人员测试自己的程序会出现的认知问题”。这些人穿着特殊的衣服（黑衣服，以区别于公司中的传统工作服），并受到极大重视。所有的这些增加了他们的工作自豪感，使他们的成绩不可思议地飞跃增长。最初的团队成员离开并补充新成员很久以后，“黑衣团队”依然存在并且声名依旧。

选择测试作为职业的最大瓶颈在于缺乏自信。这种自信的缺乏和明显的对测试职业选择的不信任，使他们把测试看作是转到其他岗位的跳板（明显地，“开发”就是编写代码的委婉

提法)。结果是，测试人员没有能够在测试中寻找职业发展道路，而是增加了对测试这种职业的怀疑。

本书的第三部分将专门讨论职业期望和人们面临的类似问题。我们面对的一部分挑战是面对全球化，即合理利用全球资源保持竞争优势。第三部分的另一章将专门讨论由此引起的组织问题。

1.13 自动化综合症



一个农场主需要到一英里以外的井里提水用。于是，他雇了100个人从井里提水浇地。每人每天提一罐水，但这远远不够。庄稼枯死了。

在下一轮播种之前，农场主吸取了去年的教训，他想到自动化是提高生产率避免再次失败的关键方法。他听说摩托车可以更快地运水。于是，他买了50辆摩托车，解雇了50个工人，同时要求剩下的每人开摩托车运两罐水。看起来由于生产率的提高（由于摩托车的速度和便捷性），他需要的人更少了。遗憾的是，他是在庄稼生长期开始时，才选择使用摩托车的。因此，在开始的几周里，工人忙着学习使用摩托车。在学习摩托车的过程中，每天能运送的水罐数没有预想的那么多，加之工人的数量也减少了，运水能力实际上减少。庄稼又枯死了。

下一个播种季节又来了。现在除了一个人之外所有的工人都解雇了。农场主这次买了一辆卡车来运水。这次他也意识到需要训练工人学习驾驶。可是，从农场到井之间的路太窄，卡车不能帮他运来水。同样，这次庄稼也都枯死了。

经过这些经历后，农场主说：“没有自动化，我的日子还会好过点！”

如果仔细捉摸这个故事就会发现，庄稼枯死的多个原因都不是自动化引起的。农场主的失败不该直接归于自动化，而应该归于自动化所遵循的过程和不恰当的选择。第二年失败的原因是没有技能，第三年失败的原因是选择了不合适的工具。

自动化失败的例子比成功的多。自动化需要与产品开发一样的技能和关注。

第二年播种期到来时，农场主购买摩托车后立刻解雇工人，预期的金钱和时间都落空了。第三次他犯了同样的错误。自动化没有让他立刻得到回报。

上述故事的寓意同样适用于测试。自动化需要仔细地策划、评估和训练。自动化可能不会立刻产生回报。如果一个公司希望立刻通过自动化产生回报最终将会失望，并会错误地怪罪是自动化导致他们的失败，而不是客观地看待他们为自动化所作的策划、评估和训练等准备工作的水平。

大多数公司都因为自动化初期的失败而转向手工测试。遗憾的是，他们得出了错误的结论——自动化永远不能奏效。

测试的天性就包括重复性工作。也就是说，测试本身会自然地导致自动化。但是，自动化也是一把双刃剑。下面给出一些需要注意的关于自动化的要点：

- 在为了自动化而建议自动化之前，首先要知道为什么要采用自动化，以及想要自动化哪些内容。

- 在选择最适合需要的工具之前多做比较。
- 根据需要选择工具，而不是为配合工具的能力改变需要。
- 在指望测试人员提高生产率之前先进行培训。
- 不要期望自动化一夜之间就会产生回报。

1.14 小结

本章讨论了测试的一些基本原理，为后面内容的展开奠定了基础。本书由五部分组成。第一部分（包括本章）为本书其余部分确立了背景。下一章将讨论测试、验证和确认活动背景下的软件开发生存周期（SDLC）模型。

第二部分将讨论常见的测试类型。第3～10章涵盖了白盒测试、黑盒测试、集成测试、系统测试、确认测试、性能测试，回归测试、国际化测试和即兴测试的内容。

第三部分将讨论两个特殊且有些深奥的特殊测试问题，即第11章要讨论的面向对象测试、第12章要讨论的可使用性和易获得性测试。

第四部分将讨论一个经常被忽视的问题测试中的人员和组织问题。第13章将讨论常见的人员问题，包括一些误解、职业发展道路等问题。第14章将讨论当前流行的用于建立有效测试团队的不同组织结构，特别是在全球化背景下的组织结构。

最后的第五部分将讨论过程、管理和自动化等确保公司内部测试有效性的测试管理和自动化问题。第15章将阐述测试策划管理和执行，论述测试策划的内容、测试项目跟踪及相关问题。第16章将详述在测试领域正在出现且重要性日益凸显的重要问题，即实现测试自动化的好处、挑战和途径。第17章将详细说明衡量测试的有效性、产品的质量等，都需要采集什么数据，进行什么分析，以及如何使用这些信息实现可量化的持续改进。

虽然本书的各个部分都提供了必要的理论基础，但我们还是重点强调实践问题。通过阅读以上内容，读者可以概略了解本书其余部分的内容和背景关系。

问题与练习

1. 我们已经谈到了软件的普适性，它是迟早要对遗留在产品中的缺陷进行检测的一个原因。假设带有嵌入式软件的电视机可以在整个有线网络上自动下载、安装纠正错误的补丁，同时电视机制造商告诉你，这只需每周花5分钟的时间，而且“对用户免费。”你会同意吗？请给出一些不能接受的理由。
2. 你所在的公司已成功地开发出安装在几个客户那里的客户 - 服务器应用软件。你打算把它改成一个基于Web的应用，这样使任何人只需注册就能使用。请从质量和测试角度列出一些你认为这个经过修改的应用软件会面临的一些挑战。
3. 下面是一些来自产品开发公司的声音。找出这些观点中和本章相关原理不相符的错误。
   a. “本产品的代码是由CASE工具自动生成的，因此没有缺陷。”
   b. “我们已经通过了最新的过程模型认证，因此不需要测试。”
   c. “我们需要用针式打印机测试软件，我们从来没有发布过没有经针式打印机测试的软件。”
   d. “我已经运行了两个最新版运行过的所有测试用例，因此，不需要再运行任何其他测试用例了。”
   e. “我们的竞争对手用的就是这个自动工具，因此我们也应该用同样的工具。”

4. 假设需求获取时引入的每个缺陷到达客户的成本是10 000美元，设计缺陷和编码缺陷对应的成本分别为1000和100美元。还假设依据目前的统计数字，每个阶段要产生平均10个新的缺陷。此外，每个阶段的缺陷都会传递到下一阶段，那么现在的情况下，缺陷总成本是多少呢？如果采用了质量保证过程，每个阶段捕获50%的缺陷，使其不能传递到下一阶段，这可以节省多少成本呢？
5. 你要写一段两个两位整型数字相加的程序。你能穷尽测试这个程序吗？如果可以，需要多少个测试用例？假设每个测试用例的执行和分析需要一秒，运行所有的测试用例要花多长时间？
6. 我们认为程序中遗留的缺陷数量和检测出的缺陷数量成正比。为什么这个论点看似矛盾？同样，解释为什么在程序测试中会产生这种现象。