第一章　软件测试概述

工作目标

知识目标

- 了解软件测试的背景。
- 掌握软件缺陷的定义及缺陷跟踪流程。
- 熟悉软件测试的复杂性与经济性分析。
- 掌握软件测试的定义。
- 熟悉软件测试人员应具备的素质。

技能目标

- 掌握缺陷跟踪流程。

素养目标

- 培养学生的动手和自学能力。

工作任务

软件测试是软件开发过程的重要组成部分，是用来确认一个程序的品质或性能是否符合开发之前所提出的一些要求。软件测试的目的，第一是确认软件的质量，其一方面是确认软件做了你所期望的事情（Do the right thing），另一方面是确认软件以正确的方式来做了这个事件（Do it right）；第二是提供信息，比如提供给开发人员或程序经理的反馈信息、为风险评估所准备的信息；第三，软件测试不仅是测试软件产品的本身，而且还包括软件开发的过程。如果一个软件产品开发完成之后发现了很多问题，这说明此软件开发过程很可能是有缺陷的。因此软件测试的第三个目的是保证整个软件开发过程是高质量的条件。

软件质量是由以下几个方面来衡量的：

（1）在正确的时间用正确的的方法把一个工作做正确（Doing the right things right at the right time）。

（2）符合一些应用标准的要求，比如不同国家的用户不同的操作习惯和要求，项目工程中的可维护性、可测试性等要求。

（3）质量本身就是软件达到了最开始所设定的要求，而代码的优美或精巧的技巧并不代表软件的高质量（Quality is defined as conformance to requirements, not as "goodness" or "elegance"）。

（4）质量也代表着它符合客户的需要（Quality also means "meet customer needs"）。作为软件测试这个行业，最重要的一件事就是从客户的需求出发，从客户的角度去看产品，思考客户会怎么去使用这个产品，使用过程中会遇到什么样的问题。只有这些问题都解决了，软件产品的质量才可以说是上去了。

测试人员在软件开发过程中的任务：

（1）寻找 Bug；

（2）避免软件开发过程中的缺陷；

（3）衡量软件的品质；

（4）关注用户的需求。

总的目标是：确保软件的质量。

工作计划与实施

任务分析之问题清单

- 软件测试产生的背景。
- 软件测试。
- 软件缺陷的定义及跟踪管理流程。
- 软件测试的复杂性与经济性分析。
- 软件测试人员应具备的素质。

任务解析与实施

一、软件测试产生的背景

软件测试是伴随着软件的产生而产生的。早期的软件开发过程中，软件规模都很小、复杂程度低，软件开发的过程混乱无序、相当随意，测试的含义比较狭窄，开发人员将测试等同于“调试”，目的是纠正软件中已经知道的故障，常常由开发人员自己完成这部分的工作。对测试的投入极少，测试介入也晚，常常是等到形成代码、产品已经基本完成时才进行测试。

直到 1957 年，软件测试才开始与调试区别开来，作为一种发现软件缺陷的活动。由于一直存在着“为了让我们看到产品在工作，就得将测试工作往后推一点”的思想，潜意识里对测试的目的就理解为“使自己确信产品能工作”。测试活动始终落后于开发活动，测试通常被作为软件生命周期中的最后一项活动而进行。当时也缺乏有效的测试方法，主要依靠“错误推测（Error Guessing）”来寻找软件中的缺陷。因此，大量软件交付后，仍存在很多问题，软件产品的质量无法保证。

20 世纪 70 年代开发的软件仍然不复杂，但人们已开始思考软件开发流程的问题，尽管对“软件测试”的真正含义还缺乏共识，但这一词条已经频繁出现，一些软件测试的探索者们建议在软件生命周期的开始阶段就根据需求制订测试计划，这时也涌现出一批软件测试的宗师，Bill Hetzel 博士和 Glenford J. Myers 就是其中的领导者。

20 世纪 80 年代初期，软件和 IT 行业进入了大发展，软件趋向大型化、高复杂度，软件

的质量越来越重要。这个时候，一些软件测试的基础理论和实用技术开始形成，并且人们开始为软件开发设计各种流程和管理方法，软件开发的方式也逐渐由混乱无序的开发过程过渡到结构化的开发过程，以结构化分析与设计、结构化评审、结构化程序设计以及结构化测试为特征。人们还将“质量”的概念融入其中，软件测试定义发生了改变，测试不单纯是一个发现错误的过程，而且将测试作为软件质量保证（SQA）的主要职能，包含软件质量评价的内容。此时软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题。软件测试已有了行业标准（IEEE/ANSI），软件测试已成为一个专业，需要运用专门的方法和手段，需要专门人才和专家来承担。

在竞争激烈的今天，无论是软件的开发商还是软件的使用者，都生存在竞争环境中。软件开发商为了占有市场，必须把产品质量作为企业的重要目标之一，以免在竞争中被淘汰出局。用户为了保证自己的业务的顺利完成，当然希望选用有质量的软件。质量不佳的软产品不仅会使开发上的维护费用和用户的使用成本大副增加，还可能产生其他的责任风险，造成公司信誉下降。如果一些关键的应用领域质量有问题，还可能造成灾难性的后果。现在人们已经逐步认识到，软件中存在的错误导致了软件开发在成本、进度和质量上的失控。由于软件是由人来完成的，所以它不可能十全十美，虽然不可能完全杜绝软件中的错误，但是可以用软件测试等手段使程序中的错误数量尽可能少，密度尽可能小。

二、软件测试

1983 年 IEEE 提出的软件工程术语中给软件测试下的定义是：“使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别”。这个定义明确指出：软件测试的目的是为了检验软件系统是否满足需求。它再也不是一个一次性的，也不只是开发后期的活动，而是与整个开发流程融合成一体。

扩展定义：软件测试就是在软件投入运行前，对软件需求分析、设计规格说明和编码的最终复审，是软件质量保证的关键步骤。

软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计一批测试用例（包括输入数据与预期输出结果），并利用这些测试用例运行软件，以发现软件错误的过程。广义的软件测试由确认、验证、测试 3 个方面组成。

- 确认：评估将要开发的软件产品是否正确无误、可行和有价值。确认意味着确保一个待开发软件是正确无误的，是对软件开发构想的检测。主要体现在计划阶段和需求分析阶段，也会出现在测试阶段。
- 验证：检测软件开发的每个阶段、每个步骤的结果是否正确无误，是否与软件开发各阶段的要求或期望的结果相一致。验证意味着确保软件会正确无误地实现软件的需求，开发过程是沿着正确的方向进行的。主要体现在设计阶段和编码阶段。
- 测试：与狭隘的测试概念统一。主要体现在编码阶段和测试阶段。

确认、验证与测试是相辅相成的。确认产生验证和测试的标准，验证和测试帮助完成确认。

三、软件缺陷的定义及跟踪管理流程

缺陷跟踪管理是软件测试工作的一个重要部分，软件测试的目的是尽早发现软件系统中

的缺陷，因此对缺陷进行跟踪管理，确保每个被发现的缺陷都能够及时得到处理是测试工作的一项重要内容。错误一般有以下几类：

软件错误（Software Error）：指在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生；

- 软件缺陷（Software Defeat）：存在于软件之中的那些不希望或不可接受的偏差，如少一个逗点、多一个语句等；
- 软件故障（Software Fault）：软件运行过程中出现的一种不希望或不可接受的内部状态；
- 软件失效（Software Failure）：指软件运行时产生的一种不希望或不可接受的外部行为结果。

软件错误是一种人为错误。一个软件错误必定产生一个或多个软件缺陷，当一个软件缺陷被激活时，便产生一个软件故障；同一个软件缺陷在不同条件下被激活，可能产生不同的软件故障。软件故障如果没有及时的容错措施加以处理，便不可避免地导致软件失效，同一个软件按故障在不同条件下可能产生不同的软件失效。在软件开发过程中产生的缺陷，我们一般称之为 Bug。

1. 缺陷跟踪的目的

缺陷能够引起软件运行时产生的一种不希望或不可接受的外部行为结果，软件测试过程简单说就是围绕缺陷进行的，对缺陷的跟踪管理，一般需要达到以下目标：

- 确保每个被发现的缺陷都能够被解决；这里“解决”的意思不一定是被修正，也可能是其他处理方式（例如，在下一个版本中修正或是不修正）。总之，对每个被发现的 Bug 的处理方式必须能够在开发组织中达到一致。
- 收集缺陷数据并根据缺陷趋势曲线识别测试过程的阶段；决定测试过程是否结束有很多种方式，通过缺陷趋势曲线来确定测试过程是否结束是常用并且较为有效的一种方式。
- 收集缺陷数据并在其上进行数据分析，作为组织的过程财富。

上述的第一条是最受到重视的一点，在谈到缺陷跟踪管理时，一般人都会马上想到这一条，然而对第二和第三条目标却很容易忽视。其实，在一个运行良好的组织中，缺陷数据的收集和分析是很重要的，从缺陷数据中可以得到很多与软件质量相关的数据。

2. 缺陷的来源

按照一般的定义，只要软件出现的问题符合下列 5 种情况中的任何一种，就叫做软件缺陷。

- 软件未达到产品说明书标明的功能。
- 软件出现了产品说明书指明不会出现的错误。
- 软件功能超出产品说明书指明范围。
- 软件未达到产品说明书虽未指出但应达到的目标。
- 软件测试员认为软件难以理解、不易使用、运行缓慢，或者最终用户认为不好。

实践表明，大多数软件缺陷产生的原因并非源自编程错误，主要来自产品说明书的编写和产品方案设计。例如，产品说明书编写得不全面、不完整和不准确，而且经常更改，或者整个开发组没有很好地沟通和理解。

软件缺陷的第二大来源是设计方案，也就是软件设计说明书，这是程序员开展软件计划和构架的地方，就像建筑师为建筑物绘制蓝图一样，这里产生软件缺陷的原因与产品说明书或需求说明书是一样的，片面、多变、理解与沟通不足。

3．错误与缺陷的分布

开发早期的错误通常是很多的，而且还会转移到后期。没有被发现的错误，以及那些在开发过程中很晚才被发现的错误成本非常高，没有被发现的错误就在系统迁移，扩散，最终导致系统失效，直到很晚才发现的错误往往造成昂贵的返工代价。缺陷与错误的分布：需求占56%，设计占 27%，代码占 7%，其他占 10%。

四、软件测试的复杂性与经济性分析

人们在对软件工程开发的常规认识中，认为开发程序是一个复杂而困难的过程，需要花费大量的人力、物力和时间，而测试一个程序则比较容易，不需要花费太多的精力。这其实是人们对软件工程开发过程理解上的一个误区。在实际的软件开发过程中，作为现代软件开发工业一个非常重要的组成部分，软件测试正扮演着越来越重要的角色。随着软件规模的不断扩大，如何在有限的条件下对被开发软件进行有效的测试正成为软件工程中一个非常关键的课题。

设计测试用例是一项细致并且需要具备高度技巧的工作，稍有不慎就会顾此失彼，发生不应有的疏漏。下面分析了容易出现问题的根源。

（1）完全测试是不现实的。

在实际的软件测试工作中，不论采用什么方法，由于软件测试情况数量极其巨大，都不可能进行完全、彻底的测试。所谓彻底测试，就是让被测程序在一切可能的输入情况下全部执行一遍。通常也称这种测试为“穷举测试”。穷举测试会引起以下几种问题：输入量太大；输出结果太多；软件执行路径太多；说明书存在主观性。

E.W.Dijkstra 的一句名言对测试的不彻底性作了很好的注解：“程序测试只能证明错误的存在，但不能证明错误的不存在”。由于穷举测试工作量太大，实践上行不通，这就注定了一切实际测试都是不彻底的，也就不能够保证被测试程序在理论上不存在遗留的错误。

（2）软件测试是有风险的。

穷举测试的不可行性使得大多数软件在进行测试的时候只能采取非穷举测试，这又意味着一种冒险。比如在使用 Microsoft Office 工具中的 Word 时，可以作这样的一个测试：①新建一个 Word 文档；②在文档中输入汉字“胡”；③设置其字体属性为“隶书”，字号为“初号”，效果为“空心”；④将页面的显示比例设为“500%”。这时在“胡”字的内部会出现“胡万进印”四个字。类似问题在实际测试中如果不使用穷举测试是很难发现的，而如果在软件投入市场时才发现，则修复代价就会非常高。这就会产生一个矛盾：软件测试员不能做到完全的测试，不完全测试又不能证明软件的百分之百可靠。那么如何在这两者的矛盾中找到一个相对的平衡点呢？

由如图 1-1 所示的最优测试量示意图可以观察到，当软件缺陷降低到某一数值后，随着测试数量的不断上升，软件缺陷并没有明显地下降。这是软件测试工作中需要注意的重要问题。如何把测试数据量巨大的软件测试减少到可以控制的范围、如何针对风险做出最明智的选择是软件测试人员必须能够把握的关键问题。

图 1-1 的最优测试量示意图说明了发现软件缺陷数量和测试量之间的关系，随着测试量的

增加，测试成本将呈几何数级上升，而软件缺陷降低到某一数值之后将没有明显的变化，最优测量值就是这两条曲线的交点。

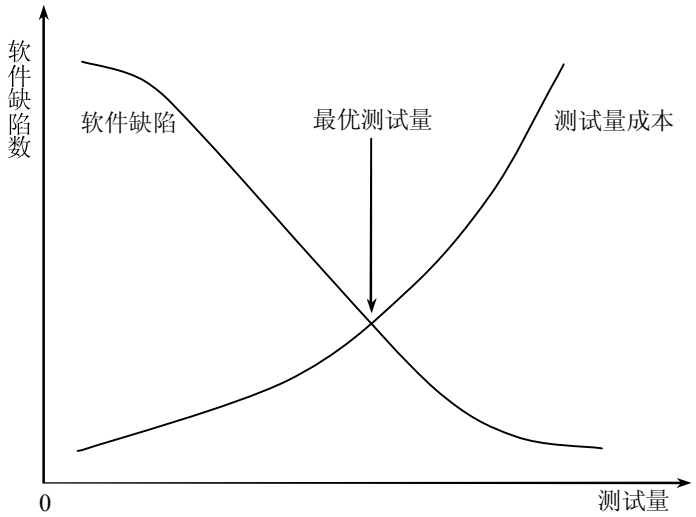

图 1-1　最优测试量示意图

（3）杀虫剂现象。

1990 年，Boris Beizer 在其编著的《Software Testing Techniques》（第二版）中提到了“杀虫剂怪事”一词，同一种测试工具或方法用于测试同一类软件越多，则被测试软件对测试的免疫力就越强。这与农药杀虫是一样的，总用一种农药，则害虫就有了免疫力，农药就失去了作用。

由于软件开发人员在开发过程中可能碰见各种各样的主客观因素，再加上不可预见的突发性事件，所以再优秀的软件测试员，采用一种测试方法或者工具也不可能检测出所有的缺陷。为了克服被测试软件的免疫力，软件测试员必须不断编写新的测试程序，对程序的各个部分进行不断的测试，以避免被测试软件对单一的测试程序具有免疫力而使软件缺陷不被发现。这就对软件测试人员的素质提出了很高的要求。

（4）缺陷的不确定性。

在软件测试中，还有一个让人不容易判断的现象是缺陷的不确定性，即并不是所有的软件缺陷都需要被修复。究竟什么才算是软件缺陷是一个很难把握的标准，在任何一本软件测试的书中都只能给出一个笼统的定义。实际测试中需要把这一定义根据具体的被测对象明确化。即使这样，具体的测试人员对软件系统的理解不同，还是会出现不同的标准。

软件测试的经济性有两方面体现：一是体现在测试工作在整个项目开发过程中的重要地位；二是体现在应该按照什么样的原则进行测试，以实现测试成本与测试效果的统一。

软件工程的总目标是：充分利用有限的人力和物力资源，高效率、高质量地完成测试。

五、软件测试人员应具备的素质

计算机领域的专业技能是测试工程师应该必备的一项素质，是做好测试工作的前提条件。尽管没有任何 IT 背景的人也可以从事测试工作，但是一名要想获得更大发展空间或者持久竞

争力的测试工程师，其计算机专业技能是必不可少的。一个有竞争力的测试人员要具有下面三个方面的专业技能：

（1）测试专业技能。

现在软件测试已经成为一个很有潜力的专业。要想成为一名优秀的测试工程师，首先应该具有扎实的专业基础，这也是本书的编写目的之一。因此，测试工程师应该努力学习测试专业知识，告别简单的“单击”之类的测试工作，让测试工作以自己的专业知识为依托。

测试专业知识很多，本书内容主要以测试人员应该掌握的基础专业技能为主。测试专业技能涉及的范围很广，既包括黑盒测试、白盒测试、测试用例设计等基础测试技术，也包括单元测试、功能测试、集成测试、系统测试、性能测试等测试方法，还包括基础的测试流程管理、缺陷管理、自动化测试技术等知识。

（2）软件编程技能。

“测试人员是否需要编程”可以说是测试人员最常提出的问题之一。实际上，由于在我国，开发人员待遇普遍高于测试人员，因此能写代码的几乎都去做开发了，而很多人则是因为做不了开发或者不能从事其他工作才“被迫”从事测试工作。最终的结果则是很多测试人员只能从事相对简单的功能测试，能力强一点的则可以借助测试工具进行简单的自动化测试（主要是录制、修改、回放测试脚本）。

软件编程技能实际应该是测试人员的必备技能之一，在微软，很多测试人员都拥有多年的开发经验。因此，测试人员要想得到较好的职业发展，必须能够编写程序。只有能编写程序，才可以胜任诸如单元测试、集成测试、性能测试等难度较大的测试工作。

此外，对软件测试人员的编程技能要求也有别于开发人员：测试人员编写的程序应着眼于运行正确，同时兼顾高效率，尤其体现在与性能测试相关的测试代码编写上。因此测试人员要具备一定的算法设计能力。依据作者的经验，测试工程师至少应该掌握 Java、C#、C++之类的一门语言以及相应的开发工具。

（3）网络、操作系统、数据库、中间件等知识。

与开发人员相比，测试人员掌握的知识具有“博而不精”的特点，“艺多不压身”是个非常形象的比喻。由于测试中经常需要配置、调试各种测试环境，而且在性能测试中还要对各种系统平台进行分析与调优，因此测试人员需要掌握更多网络、操作系统、数据库等知识。

在网络方面，测试人员应该掌握基本的网络协议及网络工作原理，尤其要掌握一些网络环境的配置，这些都是测试工作中经常遇到的知识。

操作系统和中间件方面，应该掌握基本的使用、安装、配置等。例如很多应用系统都是基于 UNIX、Linux 来运行的，这就要求测试人员掌握基本的操作命令及相关的工具软件。而 WebLogic、WebSphere 等中间件的安装、配置很多时候也需要掌握一些。

数据库知识则更是应该掌握的技能，现在的应用系统几乎离不开数据库。因此不但要掌握基本的安装、配置，还要掌握 SQL。测试人员至少应该掌握 My SQL、MS SQL Server、Oracle 等常见数据库的使用。

作为一名测试人员，尽管不能精通所有的知识，但要想做好测试工作，应该尽可能地去学习更多与测试工作相关的知识

根据有关职位统计资料显示，在国外大多数软件公司，1 个软件开发工程师就需要辅有 2 个软件测试工程师。目前，软件测试自动化技术在我国则刚刚被少数业内专家所认知，而这方

面的专业技术人员在国内更是凤毛麟角。根据对近期网络招聘 IT 人才情况的了解，许多正在招聘软件测试工程师的企业很少能够在招聘会上顺利招到合适的人才。

随着中国 IT 行业的发展，产品的质量控制与质量管理正逐渐成为企业生存与发展的核心。从软件、硬件到系统集成，几乎每个中大型 IT 企业的产品在发布前都需要大量的质量控制、测试和文档工作，而这些工作必须依靠拥有娴熟技术的专业软件人才来完成。而软件测试工程师就是其中之一。

据了解，由于软件测试工程师处于重要岗位，所以必须具有电子、电机类相关专业知识背景，并且还应有两年以上的实际操作经验。他们应熟悉中国和国际软件测试标准，熟练掌握和操作国际流行的系列软件测试工具，能够承担比较复杂的软件分析、测试、品质管理等任务，并能独立担任测试、品质管理部门的负责人。一般情况下，软件测试工程师可分为测试工程师、高级测试工程师和资深测试工程师三个等级。

在具体工作过程中，测试工程师的工作是利用测试工具，按照测试方案和流程对产品进行功能和性能测试，甚至需要编写不同的测试工具，设计和维护测试系统，对测试方案可能出现的问题进行分析和评估。对软件测试工程师而言，必须具有高度的工作责任心和自信心。任何严格的测试必须是一种实事求是的测试，因为它关系到一个产品的质量问题，而测试工程师则是产品出货前的把关人，所以，没有专业的技术水准是无法胜任这项工作的。同时，由于测试工作一般由多个测试工程师共同完成，并且测试部门一般要与其他部门的人员进行较多的沟通，所以要求测试工程师不但要有较强的技术能力，而且要有较强的沟通能力。

因此，在企业内部，软件测试工程师基本处于“双高”地位，即地位高、待遇高，有的人月薪可高达 8000 元。可以说他们的职业前景非常广阔，从近期的企业人才需求和薪金水平来看，软件测试工程师的年薪有逐年上升的明显迹象。测试工程师这个职位必将成为 IT 就业的新亮点。

缺陷跟踪管理流程如图 1-2 所示，测试人员发现一个新 Bug，则将其状态置为 New，由项目经理或者测试经理审核该 Bug 是否为真正的缺陷，如果是，则将 Bug 状态置为 Open，并将该 Bug 进行评级，分配给相应的开发人员，开发人员在接到该 Bug 后对其进行修复（如果项目经理没有进行审核就直接分配，此时开发人员可以拒绝修复该 Bug），修复完成后则提交给测试人员做相应的验证，如果验证通过，则关闭该 Bug；如果没有通过，则重新打开该 Bug。

在 Bug 的跟踪管理过程中有很多的关键字，下面就对这些关键字进行一一说明：

Bug 的流转状态关键字

- 未确定的（Unconfirmed）。这个 Bug 最近才被发现，还没有人确认它是否真的存在，如果有其他测试人员碰到了同样的问题，就可以将这个 Bug 标志为 New，或者将这个 Bug 删除，或者做上 Closed 标记。
- 新加入的（New）。这个 Bug 最近被测试人员添加到 Bug 列表中，是已经被证实存在且必须修改的。即将被分配，如果分配了，可以标志为 Assigned，未分配则将保留 New 标志，或者做上 Resolved 标记。
- 确认分配的（Assigned）。测试人员将 Bug 的修复任务分配给具体的实现人员，如果 Bug 不属于被分配实现人员的范围，可置为 Reassigned，等待被重新指定相关修改人员。

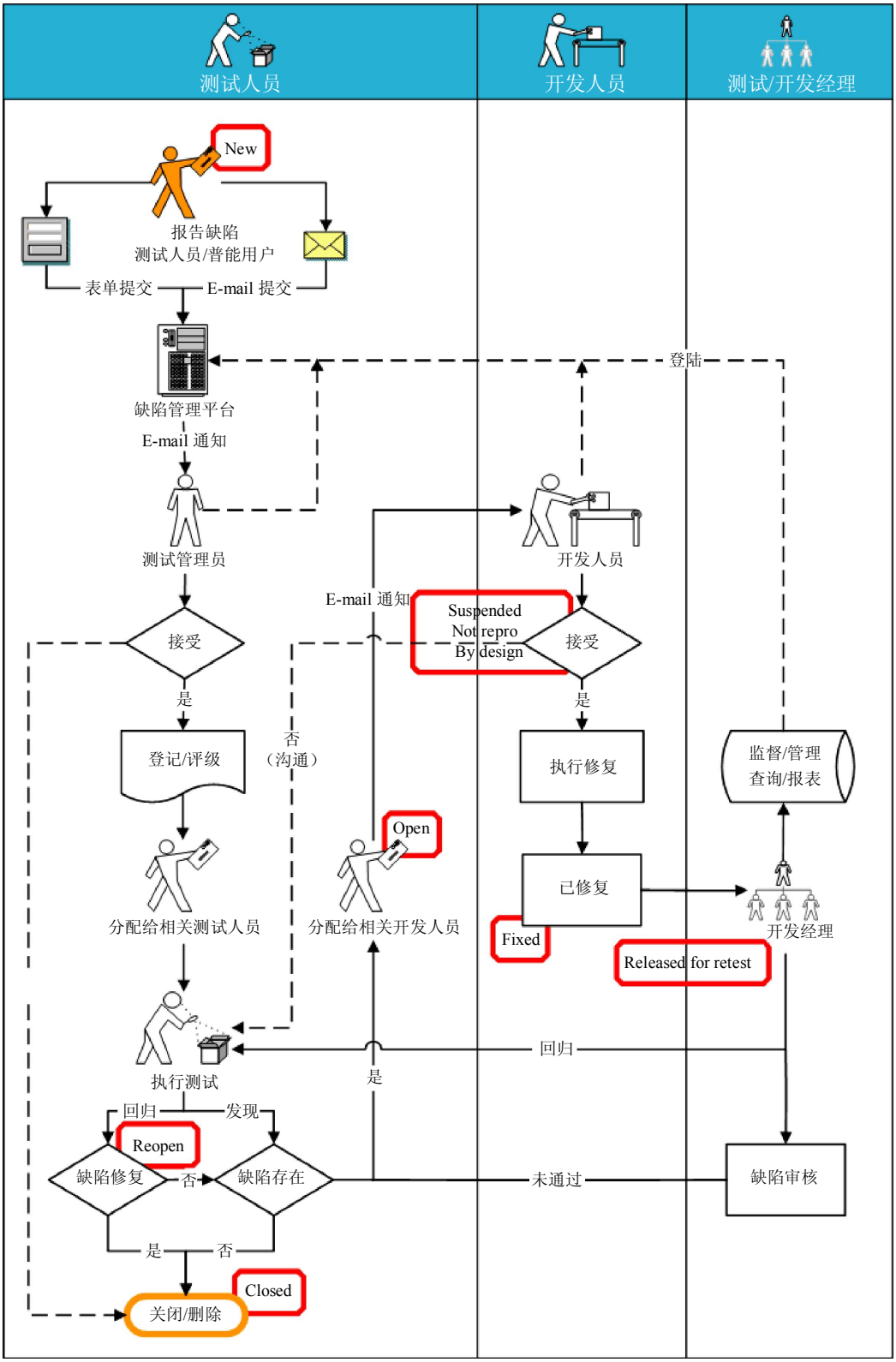
测试人员
开发人员
测试/开发经理
New
报告缺陷
测试人员/普能用户
表单提交
E-mail 提交
缺陷管理平台
E-mail 通知
测试管理员
登陆
开发人员
E-mail 通知
Suspended
Not repro
By design
接受
是
否
（沟通）
登记/评级
执行修复
监督/管理
查询/报表
Open
分配给相关测试人员
分配给相关开发人员
已修复
开发经理
Fixed
Released for retest
回归
执行测试
是
回归
发现
Reopen
缺陷修复
否
缺陷存在
未通过
缺陷审核
是
否
Closed
关闭/删除

图 1-2　缺陷跟踪流程图

- 重新分配的（Reassigned）。该 Bug 不属于被分配实现人员的范围，可置为 Reassigned 等待被重新指定相关修改人员。
- 需要帮助的（Needinfo）。测试人员或实现人员无法对发现的 Bug 进行精确定位或描述，需要相关实现人员协助，以更深刻地认识和修复这个 Bug。
- 重复出现的（Reopened）。该 Bug 已经不是第一次被发现，它可以被标志为 Assigned 或者 Resolved。
- 已解决的（Resolved）。实现人员对被分配给自己的 Bug 进行修改，修改完以后，修改状态。
- 重新启用的（Reopen）。当实现人员发现某些 Bug 具有关联性，即使该 Bug 被正确修复了，也会被发送到起始状态，等待回归再次确认。或测试人员发现该 Bug 没有被真正修改后重置状态。
- 正在验证的（Verified）。测试人员对标记为 Resolved 状态的 Bug 进行验证。
- 安全关闭的（Closed）。该 Bug 已经被完全解决。

Bug 的解决关键字：

- 已经修复（Fixed）。该 Bug 被正确修复了，并且得到了测试人员的确认。
- 无法修复（Wontfix）。发现的 Bug 永远不会被修复，或者该 Bug 牵涉面太广，需要委员会决定。
- 下版本解决（Later）。发现的 Bug 不会在产品的这个版本中解决，将在下一个版本中被修复。
- 无法确定（Remind）。发现的 Bug 可能不会在产品的这个版本中解决。
- 重复的（Duplicate）。发现的 Bug 是一个已存在 Bug 的复件。
- 无法证实（Incomplete）。用了所有的方法都不能再现这个 Bug，没有更多的线索来证实此 Bug 的存在，即使看程序源代码也无法确认这个 Bug 的出现。
- 测试错误（NotaBug）。Bug 报告出现了错误，将正确的软件过程报告成 Bug 了。
- 无效的（Invalid）。描述的问题不是 Bug，属于测试人员输入错误，通过此项来取消。
- 问题归档（Worksforme）。所有要重现这个 Bug 的企图都是无效的，如果该 Bug 有更多的信息出现，则重新分配这个 Bug，而现在只把它列入问题归档。

Bug 的严重等级：

- 危急的（Critical）。能使不相关的系统内软件（或整个系统）损坏，或造成严重的信息遗失，或为安装该软件包的系统引入安全漏洞。
- 重大的（Grave）。使该软件包无法或几乎不可用，或造成数据遗失，或引入一个允许侵入此软件包用户账号的安全漏洞。
- 严重的（Serious）。该软件包违反了“必须”或“必要”的规定，或者是软件包维护人员和测试人员认为该软件包已不适合发布。
- 锁定的（Blocker）。这个 Bug 阻碍了后面的操作，需要马上或者尽快排除
- 重要的（Important）。该错误影响了软件包可用性，但不致于造成所有人都不可用。
- 常规的（Normal）。为默认，适用于大部分的错误。
- 轻微的（Minor）。该错误不致于影响软件包的使用，而且应该很容易解决。

- 微不足道的（Trivial）。该错误无关紧要，多指外观 GUI 上的字符拼写错误，不影响整个项目。

在实际工作中一般分为“致命”、“严重”、“一般”、“建议”四种缺陷的紧急程度就可以了，如果用 1～4 表示其严重级别，则 1 是优先级最高的等级，4 是优先级最低的等级。

理论上 Bug 处理的优先级有以下 5 级：

Bug 处理的优先等级

- 立刻修复（Immediate）。这个 Bug 已经阻碍了开发工作或者测试工作，需要立刻修复。
- 马上修复（Urgent）。这个 Bug 阻碍了软件的一部分应用，如果不修复将妨碍下面计划的实施。
- 尽快修复（High）。真实存在的并不是很严重，在版本发布之前修复。
- 正常修复（Normal）。有充足的时间来修复这个问题，并且这个 Bug 给现行的系统的影响不大。
- 考虑修复（Low）。不是什么关键 Bug，在时间允许的时候可以考虑修复。

一个完成的缺陷应该包括以下几个方面的内容，如表 1-1 所示。

表 1-1　缺陷内容列表

可追踪信息	缺陷 ID	唯一的缺陷 ID，可以根据该 ID 追踪缺陷
缺陷基本信息	缺陷状态	缺陷的状态，分为“待分配”、“待修正”、“待验证”、“待评审”、“关闭”
	缺陷标题	描述缺陷的标题
	缺陷的严重程度	描述缺陷的严重程度，一般分为“致命”、“严重”、“一般”、“建议”四种
	缺陷的紧急程度	描述缺陷的紧急程度有 1～4 级，1 是优先级最高的等级，4 是优先级最低的等级
	缺陷提交人	缺陷提交人的名字（邮件地址）
	缺陷提交时间	缺陷提交的时间
	缺陷所属项目/模块	缺陷所属的项目和模块，最好能较精确地定位至模块
	缺陷指定解决人	缺陷指定的解决人，在缺陷“提交”状态为空，缺陷“分发”状态下，由项目经理指定相关开发人员修改
	缺陷指定解决时间	项目经理指定的开发人员修改此缺陷的 deadline
	缺陷处理人	最终处理缺陷的处理人
	缺陷处理结果描述	对处理结果的描述，如果对代码进行了修改，要求在此处体现出来
缺陷基本信息	缺陷处理时间	缺陷处理的时间
	缺陷验证人	对被处理缺陷验证的验证人
	缺陷验证结果描述	对验证结果的描述（通过、不通过）
	缺陷验证时间	对缺陷验证的时间
缺陷的详细描述		对缺陷的详细描述；之所以把这项单独列出来，是因为对缺陷描述的详细程度直接影响开发人员对缺陷的修改，描述应该尽可能详细
测试环境说明		对测试环境的描述
必要的附件		对于某些文字很难表达清楚的缺陷，使用图片等附件是必要的

巩固与提高

一、选择题

1．实施缺陷跟踪的目的是（　　）。

A．软件质量无法控制　　B．问题无法量化

C．重复问题接连产生　　D．解决问题的知识无法保留

E．确保缺陷得到解决　　F．使问题形成完整的闭环处理

2．TTP 支持以下（　　）平台。

A．Windows　　B．Solaris　　C．Linux　　D．Mac OS X

3．Fixed 的意思是指（　　）。

A．该 Bug 被正确修复了，并且得到了测试人员的确认

B．该 Bug 被拒绝了，得到了测试人员的确认

C．该 Bug 没有被修复，但得到了测试人员的确认

D．该 Bug 被关闭了，但得到了测试人员的确认

二、填空题

1．_____表示已经修复，该 Bug 被正确修复了，并且得到了测试人员的确认。

2．描述缺陷的严重程度，一般分为_____、_____、_____、_____四种。

3．_____指在软件生存期内不希望或不可接受的人为错误，其结果是导致软件缺陷的产生。

三、操作题

在您以往的工作中，一条软件缺陷（Bug）记录都包含了哪些内容？如何提交高质量的软件缺陷记录？