第3章 生命周期软件测试方法

如同软件生命周期一样，我们也可以将软件测试阶段按照软件生命周期去划分，形成了基于生命周期的软件测试（后面简称生命周期测试）。此时，软件测试可以划分为测试需求分析、测试计划、测试设计、测试开发、测试执行、测试评估。这样，生命周期测试方法将测试延伸到需求分析、设计审查活动中去，也就是将“质量保证”的部分活动归为测试活动，真正体现了“尽早地和不断地进行软件测试”的原则，确保了对软件生命周期的每个阶段进行质量管理，并通过测试手段实现对各个阶段的质量保证。

3.1 生命周期测试的概念

按照传统的软件生命周期的观点，测试是在编程活动之后进行的，是软件开发的最后一个阶段。随着人们对软件工程化的重视以及软件规模的日益扩大，软件分析、设计的作用越来越突出，而且有资料表明，60％以上的软件错误并不是程序错误，而是需求分析和系统设计错误。如，从 IBM 提供的数据来看，对一个大约 60 个缺陷/千行的软件，2/3 的缺陷产生在需求和设计阶段，而在需求和设计阶段发现缺陷并进行修正的花费最小，否则，到了系统测试阶段来修正所发现的缺陷，花费是以上的 10 倍，到了产品发布阶段来修正所发现的缺陷，花费将是 100 倍。这说明，在需求和设计阶段就能发现软件的缺陷，那么修正所需的花费比在编程完成后再进行测试所需的花费少很多。因此，做好软件需求和设计阶段的测试工作就显得非常重要，这就使得传统的测试概念扩大化，从而提出了软件生命周期测试的概念。

生命周期测试伴随着整个软件开发周期，此时测试的对象不仅仅是程序，需求、功能和设计同样要测试。如在项目需求分析阶段就要开始参与，审查需求分析文档、产品规格说明书；在设计阶段，要审查系统设计文档、程序设计流程图、数据流图等；在代码编写阶段，需要审查代码，看是否遵守代码的变量定义规则、是否有足够的注释行等。测试与开发同步进行，有利于尽早地发现问题，同时缩短项目的开发建设周期。

3.1.1 生命周期测试的工作划分

生命周期测试意味着测试与软件开发平行，在软件开发的所有阶段进行测试，确保在尽可能早的阶段点去修正缺陷，用来减少测试成本。与软件开发一样，生命周期测试需要正式的测试流程来支持。即在软件开发团队组建时，测试小组也同时建立，在一个项目开始时，测试计划和测试条件也随着开始，并在生命周期的各阶段结束点测试系统，以确保能正确地开发系统，和尽可能在生命周期的最早的可能点发现软件的缺陷。（图 3-1）

图 3-1 表示当项目开始时，系统开发过程和系统测试过程同时开始，开发小组开始系统开发过程，而系统测试小组开始计划系统测试过程。两个小组在同一点开始，使用相同的信息。

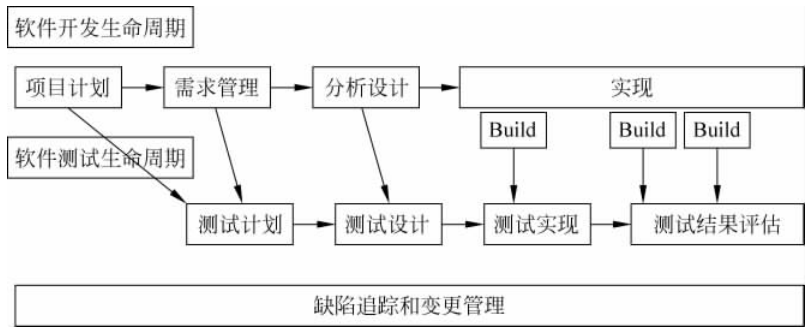

图 3-1　生命周期测试概念

测试小组在开发过程的若干个预定点对系统进行连续的测试，检查开发过程的结果。软件生命周期中要进行的测试如表 3-1 所示。

表 3-1　生命周期各阶段测试工作划分

生命周期阶段	验 证 活 动
需求	决定验证的方法 决定需求的充分程度 生成功能测试 决定与需求符合的设计
设计	决定设计的充分程度 生成结构和功能测试数据 决定设计与需求的一致性
编程	决定实现的充分程度 生成各种程序/单元的结构和功能测试数据 决定与设计的一致性
测试	决定测试计划的充分性 测试应用系统
安装/集成	把经测试的系统放入产品
维护	修改和重新测试

在需求阶段，重点是确认定义的需求符合机构的要求；在设计和编程阶段，重点是验证设计和程序实现了需求；而在测试和安装阶段，重点是检查实现的系统符合系统规格说明；在维护阶段，系统将重新测试以决定改变的部分和未改变的部分能继续工作。

3.1.2　生命周期测试的主要任务

基于生命周期测试方法对一个应用系统进行测试的测试工作过程是一个三维过程。一维概述测试要素，二维定义每个阶段要测试的事务，三维是一个测试计划，如图 3-2 所示。测试策略描述测试工程的总体方法和目标，描述目前在进行哪一阶段的测试(单元测试、集成测试、系统测试)以及每个阶段内在进行的测试种类(功能测试、性能测试、压力测试等)，给出为什么要执行测试和达到测试目标的最有效的途径。这通常由非常熟悉该软件的商业风险的小组开发；而测试种类/技术详细地解释测试的类型或采用的技术，说明执行什么测试和如何进行测试，由测试小组确定测试方法和技术的选择。

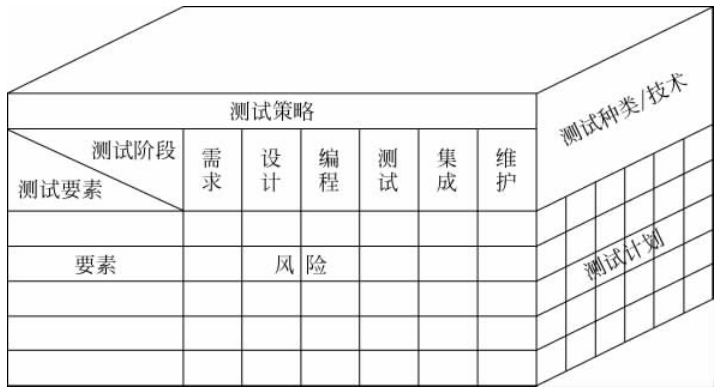

图 3-2 生命周期测试工作三维图

1．测试要素

测试要素是计算机软件的属性，描述测试的主要目标，一个测试要素有若干个测试事件，一个事件描述测试条件和可能发生的事件，在生命周期各个阶段中，对每一个测试要素所进行的测试是不同的，也就是有不同的测试事件，它们的关系如表 3-2 所示。其中，

(1) 正确性：数据输入、过程处理和输出的正确性(IPO)。

(2) 文件完整性：文件被正确使用，恢复和存储的数据正确。

(3) 授权：特殊的授权可以执行一个特殊的操作。

(4) 进程追踪：当进程运行中，程序有能力证实进程在正常工作。

(5) 系统运行的连续性：当有非致命性问题发生后，系统有能力继续运行关键的任务。

(6) 服务水平：系统有紧急情况发生时，要求程序的输出结果不经过处理或进行简单的处理后就可以直接使用。

(7) 权限控制：防止系统被误用(意外或者有意的)。

(8) 一致性：确保最终设计和用户需求完全一致。

(9) 可靠性：在规定的时间内都可以正常运转。

(10) 易于使用：多数人均感觉易于使用。

(11) 可维护性：可以很容易地定位问题，并且进行修改。

(12) 可移植性：数据或者程序易于移植其他系统上。

(13) 耦合性：系统中的组件可以很容易地联接。

(14) 性能：系统资源的占用率、响应时间、并发处理。

(15) 操作性：易于操作(Operator)。

我们在确定测试策略时，首先选择并确定测试要素的等级(多数情况下选择 3～7 个)，并确定开发阶段；然后明确商业风险，此时开发人员、重要用户和测试人员通过评审的方式对这些风险达成一致的意见；最后把风险列表存放在需求矩阵中，矩阵中可以将风险同测试用例对应起来。

2．风险

风险是导致失败的条件，计算机系统的风险是始终存在的。有些风险并不一定导致系统失败。我们不能消除风险，但可以减少风险发生的概率。在软件生命周期各阶段，要标识和评估计算机系统的风险，风险的概念决定测试的类型和测试工作量。决定哪些风险是可以接受的，把这些风险变成测试的领域，然后制定测试计划达到这个目标。

表 3-2　测试要素和测试事件

序　号	测试要素	需　　求	设　　计	编　　程	测　　试	安　　装	维　　护
1	可靠性	建立精度等级	设计数据完整性控制	实现数据完整性控制	人工、回归和功能测试	验证安装的精度和完整性	修改精度要求
2	授权	定义授权规则	设计授权规则	实现授权规则	符合性测试	禁止改变数据	保存授权规则
3	文件完整性	定义文件完整性需求	设计文件完整性控制	实现文件完整性控制	功能测试	检查产品文件的完整性	保存文件完整性
4	审计追踪	定义重构处理需求	设计审计追踪	实现审计追踪	功能测试	记录安装审计追踪	修改审计追踪
5	处理连续性	定义失效的影响	设计中断计划	编写中断计划和过程	恢复性测试	保证以前测试的完整性	修改中断计划
6	服务级别	定义希望的服务级别	设计达到服务级别的方法	达到服务级别的服务系统	强度测试	实现故障安装计划	保存服务级别
7	存取控制	定义系统的存取	设计存取过程	实现安全过程	符合性测试	集成期间的存取控制	保存安全级别
8	方法论	按系统开发方法论定义需求	按系统设计方法论设计系统	按编程方法论编写程序	按测试方法论执行测试	在产品环境中集成系统	按系统维护方法论维护系统
9	正确性	定义功能规格说明	设计符合需求	程序符合设计	功能测试	程序和数据安装正确	修改需求
10	容易使用	定义可用性规格说明	系统设计要便于实现可用性需求	程序编写符合易用性设计要求,并进行了优化	人工支持测试	传播可用性指令	保存容易使用
11	可维护	决定可维护的规格说明	设计是可维护的	程序是可维护的	检查	文档齐全	保存可维性
12	可移植	决定可移植的要求	设计是可移植的	程序符合设计	程序符合设计灾难性测试	文档齐全	保存可移植性
13	耦合	定义系统间的接口	设计考虑接口需求	程序符合接口设计说明	功能和回归测试	调整接口	保证正确的接口
14	性能	建立性能准则	保证实际要达到这些准则	程序的实际和实现达到这些准则	符合性测试	监控集成性能	保存性能级别
15	容易操作	定义操作要求	把要求传递给操作	开发操作过程	操作测试	实现操作过程	修改操作过程

计算机系统的风险表现为：产生不正确的结果、系统接受未授权的事务、破坏计算机文件的完整性、不能重新构造处理、破坏处理的连续性、向用户提供的服务将降低到不可接受的程度、将危及系统的安全、结果不可靠、系统难于使用、程序难于维护、不能移植到其他计算机软硬件环境、不可接受的性能级别以及系统难以操作等。

3．测试计划

1）常见问题

在制定测试计划时我们可能经常遇到下面的问题：

（1）测试计划经常是等到开发后期才开始实行，使得没有时间有效地执行计划。

（2）测试计划的组织者可能缺乏对特殊应用软件测试经验（如嵌入式软件）。

（3）测试的难度和复杂性可能太大，没有自动化工具，很难计划和控制。

2）确定测试策略的因素

在确定测试策略时要考虑以下几个方面：

（1）要使用的测试技术和工具。

（2）测试完成标准。

（3）影响资源分配的因素（如外部接口出现故障、物理设备损坏以及安全受到威胁等）。

测试计划最关键的一步就是将软件分解成单元，写成测试需求。测试需求有很多分类方法，最普通的一种就是按照商业功能分类。把软件分解成单元有几个好处：

① 测试需求是测试设计和开发测试用例的基础，分成单元可以更好地进行设计；

② 详细的测试需求是用来衡量测试覆盖率的重要指标；

③ 测试需求包括各种测试实际所要做的工作，以及所需资源。

3）测试的工作量

测试计划中估计测试工作量一般要从如下几个方面进行综合考虑。

（1）效率假设：测试队伍的工作效率。对于功能测试，这主要依赖于应用的复杂性，如窗口的个数，每个窗口中的动作数目。对容量测试，主要依赖于建立测试所需数据的工作量大小。

（2）测试假设：为了验证一个测试需求所需测试动作数目。

（3）应用的维数：应用的复杂度指标。例如要加入一个记录，测试需求的维数就是这个记录中域的数目。

（4）所处测试周期的阶段：有些阶段主要工作是设计，有些阶段主要是测试执行。

（5）确定测试资源，如硬件和软件环境以及测试工具的系统资源。当然最重要的是人力资源，包括测试项目负责人、测试分析员、测试设计员、测试程序员、测试员、测试系统管理者以及配置管理员等。这些工作人员的职责见表 3-3。

表 3-3　软件测试人员配备情况

工作角色	具体职责
测试项目负责人	管理监督测试项目，提供技术指导，获取适当的资源，制定基线，技术协调，负责项目的安全保密和质量管理
测试分析员	确定测试计划、测试内容、测试方法、测试数据生成方法、测试（软、硬件）环境、测试工具，评价测试工作的有效性
测试设计员	设计测试用例、确定测试用例的优先级，建立测试环境
测试程序员	编写测试辅助软件
测试员	执行测试、记录测试结果

续表

工作角色	具体职责
测试系统管理员	对测试环境和资产进行管理和维护
配置管理员	设置、管理和维护测试配置管理数据库
注 1：当软件的供方实施测试时，配置管理员由软件开发项目的配置管理员承担；当独立的测试组织实施测试时，应配备测试活动的配置管理员。 注 2：一个人可承担多个角色的工作，一个角色可由多个人承担。	

测试计划按国家标准或行业标准规定的格式和内容编写。

4．测试种类/技术

软件生命周期中所执行的各类测试如图 3-3 所示。

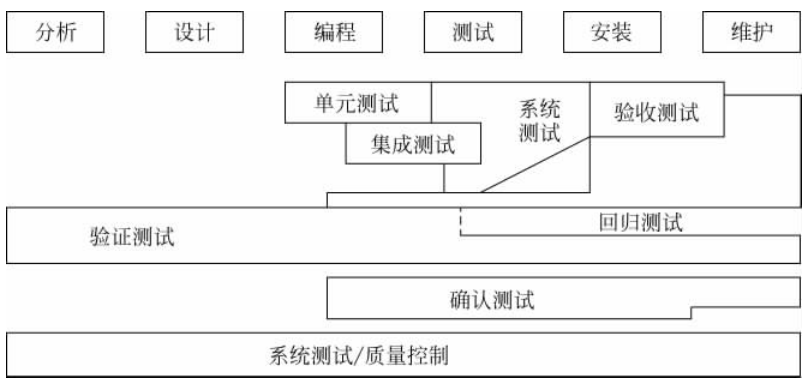

图 3-3　生命周期中的软件测试

下面是软件生命周期中的各类软件测试的定义和概念。

(1) 质量控制(Quality Control)：

① 决定软件产品正确性的过程和动作。

② 一组功能基线，保证产品符合标准/需求所做的工作。

(2) 缺陷(Defect)：

① 偏离规格说明，有三种表现形式(遗漏、错误和多余)。

② 用户不满意的任何事情，不管是否在规格说明书中规定。

(3) 验证(Verification)：在整个软件生命周期中的全部质量控制活动，确保交付的中间产品符合输入规格说明。

(4) 确认(Validation)：软件生命周期中的测试阶段，保证最终产品符合规格说明。

(5) 静态测试：在系统编码之前进行的验证。

(6) 动态测试：在系统编码之后进行的验证和确认。

(7) 单元测试：对单一、独立的模块或编码单元进行的测试。

(8) 集成测试：对一组模块进行的测试，确保模块之间的数据和控制能正常地传递。

(9) 系统测试：

① 一个预先确定的测试组合，当执行成功时，系统符合需求(即确认系统开发正确)；

② 与单元测试不同的各种更高等级测试类型的通用术语。

(10) 验收测试：保证系统符合最终用户要求的测试。

(11) 回归测试：在系统改变后进行的测试，以确保不希望的变化被引入到系统。

(12) 功能测试：认为系统应该做什么的业务需求测试。

(13) 结构测试：确认系统是如何实现的系统结构测试。

(14) “黑盒”测试：数据驱动的、基于外部规格说明而不需了解系统是如何构造的测试。

(15) “白盒”测试：逻辑驱动的、基于编码内部的结构和逻辑的测试。

5. 测试的准入准出条件

1) 准入条件

进入软件测试生命周期一般应具有下列条件，即我们说的测试准入条件：

(1) 具有测试合同(或项目计划)。

(2) 具有软件测试所需的各种文档。

(3) 所提交的被测软件受控。

(4) 软件源代码正确通过编译或汇编。

(5) 能够从一开始介入被测软件的开发周期。

2) 准出条件

软件测试工作结束一般应达到下列要求，即我们说的测试准出条件：

(1) 已按要求完成了合同(或项目计划)所规定的软件测试任务。

(2) 实际测试过程遵循了原定的软件测试计划和软件测试说明。

(3) 客观、详细地记录了软件测试过程和软件测试中发现的所有问题。

(4) 软件测试文档齐全、符合规范。

(5) 软件测试的全过程自始至终在控制下进行。

(6) 软件测试中的问题或异常有合理解释或正确有效的处理。

(7) 软件测试工作通过了测试评审。

(8) 全部测试软件、被测软件、测试支持软件和评审结果已纳入配置管理。

3.1.3 基于风险的软件测试方法

风险可以定义为事件、危险、威胁或情况等发生的可能性以及由此产生的不可预料的后果，即一个潜在的问题。

风险级别由出现不确定事件的可能性和出现后所产生的影响(事件引发的不好的结果，即严重性)两个方面来决定。

在软件测试中，由于测试团队需要在时间、成本和质量等各个方面进行平衡和协调，使得我们无法做到穷尽测试，加上测试时间以及测试资源的限制(还要考虑测试人员的水平因素)，我们很难达到理想的测试目标：使被测软件“零”缺陷。这样，软件就可能存在缺陷和质量问题，由此带来了软件存在着应用上的风险。如故障频发的软件交付使用、软件/硬件对个人或公司造成潜在损害、劣质的软件特性(如功能性、可靠性、可用性和性能)、低劣的数据(如数据迁移问题、数据转换问题、数据传输问题、违反数据标准问题)、软件没有实现既定的功能等。风险的存在也会导致用户或者利益相关者对软件质量或项目成功的信心不足。

1. 什么是基于风险的软件测试

基于风险的软件测试(Risk-Based software Testing，RBT)是指首先评估被测软件的风险，然后根据不同的风险采用不同的测试力度。通常的方法是：

① 列出一个风险的列表。

② 对每个风险进行分析和评估，确定风险级别。

③ 考察每项风险的测试。

④ 当风险消失而新的风险出现的时候，调整测试策略。

现在业界通常对风险进行评估的做法是对每个功能点从业务和技术上考察。业务上是指这项功能失效，对系统的影响。从技术上考察是指实现这个功能的技术难度大不大，是移植的还是新研发的？一般将此两项称为重要性和概率，分别赋予 1 到 5 的权值，5 为最大可能或最重要。

对于重要性为 5、概率为 4 的一个功能点，其乘积为 20，这是一个高风险。对于高风险，就应该用充足的时间、充足的人员来进行测试。

在基于风险的软件测试中，需要解决的主要问题包括确定测试的优先级、选择测试的重点、配置测试的资源、分析和评估测试的有效性等。要有效地选择测试重点和测试优先级，风险测试将测试活动和测试任务根据风险划分优先等级，将测试资源分配在高风险的部分。

这种基于风险的软件测试方法目前得到了广泛的关注，很多机构在他们针对基础级别和进阶级别的测试认证中，将它认定为一种重要的测试方式。

2. 基于风险的软件测试所能解决的问题

基于风险的测试作为软件测试的一种有效方法，可以解决测试过程中面临的一些问题。

1）测试团队面临的问题是测试任务的时间压力

很少有测试项目可以获得足够的时间进行充分的测试。相反，测试一般都是有时间限制的，例如：项目具体里程碑时间、客户或者用户要求产品提交的时间等。基于风险的测试可以提供一种方法，对测试用例和测试任务进行优先级排列。测试的时间限制，其面临的挑战实际就是确定测试的覆盖率。通过基于风险的测试，可以从几乎无限的测试中选择重要的和风险高的测试来开展，从而降低风险和尽快提高质量，提高对产品的信心。测试的时间压力不仅仅存在于测试实现和执行阶段，同样也存在于测试分析和设计阶段。通过基于风险的测试，可以在早期将测试工作量放在高风险的地方，同时可以告知利益相关者这样做可能存在的风险。

2）测试团队经常面临的问题是系统需求质量低下或不完整

通过召集利益相关者讨论哪些是需要测试的、哪些是不需要测试的、测试的深度是多少等问题。基于风险的测试可用来识别需求规格说明中存在的不足。基于风险的测试也可以帮助其他利益相关者认识到测试在确定测试范围和测试深度方面面临的挑战，有助于项目团队成员之间更好地理解和沟通。

3）在项目测试的后期。例如，完成测试执行之后，测试团队需要提供相关的信息给其他的利益相关者，以帮助做出合适的决定

基于风险的测试可以允许测试团队和其他利益相关者一起，根据剩余的风险确定一个可接受的风险级别。而不是仅仅依赖于其他一些不充分的度量，例如缺陷数目、测试用例执行数目等。

3. 基于风险的软件测试的活动实践

有效地应用基于风险的软件测试可以较好地指导软件测试活动的开展，更好地使软件测试活动在时间、成本、质量等方面进行平衡，从而提高测试质量、降低测试成本、缩短测试时间等。下面是基于风险的软件测试方法。

1）确定测试优先级

根据测试风险的分析和评估得到风险分布，确定测试的优先级（风险级别分析也适用于测试的设计和测试实现等阶段，即通过风险分析，确定测试设计和测试实现的优先级）。测试风险的分析基于两个方面：发生的可能性和发生的严重程度。其中，风险发生的可能性主要是从技术方面考虑；而风险发生的严重程度主要是从客户或者用户的角度考虑。

2）确定测试完备性

前面提到的一个假设条件：并不是所有的测试对项目而言是同等重要的。同样的道理，并不需要对测试对象的不同内容进行同等重要的测试。例如，最重要或者风险最大的模块/对象需要进行更加彻底、更加全面的测试。而对于风险比较小、优先级低的模块/对象，可以进行简单测试。对于优先级最低的对象，在时间和成本等不允许的时候，甚至不进行测试。

3）确定测试资源分配

根据测试风险的分析和测试优先级的评估，将经验丰富和技术能力丰富的测试人员（不管是设计人员、实现人员，还是执行人员）放在最重要的模块或测试对象中，以达到最佳效果：①设计更加完善、完备和准确的测试用例；②实现高质量的测试用例脚本和代码；③更加高效地发现测试对象中的缺陷。

4）监控测试进度

根据测试风险的分析和评估，得到测试的优先级和测试重点。接下来，可以根据风险的分布对测试进度进行汇报和控制。例如：测试经理可以根据测试工作的侧重点、测试进度协调人力资源和测试环境的分配，将测试的资源放在最重要的部分。

5）加速测试信心提升

依据测试风险分析和评估得到的测试优先级和测试重点，可以更好、更快地提供产品或者被测系统在质量方面的信心。对被测对象的质量，根据不同的测试策略，得到不同的信心演变过程。

策略 1：随机执行测试用例，不分优先级和测试重点，被测系统质量信心的递增是随着测试完成率的递增而线性增加的。

策略 2：先执行低复杂度的测试，因此，测试完成率增加很快，但是相应的被测对象质量的信心却增加很慢。而对于高风险（如测试难度较大的大容量用户数据模拟测试）的区域，很可能放在测试的后期进行。

策略 3：基于风险的测试，将高风险区域首先进行测试，尽管测试完成率增加比较慢，但是对被测对象质量的信心却增加很快。

3.2 生命周期各个阶段的测试要求

全生命周期中软件测试的最终要求是：①保证软件系统在全生命周期中每个阶段的正确性，验证在整个软件开发周期中各个阶段的软件质量是否合格；②保证最终系统符合用户的要求和需求，验证最终交付给用户的系统是否满足用户需要、符合其需求；③用样本测试数据检查系统的行为特性；④测试的最终目的是确保最终交给用户的产品的功能符合用户的需求，原则是把尽可能多的问题在产品交给用户之前发现并改正。为此，要努力保证生命周期中每个阶段的正确性，使其满足阶段出口的要求。

3.2.1 需求阶段测试

据软件工程统计结果发现 50％以上的系统错误是由于错误需求或缺少需求导致的，在需求上发生错误将导致相互纠缠和重复劳动，因而测试费用的 80％是花在需求错误的追踪上。

需求测试贯穿了整个软件开发周期，通过需求测试可以知道软件测试的各个阶段，帮助我们设计测试过程、安排测试计划、编写测试用例以及确认测试结果等。有一个正确的需求分析，则大部分缺陷将不会进入到设计和编码阶段，测试所需的费用自然会大大减少，因此，需求

阶段测试的所有花费都是值得的。

1. 需求阶段测试的目标

简单来说，需求阶段测试的目标就是保证需求分析的正确性和充分性。具体地说，需求阶段测试的目标则是保证需求正确反映出用户的需要，需求已经被定义和文档化，项目的花费和收益成正比，需求的控制被明确，有合理的流程可以遵循，有合理的方法可供选择。

2. 需求阶段的测试要素分析

需求阶段的测试要素分析以表 3-2 为基础，包含的内容有：

① 需求的设计是否遵循了已定义的方法。

② 提交了已定义的功能说明。

③ 定义了系统界面。

④ 已经估计了性能标准。

⑤ 容忍度被预先估计。

⑥ 预先定义了权限规则。

⑦ 需求中预先定义了文件完整性。

⑧ 预先定义了需求的变更流程。

⑨ 预先定义了失败的影响。

3. 需求阶段的测试活动

在需求阶段测试中，需要建立风险列表，进行风险分析和检查，以此确定项目的风险；并且要建立控制目标，确保有足够的控制力度来保证软件项目的开发和测试。在彻底地分析需求的充分性后，生成基础的测试用例。澄清和确定哪些需求是可测试的，舍去含糊的、不可测试的需求，建立产品的测试需求和确认需求。

3.2.2　设计阶段测试

在设计阶段，设计人员需要根据需求分析详细定义要交付的产品——硬件和软件的需求、操作手册说明书、数据保留的策略、输入/输出说明、过程说明、控制说明、系统流程图等。而测试的任务是对设计进行评审，分析测试要素，给测试要素打分，当需求分析发生改变，设计文档也要修改，测试要对修改的部分进行检查，以保证设计和需求的一致性。

1. 设计阶段的测试活动

设计阶段包括概要设计和详细设计。在概要设计阶段，测试人员应阐述测试方法和测试评估准则，编写测试计划，组织成立一个独立的测试小组，安排具有里程碑的测试日程；在详细设计阶段，测试人员要开发或获取确认支持工具，生成功能测试数据和测试用例，以此来检查设计中的遗漏情况、错误逻辑、模块接口不匹配、数据结构不合理、错误 I/O 假定、用户界面不充分等。

2. 设计阶段的评审

设计阶段的评审是对设计的完整性和正确性进行正式的评价。在对设计进行评审之前，要为评审分配足够的时间，成立评审组，并对组员进行培训；在评审时，要通报项目组，和项目组一起进行评审，并且只对文档进行评审；最后，要将评审的结果写成正式报告。

3. 设计阶段工具的应用

在设计阶段使用静态和动态测试工具测试系统的结构。评分工具和设计评审工具是广泛使用的两种测试工具，评分是标识风险的一种工具，根据得分的结果确定系统的风险程度；而设计评审是对实际阶段处理的完整性进行正式的评价，它是测试设计规格说明的工具，风险越

高，设计评审越详细。

另外，可利用评分工具对测试要素进行分析，给测试要素打分，如：是否设计了对数据完整性的控制？是否设计了权限规则？是否设计了对文件完整性的控制？是否设计了审计追踪？是否设计了发生意外情况时的计划？是否设计了如何达到服务水平的方法？是否定义了权限流程？是否定义了完整的方法学？是否设计了保证需求一致性的方法？是否进行了可用性的设计？设计是否是可维护的、简单的？交互界面设计是否完毕？是否定义了成功的标准？

上述评分过程需要同实际操作者沟通。

3.2.3 编码阶段测试

在编码阶段，测试需要解决的首要问题是编码是否和设计一致；其次是系统是否可维护，系统的规格说明是否正确地实现，编码是否按照既有的标准进行，是否有充分的评价测试计划的可执行程序，程序是否提供了足够的文档资料，程序内部是否有足够的注释等。在测试完成后，要形成下列输出：编码说明书、程序文档、计算机程序列表、可执行的程序、程序流程图、操作介绍和单元测试结果等。

在编码阶段已经开发了很多的测试工具，例如支持程序走查和检查的代码静态分析工具和支持单元“黑盒”测试和单元“白盒”测试的动态测试工具。在编码阶段的测试活动中，有几个方面是需要特别关注的：

① 完成对数据和文件完整性的控制。

② 定义完毕授权的规则。

③ 实现审计追踪。

④ 规划出意外情况发生后的处理计划。

⑤ 编码工作是依据规定的方法完成的(这样易于测试和维护工作的进行)。

⑥ 编码与设计相一致(包括编码的正确性、可用性、间接性和耦合性)。

⑦ 代码是可维护的(代码的维护性在一定程度上决定了项目维护的难易程度)。

⑧ 在性能上定义出程序成功的标准。

3.2.4 测试阶段

测试阶段就是传统软件工程中的软件测试。在全生命周期软件测试方法中，由于在需求、设计、编码阶段都进行了测试，因此测试阶段的问题相对传统的软件测试中的问题要少一些。在测试阶段要进行第三方的正式确认测试，检验所开发的系统是否能按照用户提出的要求运行。在测试阶段要使得用户能成功地安装一个新的应用系统来进行测试。

1. 典型的测试类型

典型的测试类型如下。

1) 手册与文档测试

测试软件的操作说明文档是否全面、正确、简单和满足标准，即测试软件文档的可用性。

2) 一致性测试

测试软件的授权、安全性和性能等是否达到需求分析中的要求。

3) 符合性测试

验证软件系统与相应标准的符合程度。如授权规则是否正确实现，安全方法是否合适，是否按照相应的标准、指南、规程执行测试等。

4）功能测试

运行部分或全部系统，确认用户的需求被满足。这包括可靠性、文件完整性、审计追踪、功能正确性、互连等项测试，检验系统在各种环境和重复的事务条件下能否正确地执行系统的需求，控制计算机文件的完整性，追踪一个原始事务到总的控制，按用户规定的需求测试应用功能，与其他应用系统能否正确地通信。

5）覆盖性测试

检验软件代码各个语句及分支等是否全部执行到。

6）性能测试

通过测量响应时间、CPU 使用和其他量化的操作特征，评估软件系统的性能指标。

7）压力测试

以大信息量的数据进行输入，测试软件的性能。但是这是一个很昂贵的测试，应根据需要来选择。但是在线系统必须要进行压力测试。

8）强度测试

将系统置于强度下进行验收测试，测试系统对极端条件的反应，标识软件的薄弱点，指出系统能够经受的正常的工作量。

9）操作测试

在没有开发人员的指导和帮助情况下，由操作人员进行测试，以评估操作命令的完整性和系统是否容易操作。

10）恢复测试

故意使系统失败，测试人工和自动的恢复过程。

2. 测试用例

设计测试用例的输入数据时，要包含合法和非法的输入（要尝试将测试数据违反程序的规则进行输入），也要描述运行测试用例的预期结果。测试用例需要包含的一些基本信息有标识符（测试设计过程说明和测试程序说明引用的唯一标识符）、输入说明（列举软件执行测试用例的所有输入内容或条件）、输出要求（描述运行测试用例预期的结果）、环境要求（执行测试用例必要的硬件、软件等）、特殊过程要求（描述运行测试必须达到的特殊要求）以及用例之间的依赖性（当一个测试用例依赖于其他测试用例，或者受其他用例影响时，需在此说明）。

3. 测试报告

在测试阶段开始之前，测试人员要参考前期的测试结果和第三方测试反馈（例如计算机操作人员）准备测试阶段的测试计划和测试用例，在完成测试时要生成正式的测试总结报告。生成测试报告的目的是要表示出目前项目的实际情况，给出系统的操作性的评价，为软件的产品化提供一个参考。

测试总结报告的内容有测试结果数据、测试任务、测试集合和测试事件的描述、当前软件状态的描述、各个阶段的项目测试总结等。

3.2.5　安装阶段测试

在进行安装测试时要保证被测试系统没有问题，校验产品文件的完整性。安装要遵循一定的方法和步骤，注重对程序安装的正确性和完整性进行核对。如果安装失败，系统要有相应的解决方案。最后也是最重要的是要保证系统综合的性能达到了用户要求。

1. 安装测试的基本要求

在安装过程中，要根据系统安装手册制定好安装计划，确定好安装流程图，准备好安装文

件和程序清单，给出安装测试的预期结果，并对安装过程中的各种可能发生的结果进行说明和准备，将程序运行的软硬件要求放入产品说明中。同时要检查系统的用户手册和操作手册，看是否可用。

2. 安装测试工作

安装过程中我们要进行的工作有：

(1) 对程序安装的正确性和完整性进行核对。

(2) 校验产品文件的完整性。

(3) 安装的审查、追踪被记录。

(4) 安装之前，该系统已经被证实没有问题。

(5) 如果安装失败，系统有相应的解决方案。

(6) 安装过程，进行了权限控制(安全性)。

(7) 安装遵循一定的方法和步骤。

(8) 需要的配套程序和数据已经放进了产品中。

(9) 已交付使用说明。

(10) 相关文件已经完成(可维护性)。

(11) 接口已经被合理调整(耦合性)。

(12) 综合的性能达到了用户要求。

3.2.6 验收阶段测试

软件验收的流程如下。

1. 定义用户角色

定义用户角色也就是确定软件的用户范围。

2. 定义验收标准

验收标准包括功能、性能、接口质量、过载后的软件质量、软件的安全性和稳定性等方面的标准。

3. 编制验收计划

验收计划包含项目描述、用户职责描述、验收活动描述、验收项的评审和最终的验收测试步骤。

4. 执行验收计划

按照验收计划进行测试和评审。

5. 填写验收结论

验收结束后，填写得出的结论，验收问题必须在进入下一个活动之前被接受和更改。

3.2.7 维护阶段

软件交付使用后的阶段，称为维护阶段。

维护阶段指软件维护阶段的工作重点是测试和培训。

1. 维护阶段中的测试要求

由于软件产品的特殊性，测试过后，并不代表软件没有错误，只能说有些错误还没有被发现，因此在软件交付使用后，仍旧需要对其进行维护。维护人员需要开发一些测试用例，预先发现一些问题；并且要能够根据运行情况的变化和用户的反馈对软件做适当的修正。

2. 维护阶段中的培训要求

在软件交付使用的同时，也要制定培训计划，编写培训材料。培训计划包括对系统进行概览，对系统假定的一些错误给出处理的方法。培训材料包括用户使用方法、对错误列表上的问题给出解释、对输入数据进行解释等。

3.3　支持生命周期软件测试的工具

目前，能够比较好地支持生命周期软件测试的工具主要有微软的 VS 2012、IBM Rational 的一整套自动化测试工具、HP ALM 及配套测试工具，以及 NSEsoftware、LLC 的 Panorama++。为了较好地揭示生命周期软件测试的概念，这里主要以 Panorama++ 和 HP ALM 为例进行工具介绍。

3.3.1　全生命周期质量管理平台 Panorama++

Panorama++作为一个完整的软件工程与量化质量测评管理系统，支持软件开发的整个生命周期，如图 3-4 所示。

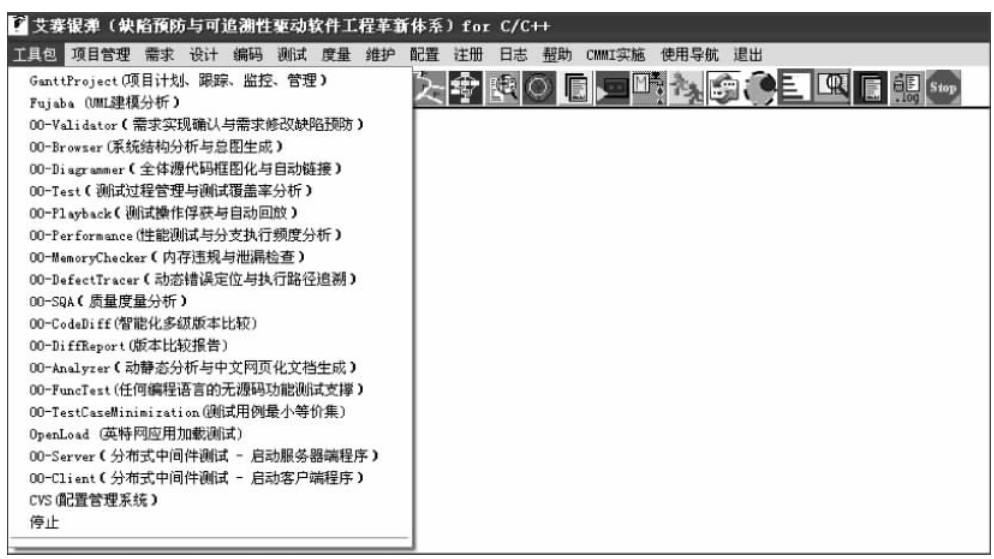

图 3-4　Panorama++工作界面

(1) 项目管理支撑：支持立项预估、规划、监控、结项等。

(2) 需求阶段支撑：支持 UML 建模分析、管理以及设计和源码间的自动追溯查错等。

(3) 设计阶段支撑：支持大型系统的快速分层设计、源码框架生成、逆向工程。

(4) 编码阶段支撑：支持增量式、高一致性与低风险编程。

(5) 测试阶段支撑：能够支持测试阶段的各种测试活动和测试方法。包括：

① 基于复杂度分析的测试规划；

② 基于整个被测软件的全部源代码的框图化与自动链接的源代码审议和走查支撑；

③ 界面功能测试(采用线性脚本的动作自动俘获与回放，有无源代码均可，后者与所使用的计算机语言种类无关，前者则可与“白盒”结构测试无缝地相结合)；

④ 性能(动态用时分配)测试分析包括各程序分支的执行频度分析；

⑤ “白盒”结构测试，支持美国和欧洲航空航天最高质量标准 RTCA/DO178B-LEVEL A 的 MC/DC(修改条件/判断覆盖)测试覆盖率分析；

⑥ 因特网应用程序的加载测试、分布式中间件与服务器端和客户端的应用程序的配对测试支撑；

⑦ 内存泄漏与违规使用检测、高效率测试用例设计支撑、测试用例有效性分析与测试用例最小等效集的自动生成。

(6) 度量与分析支撑：面向对象的个性化度量项的选择与度量标准的设定支持，动态与静态质量数据的自动收集、自动分析以及多形式的分析结果彩图显示。

(7) 维护阶段支撑：动态运行错误自动定位、错误执行路径追溯、高一致性源码修改支撑、高一致性数据修改支撑。

(8) 配置支撑：版本维护与管理、智能化多级版本比较等。

Panorama++采用以预防错误发生、杜绝错误传递为核心的设计理念，实现了软件系统需求、设计、编码、测试、维护、文档多环节自动相互追溯、精确定位图形化显示，从而也实现了软件全生命周期的测试。

如果没有一个需求及实现该需求的源码间的自动追溯工具，就很难判断功能测试是否充分，很难判断一个需求是否被完全实现，很难确定源码中的一个函数是否真有必要存在于该系统——也许它来自其他已经删去的需求；很难找出其不一致性，也不利于需求和源码的修改维护——一个模块的修改可能关系到一个以上的需求，遗漏了就会产生严重的后果。Panorama++的自动追溯精确定位功能，可以使得软件开发的每个阶段均有测试活动的参与，可以轻松面对系统开发环节中的任意变化，如图 3-5 所示。

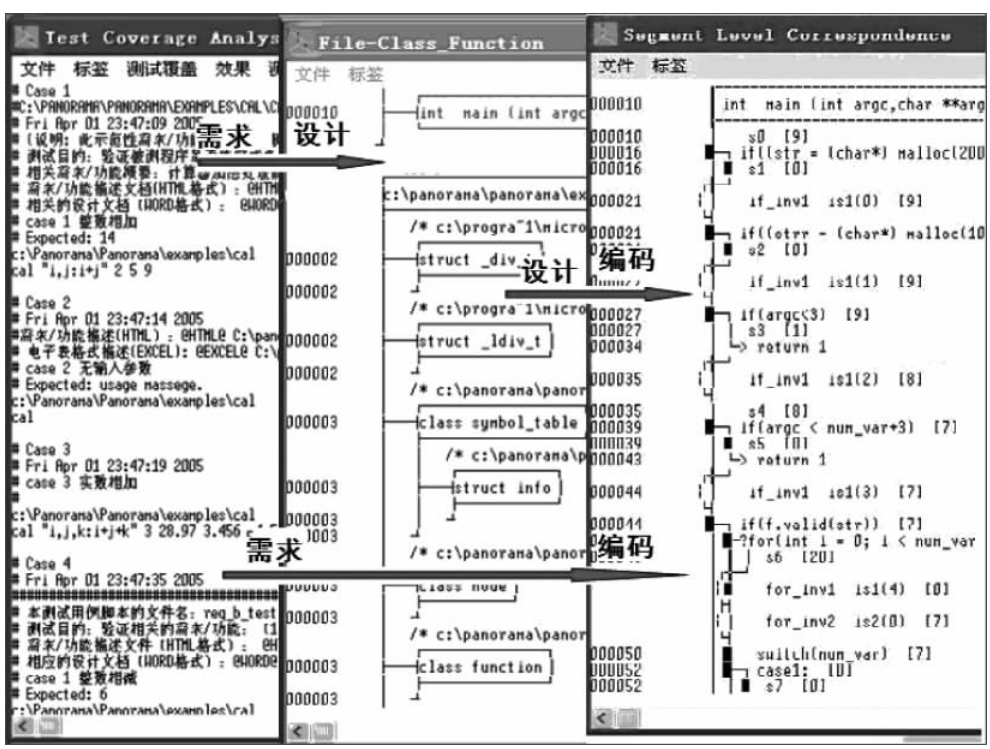

图 3-5 需求、设计、编码和测试之间的自动追溯

图 3-5　（续）

3.3.2　应用生命周期管理系统 HP ALM11

HP ALM11 是一个以任务为导向的系统，可在应用交付过程中支持各参与方，并与主要开发工具相整合。该方案实现了团队内和不同团队间的工作流程自动化，强化并加速了应用

生命周期管理和各阶段的测试。

1. HP ALM11 能为客户带来的好处

1）管理方面

（1）HP ALM11 项目规划及跟踪（ALM11 Project Planning and Tracking），建立了发布标准并在整个流程中基于实时监测来管理发布进程及准备情况。

（2）需求、开发及质量工具间的三路追踪，实现了对应用变化的快速组内分析及执行。

（3）通过惠普敏捷加速器 4.0（HP Agile Accelerator 4.0），灵活地按项目类型（瀑布式项目、定制项目、敏捷项目）支持优化交付方式。

（4）减少因应用故障导致的业务风险，这些应用故障来自由混合以及富互联网应用引起的在功能、性能和安全方面的缺陷。

（5）通过易于发现、重复利用以及分享关键应用工具（包括需求、测试及缺陷检测），降低成本并缩短交付时间。

2）测试实施方面

HP ALM11 能简化和自动化应用质量和性能验证，从而降低运营成本，并将更多的资源投入到新应用及服务中。它能带来如下好处。

（1）借助 HP Sprinter，通过自动化手动测试（如数据创建，以及在多环境下进行的重复手动测试）来加速应用部署。

（2）通过 HP TruClient 改善测试创建，TruClient 是 HP Load Runner 11.0 的一部分，主要对应用性能进行测试，无须执行耗时的脚本处理。

（3）借助 HP Unified Functional Testing 11.0，为复合应用提供单一自动化解决方案，可降低 GUI 和非 GUI 测试应用功能故障。该方案由惠普功能测试（HP Functional Test）和惠普服务测试 11（HP Service Test 11）组成。

2. HP ALM 的主要特性

HP ALM 有如下的主要特性。

1）综合管理

HP ALM 通过横跨整个应用交付流程的公共平台，为不同团队提供了不同工具组合。这些团队包括企业架构师、业务分析员、开发人员、质量保证（QA）专业人员、安全专家及生产团队。HP ALM 为规划、创建及发布高复杂性应用提供了一套完整的视角。该平台提供了可轻松拓展、现代化开放的架构。

2）增强可预见性

HP ALM 中全新的项目规划及追踪解决方案可精确地预测项目进展，从而增强可预见性。在项目经理制定了项目计划、时间点、关键绩效指标（KPI）及每个任务的退出条件后，HP ALM 在应用生命周期中可自动追踪不符合已制定的时间点的活动及关键绩效指标，并向相关负责人发出警告。通过这样一个可信赖的、能够时时监控状态的平台，哪怕最复杂的应用项目都可以轻松管理。

3）更密切的业务合作

HP ALM 使用常见工具创建应用需求，促进了业务、应用开发人员和质量保证专业人员间的交流，这些工具包括：

（1）业务流程模型真实地展现了工作进程，使业务分析员能创建一整套应用需求，避免重复或疏忽，从而使业务分析员、质量保证人员、开发人员及安全人员可高效合作，以创建满足业务需求的应用。

(2) 丰富的文本编辑器(rich text editor)提供了与微软 Word 相似的数据输入功能,可将应用需求输入 HP ALM 中,这大大加快了业务分析员使用速度。

(3) 定制化的模板及工作流程使业务分析员在企业统一的架构中轻松捕获应用需求,从而避免了重复劳动,并可一致、有效且明确地定义需求,获得高品质应用。

4) 增强协作

IDEs 对于增强应用团队中相关各方的协作非常重要。HP ALM 预先集成在全部主要 IDEs 中,从而使开发人员不必使用其他工具就能直接从工作环境中查看应用需求及缺陷。HP ALM 与微软的 Visual Studio/TFS 和 IBM 的 Eclipse 进行了集成,实现即开即用,可在需求、缺陷检测和源代码间建立可追溯性。

5) 单一的综合平台

HP ALM 平台,助力应用团队管理整个应用周期内的测试,包括监控项目的生命周期状态。因而可使客户改善应用的可预见性、可重复性和质量,同时做好准备应对从业务构想到退出的各种变化。

(1) HP Quality Center 助力客户达成:

① 通过对需求测试及缺陷检测的详细追踪改善应用质量。

② 根据风险级别调整和确定测试工作优先级,实现资源优化。

③ 通过对应用交付流程的实时报告增强可预见性。

④ 通过应用 HP Sprinter,提高手动测试效率并增强创新性。

⑤ 通过一个共享的需求、测试及缺陷检测的中央储存文件来增强可重复性。

(2) 惠普性能中心 11.0(HP Performance Center 11.0)助力客户达成:①通过对需求测试及缺陷检测的详细追踪改善应用性能;②在分散团队中通过增强对 COE(Centers Of Excellence)的支持,实现效率最大化,其中包括版本控制、资产共享和项目分组;③通过包含整个生命周期的应用质量、性能和安全等内容的统一仪表板增进协作性;④通过测试环境下的拓扑型系统基础设施视图增强可视性。

3. 应用程序生命周期管理过程

ALM 能够帮助我们组织和管理应用程序生命周期管理过程的所有阶段,包括定义版本、指定需求、计划测试、执行测试和跟踪缺陷。

使用 ALM 的应用程序生命周期管理过程包括以下阶段。

(1) 指定版本:制定一个发布周期管理计划,更高效地管理应用程序发布和周期。追踪应用程序发布,并根据计划确认发布是否正常。

(2) 指定需求:分析应用程序并确定需求。可以跨多个发布和周期管理需求,并在需求、测试、缺陷之间实现多维追踪。ALM 为需求覆盖和关联到质量评估和商业风险中的缺陷提供实时可见的功能。

(3) 计划测试:创建一个基于需求的测试计划,ALM 为手动和自动测试都提供了知识库。

(4) 执行测试:创建测试集,完成测试运行。ALM 支持健壮测试、功能测试、回归测试、更高级测试。根据计划来执行测试,从而识别和解决问题。

(5) 追踪缺陷:报告在应用程序中检测到的缺陷,跟踪修复进程。分析缺陷和缺陷趋势,帮助做出合理的“执行/不执行”决策。ALM 支持完整的缺陷生命周期——从初始问题检测到修复缺陷以及确认缺陷修复。

在每个阶段,可以通过生成的详细报告和图表来分析数据。

ALM 是一个基于 Web 的工具，用来给基于 Web 的项目创建知识库。它是一个用 J2EE 开发的客户端/服务器组织结构，是一个服务的集合体。

习题

1. 详细说明软件工程生命周期 V 形图的含义。
2. 怎样才能把好软件工程生命周期各个阶段的质量关？
3. 什么是生命周期测试方法？生命周期测试如何开展？
4. 生命周期测试有哪些测试任务？简述测试策略、测试要素及测试风险各自的含义。
5. 举例说明计算机系统的风险表现。简述基于风险的软件测试方法。
6. 如何制定测试计划？在制定测试计划时，应考虑哪些因素？
7. 测试阶段有哪些测试内容？用到哪些技术？
8. 需求阶段、设计阶段、编码阶段、测试阶段、安装阶段、验收阶段及维护阶段需要进行哪些测试？
9. Panorama++和惠普 ALM 是如何支持生命周期测试的？