西北工業大學
NORTHWESTERN POLYTECHNICAL UNIVERSITY

第五章　软件测试的过程管理

授课教师：　郑炜

第五章 软件测试的过程管理

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

- 5.1 软件测试的各个阶段
- 5.2 测试需求
  - 5.2.1 测试需求的分类
  - 5.2.2 测试需求的收集
  - 5.2.3 测试需求的分析
  - 5.2.2 测试需求的评审
- 5.3 测试计划
  - 5.3.1 测试计划的目标
  - 5.3.2 制定测试计划
  - 5.3.3 划分测试用例优先级

- 5.4 测试设计及测试用例
  - 5.4.1 测试用例的设计原则
  - 5.4.2 测试用例的设计方法
  - 5.4.3 测试用例的粒度
  - 5.4.5 测试用例的评审
- 5.5 测试的执行
  - 5.5.1 测试用例的选择
  - 5.5.2 测试人员分工
  - 5.5.3 测试环境的搭建
  - 5.5.5 BVT测试与冒烟测试
  - 5.5.5 每日构建介绍

- 5.6 软件缺陷分析
  - 5.6.1 软件缺陷分析的作用
  - 5.6.2 软件缺陷的分类
  - 5.6.3 软件缺陷分析方法
  - 5.6.5 软件缺陷分析的流程
  - 5.6.5 软件缺陷报告

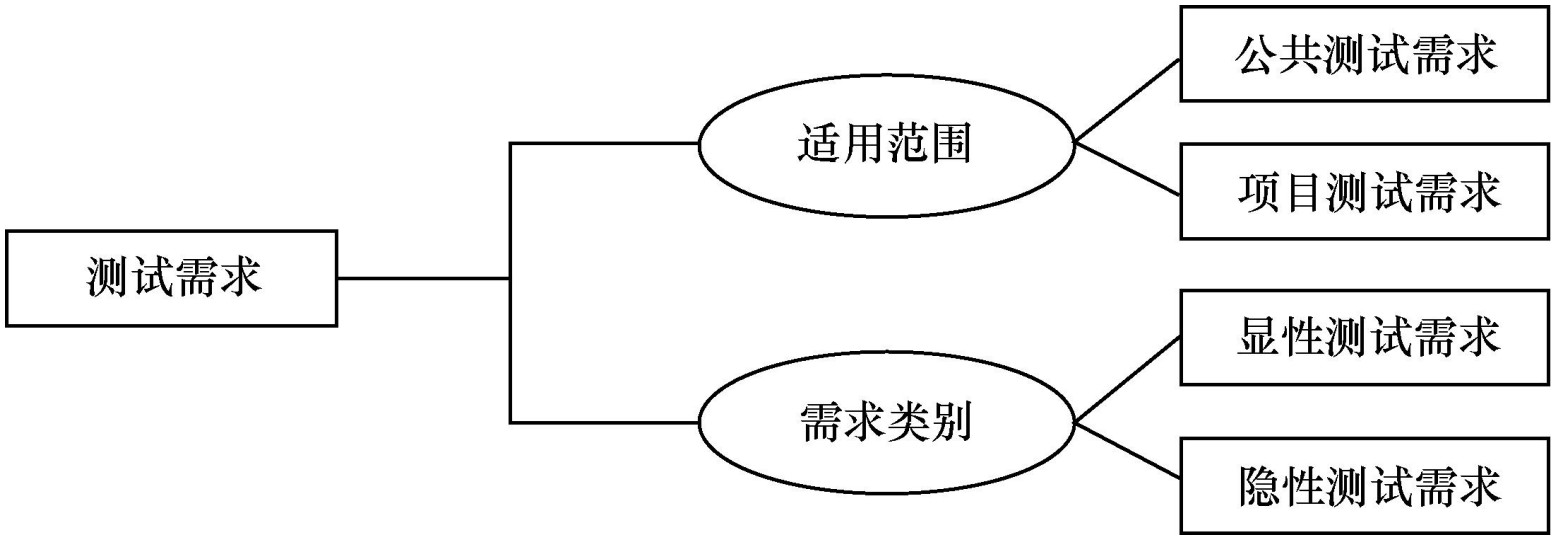
5.1.1 软件测试的各个阶段

(1) 测试需求的分析和确定
(2) 测试计划
(3) 测试设计
(4) 测试执行
(5) 测试记录和软件缺陷跟踪
(6) 回归测试
(7) 测试总结报告

- 5.2 测试需求
  - 5.2.1 测试需求的分类
  - 5.2.2 测试需求的收集
  - 5.2.3 测试需求的分析
  - 5.2.2 测试需求的评审

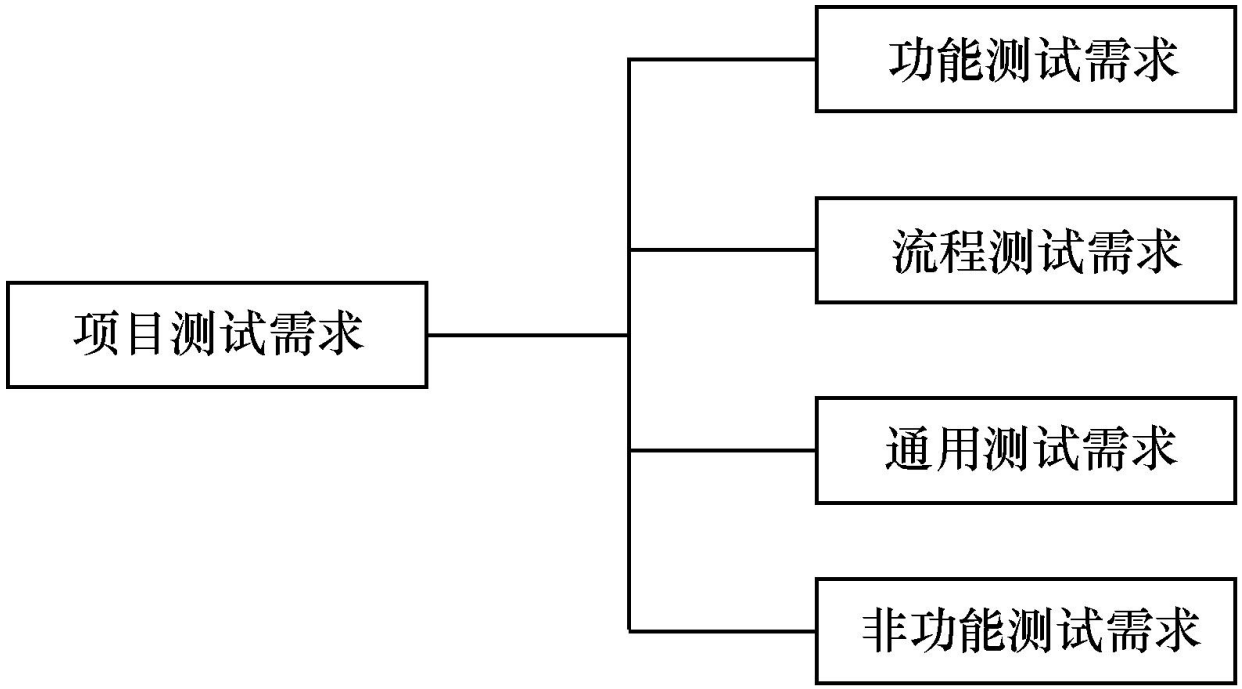
5.2.1 测试需求的分类

- 测试需求按适用范围分为公共测试需求和项目测试需求，按需求类别分为显性测试需求和隐性测试需求。



5.2.1 测试需求的分类

- 项目测试需求又分为功能测试需求、流程测试需求、通用测试需求及非功能测试需求



5.2.2 测试需求的收集

- 软件测试需求的主要来源是系统需求说明书（软件规格说明书），有了系统需求说明书基本就能画出系统。测试需求还可以通过其他途径来获得。

（1）与待测软件相关的文档资料，如用例、界面设计、项目会议或与客户沟通时关于需求信息的会议记录、其他技术文档等。

（2）与客户或系统分析员的沟通记录。

（3）业务背景资料，如待测软件业务领域的知识等。

（4）正式与非正式的培训资料。

5.2.3 测试需求的分析

- 软件需求分析、设计和实现阶段出现的问题是软件的主要错误来源。因此一旦确定软件需求，即可开始进行测试需求分析。

- 在收集完测试需求后，需要根据测试阶段和重点整理测试需求。
- 测试需求采集之后得到的是一张没有优化的需求表，需要对这份原始需求表进行初步的规划：删除冗余重复的需求，各个需求间没有过多的交集；需覆盖业务流程、功能、非功能方面的需求。
- 在做测试需求分析时需要列出以下类别。

（1）常用的或规定的业务流程。
（2）各业务流程分支的遍历。
（3）明确规定不可使用的业务流程。
（4）没有明确规定但是应该不可以执行的业务流程。
（5）其他异常或不符合规定的操作。

5.2.4 测试需求的评审

- 测试需求评审的内容包括完整性审查和准确性审查
  - 完整性审查是检查测试需求是否覆盖了所有软件需求，以及软件需求的各项特征，关注功能要求、数据定义、接口定义、性能要求、安全性要求、可靠性要求、系统约束、行业标准等，同时还要关注系统隐含的用户需求。
  - 准确性审查是检查测试需求是否清晰、没有歧义、描述准确，是否能获得评审各方的一致理解，每项测试需求是否都可以作为设计测试用例的依据。

测试需求评审可以采取正式的小组会议形式

- 5.3 测试计划
  - 5.3.1 测试计划的目标
  - 5.3.2 制订测试计划
  - 5.3.3 划分测试用例优先级

5.3 测试计划

- **测试计划活动一般包括以下8项**
  - （1）定义测试的整体方式和策略。
  - （2）确定测试环境。
  - （3）定义测试级别及它们之间的协作，将测试活动集成到其他项目活动中并进行协调。
  - （4）确定如何评估测试结果。
  - （5）选择监视和控制测试工作的度量，并定义测试出口准则。
  - （6）确定要准备的测试文档，并确定模板。
  - （7）编写测试计划并确定测试的内容、人员、进度和测试范围。
  - （8）估算测试工作量和成本，（再）估计和（再）计划测试任务。

5.3.1 测试计划的目标

- **测试计划所要达到的目标有以下6点**

（1）为测试各项活动制订一个现实可行的、综合的计划，内容包括每项测试活动的对象、范围、方法、进度和预期结果。

（2）为项目实施建立一个组织模型，并定义测试项目中每个角色的责任和工作内容。

（3）开发有效的测试模型，能正确地验证正在开发的软件系统。

（4）确定测试所需要的时间和资源，以保证其可获得性和有效性。

（5）确立每个测试阶段测试完成及测试成功的标准和要实现的目标。

（6）识别出测试活动中的各种风险，并消除可能存在的风险，降低由不可能消除的风险所带来的损失。

5.3.2 制订测试计划

（1）确定测试范围

- 首先要明确测试的对象，有些对象是不需要测试的。例如，大部分软件系统的测试不需要对硬件部分进行测试，而有些对象则必须进行测试。

（2）制订测试策略

- 测试策略一般描述软件测试活动的一般方法和目标，其中包括要进行的测试阶段（单元测试、集成测试和系统测试），以及要执行的测试类型（功能测试、性能测试、负载测试、强度测试等）。

（3）确定测试任务

- 根据测试阶段的测试需求，细化测试任务。其中包括划分测试任务的优先级、确定各任务与主要任务之间的关联关系、确定辅助任务清单。

5.3.2 制订测试计划

（4）确定测试资源与工作量

- 通过充分估计测试的难度、测试的时间、工作量等因素，决定测试资源的合理利用。可以依照测试对象的复杂度和具体标准，结合相关的数据对测试要完成的工作量进行估计，进一步确定需要的测试资源。

（5）进度安排

- 收集与进度相关的信息，如总体工作量估算、人员数量、关键资源、项目时间安排，结合项目的开发计划、产品的整体计划和测试本身的各项活动进行进度安排。

（6）风险及对策

- 计划的风险一般来源于项目计划的变更、测试资源不能及时到位等方面。若出现计划变更的情况，应该及时让测试人员知道具体的形势，以及变更所带来的影响，这样才能快速地采取相应的补救措施。

5.3.3 划分测试用例优先级

- **用于划分测试用例优先级及确定测试用例执行顺序的准则**

（1）使用频率或失效的概率。系统的某些特定的被经常使用的功能优先级更高。

（2）失效的风险。风险是严重性和失效概率的综合结果，高风险失效的用例应该比低风险失效的用例具有更高的优先级。

（3）失效的可见性。失效对用户的可见性是划分测试优先级的更进一步准则。

5.3.3 划分测试用例优先级

- **用于划分测试用例优先级及确定测试用例执行顺序的准则**

（4）需求的优先级。系统提供的不同功能对于客户来说，其重要性也不尽相同。某些功能不能正常工作，客户也许能够接受，但有些功能则不可或缺。

（5）除了功能需求，质量特性对于客户也具有不同的重要性。必须测试重要的质量特性是否已经正确实现，并保证用于验证与必要的质量特性是否一致的测试用例具有更高优先级。

（6）从系统架构开发人员的角度来确定。失效时会导致严重后果（如系统崩溃）的组件需要加强测试。

5.3.3 划分测试用例优先级

- **用于划分测试用例优先级及确定测试用例执行顺序的准则**

（7）单独的组件和系统组件的复杂性。复杂的程序组件需要加强测试，因为开发人员可能在其中引入较多的软件缺陷。不过，看起来简单的程序组件也可能会因为开发不够细致而包含很多软件缺陷。因此，对这个领域划分优先级时，应该参考从组织中早期项目得来的经验数据。

（8）存在高项目风险的失效应该尽早被发现。这些失效需要做大量的修正工作，否则会独占资源并导致项目明显延迟。

（9）项目经理应该为项目定义充分的优先级准则和优先级类别。

- 5.4 测试设计及测试用例
  - 5.4.1 测试用例的设计原则
  - 5.4.2 测试用例的设计方法
  - 5.4.3 测试用例的粒度
  - 5.4.5 测试用例的评审

5.4.1 测试用例的设计原则

- **设计测试用例时应遵循以下原则**

（1）正确性。满足需求规格说明书的要求；覆盖需求规格说明书中的各项功能。

（2）全面性。覆盖所有的需求功能项。

（3）整体连贯性。测试用例执行粒度尽量保持每个用例有一个测试点，不能同时覆盖很多功能点。

（4）可维护性。由于软件开发过程中需求变更等原因的影响，常常需要对测试用例进行修改、增加、删除等。

（5）测试结果可判定性和可再现性。测试结果可判定性是指测试执行结果的正确性是可判定的，每个测试用例都应有相应的期望结果。测试结果可再现性是指对于同样的测试用例，系统的执行结果是相同的。

5.4.2 测试用例的设计方法

1．等价类划分法

等价类划分法是一种典型的黑盒测试用例的设计方法。采用等价类划分法时，完全不用考虑程序的内部结构，设计测试用例的唯一依据是软件需求规格说明书。

- 等价类：输入条件的一个子集合，该集合中的数据对于揭示程序中的错误是等价的。
- 有效等价类代表对程序有效的输入，而无效等价类则是其他任何可能的输入（即不正确的输入值）。
- 有效等价类和无效等价类都是使用等价类划分法设计用例时所必需的，因为被测程序若是正确的，就应该既能接受有效的输入，也能经受住无效输入的考验。

1．等价类划分法示例

- 要求：注册用户名要求7～12个字符，可以由字母、数字、下画线构成；下画线不能作为开头。该示例的等价类划分如表5-1所示。

输入条件	有效等价类	无效等价类
用户名	7～12个字符（1）	少于7个字符（2） 多于12个字符（3） 空（4）
用户名	由字母、数字、下画线构成（5）	含有除字母、数字、下画线以外的特殊字符（6） 非打印字符（7） 中文字符（8）
	以字母、数字开头（9）	以下画线开头（10）

2．边界值分析法

边界值分析法就是对输入或输出的边界值进行测试的一种黑盒测试方法。

3．基本路径分析法

基本路径分析法是在程序控制流图的基础上，通过分析控制构造的圈复杂性，导出基本可执行路径集合，从而设计测试用例的方法。利用此方法设计出的测试用例要保证在测试程序的每个可执行语句中都至少执行一次。

4．因果图法

因果图法是一种利用图解法分析输入的各种组合情况，从而设计测试用例的方法。它适合于检查程序输入条件的各种组合情况。

5.4.3 测试用例的粒度

- **测试用例的设计粒度需要考虑以下4个方面的因素。**

（1）复用率：如果产品在不断地更新版本，复用率很高，测试用例的粒度需要更加细化；相反，如果产品的使用频率不高，那么测试用例就不需要设计得很复杂。

（2）项目进展：需要根据项目的进展而定，如果当前项目的时间充足，可以将测试用例设计得详细一些，但是如果项目距离截止时间较近，没有多余的时间，则需要将测试用例设计得简单一些。

（3）使用对象：如果所设计的测试用例在测试过程中是提供给多个测试人员使用，需要将测试用例设计得详细一些。

（4）测试的种类：如果采用验收测试，那么相应的测试用例的粒度就比较大。如果是系统测试，那么测试用例的颗粒度就相对较小。

5.4.4 测试用例的评审

- **下面是测试用例评审的一些检查项。**

(1) 测试用例是否是按照公司定义的模板进行编写的。
(2) 测试用例本身的描述是否清晰，是否存在二义性。
(3) 测试用例内容是否正确，是否与需求目标相一致。
(4) 测试用例的期望结果是否确定、唯一。
(5) 测试用例的操作步骤与描述是否相一致。
(6) 测试用例是否覆盖了所有的需求。
(7) 测试用例的设计是否存在冗余性。
(8) 测试用例是否具有可执行性。
(9) 是否从用户层面来设计用户使用场景和业务流程的测试用例。

- 5.5 测试的执行
  - 5.5.1 测试用例的选择
  - 5.5.2 测试人员分工
  - 5.5.3 测试环境的搭建
  - 5.5.5 BVT测试与冒烟测试
  - 5.5.5 每日构建介绍

5.5.1 测试用例的选择

- **下面是一些测试用例的选择策略**

（1）首先测试产品的核心功能，再测试其他功能

（2）若产品具有支付交易功能，则需要首先测试此功能，再测试其他功能，因为资金的问题永远是最重大的问题。

（3）首先测试常用功能，再测试其他功能。

（4）首先测试需求中被特别说明的地方，再测试没有特别说明的地方

（5）首先测试有变更的地方，后测试没有变更的地方

5.5.2 测试人员分工

（1）按照测试内容分工

- 一个项目的测试包括文档测试、易用性测试、逻辑功能测试、界面测试、配置和兼容等多个方面，管理者可以根据人员的特点为每个人员分配不同的测试内容。这样分配的优点在于分工较为明确，测试人员对于测试内容的重点具有清楚的认识。

（2）按照测试流程分工

- 项目的测试流程一般包括测试需求的检查、测试用例的设计、测试执行、输出测试报告等工作，管理者可以根据测试流程中的各个阶段来进行人员的分工。这样分配具有流程清晰的特点。

（3）按照功能模块分工

- 一般规模较大的软件项目所具有的功能模块较多，针对已划分好的功能模块给予相应的测试，不同的测试人员负责不同模块的测试工作。这样分配具有人员利用率高且容易发现深层错误的特点。

（4）按照测试类型分工

- 软件测试根据软件开发的阶段可以划分为单元测试、集成测试、系统测试、回归测试等测试类型，管理者可以根据这些类型为测试人员分配测试工作。这样分配具有对测试人员的专业性要求高的特点。

5.5.3 测试环境的搭建

● 测试环境的搭建是进行测试执行活动中的一项重要工作，它所需要的时间也较多。一般测试环境的搭建包括图5-5所示的内容。

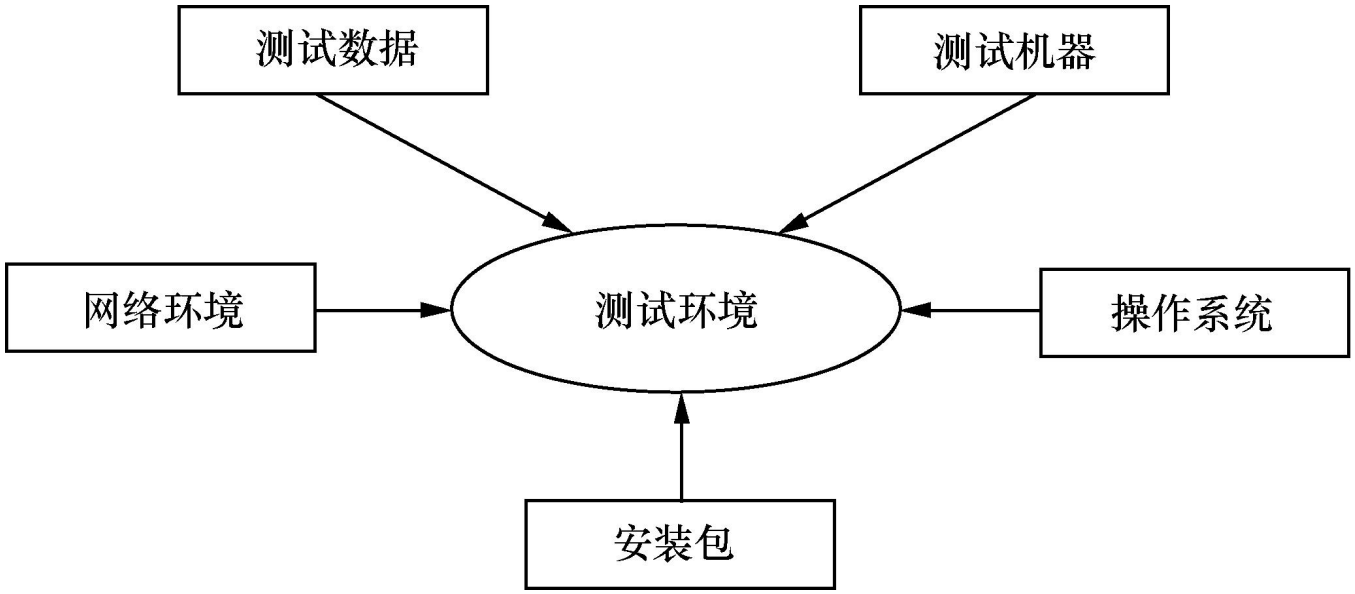


图5-5　测试环境包括的内容

5.5.4 BVT测试与冒烟测试

构建验证测试（Build Verification Test，BVT）

- BVT测试只验证构建（Build）的成功与失败，不深入测试构建好的Build的功能、性能等。BVT是在所有开发工程师都已经检查完自己的代码，项目组编译生成当天的版本之后进行。
- 主要目的是验证最新生成的软件版本在功能上是否完整，主要的软件特性是否正确。如无大的问题，就可以进行相应的功能测试。
- BVT的优点是时间短，验证了软件的基本功能。BVT的缺点是覆盖率低，因为其运行时间短，不可能把所有的情况都测试到。

冒烟测试

- 冒烟测试是对软件的基本功能进行测试，测试的对象是每个新编译的需要正式测试的软件版。
- 目的是确认软件的基本功能是否正常，保证软件系统能够正常运行，并且可以进行后续的正式测试工作。如果最基本的测试都有问题，就需要向开发人员进行反馈，所以正式交付测试的版本首先必须通过冒烟测试的考验。
- 冒烟测试，只是一个测试活动，并不是一个测试阶段。也就是说，冒烟测试贯穿于测试的任何一个阶段，单元测试里会有冒烟测试、集成测试里会有冒烟测试、系统测试里也会有冒烟测试。

5.5.4 BVT测试与冒烟测试

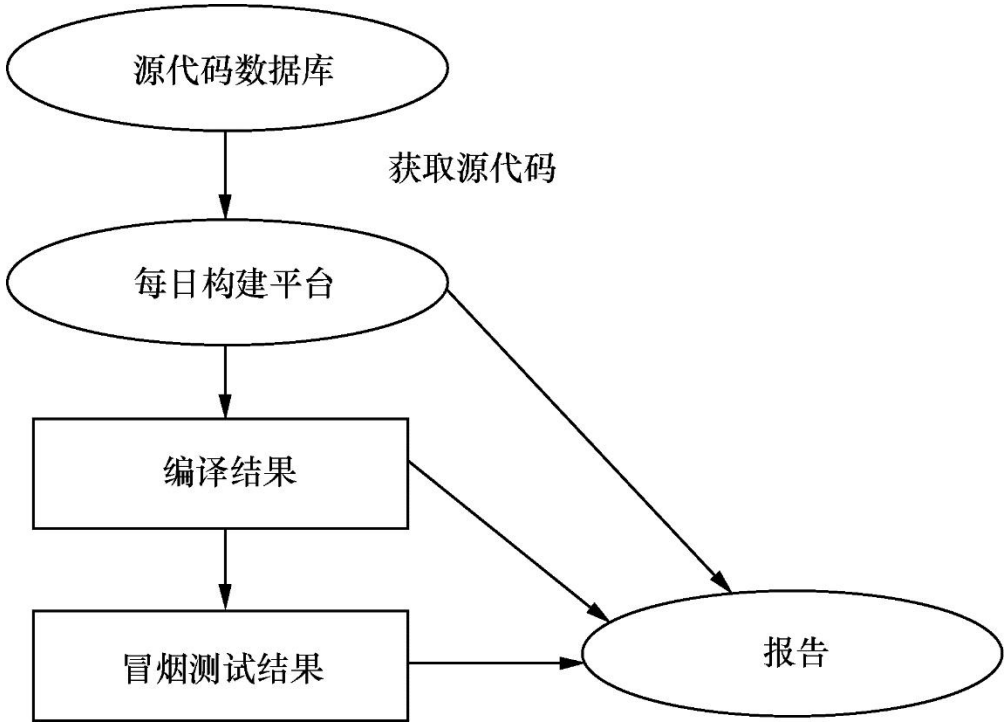
BVT与冒烟测试具有以下区别

- BVT只在构建完成时进行，而冒烟测试在各个阶段都会进行。
- BVT可以加入自动测试脚本并执行少量固定的自动化测试，但冒烟测试与Build的验证无关。
- BVT的结果直接决定新构建的Build是否交付后续测试，而冒烟测试并不影响其他日常测试工作。

5.5.5 每日构建介绍

- 每日构建也可称为持续集成（Continuous Integration）。

它强调完全自动化的、可重复的创建过程，其中包括每天运行多次的自动化测试。



5.5.5 每日构建介绍

- 每日构建具有以下优点

（1）进度可见并可以控制到1～2天的细粒度，很容易看到进度的偏差。

（2）可以尽早发现软件缺陷并分析解决，从而提高软件质量。

（3）由于将大集成分解到每日构建中的小集成，消除了传统产品集成或集成测试时出现严重问题的可能性。

（4）注重每次工作的正确性，减少了可能出现的错误。

- 5.6 软件缺陷分析
  - 5.6.1 软件缺陷分析的作用
  - 5.6.2 软件缺陷的分类
  - 5.6.3 软件缺陷分析方法
  - 5.6.5 软件缺陷分析的流程
  - 5.6.5 软件缺陷报告

5.6.1 软件缺陷分析的作用

- 软件缺陷所具有的含义不仅仅局限于程序中存在的软件缺陷。它所包含的范围更加广泛，除了源程序外，同时还包括一些关键的文档资料，如项目计划、需求规格说明、设计文档、测试用例、用户手册等存在的错误和问题。
- 软件测试的任务就是发现软件系统的缺陷，保证软件的优良品质。通过软件缺陷分析，发现各种类型软件缺陷发生的概率，掌握软件缺陷集中产生的区域、明晰软件缺陷的发展趋势、了解软件缺陷产生的主要原因，以便有针对性地提出遏制软件缺陷发生的措施、降低软件缺陷数量。
- 软件缺陷分析对于改进软件开发、提高软件质量有着十分重要的作用。软件缺陷分析报告中的统计数据及分析指标既是对软件质量的权威评估，也是判定软件是否能发布或交付使用的重要依据。

5.6.2 软件缺陷的分类

（1）按严重程度划分

按照严重程度由高到低的顺序可以将软件缺陷分为5个等级：系统崩溃、严重、一般、次要、建议。需要说明的是，在具体的项目中，需要根据实际情况来划分等级，不一定是5个等级。如果软件缺陷数比较少，就可以划分为3个等级：严重、一般、次要。一般的软件缺陷管理工具会自动给出一个默认的软件缺陷严重程度划分。

（2）按优先级划分

按照缺陷修复的优先级由高到低可以划分为3个等级：高（high）、中（middle）、低（low）。其中，高优先级的软件缺陷是应该立即修复的软件缺陷；中优先级的软件缺陷是应该在产品发布之前修复的软件缺陷；低优先级的软件缺陷是指如果时间允许，应该修复的软件缺陷或是可以暂时存在的软件缺陷。优先级的这种分法也不是绝对的，需要根据实际情况灵活划分。

5.6.2 软件缺陷的分类

（3）按测试种类划分

按测试种类可以将软件缺陷分为逻辑功能类、性能类、界面类、易用性类、兼容性类。这种划分方式可以了解不同测试方法所能发现的软件缺陷的比例，以便在测试的时候有所侧重。

（4）按功能模块划分

一般的软件产品都是分为若干个功能模块的，根据二八定理可知，通常80%的软件缺陷集中在20%的模块中，测试的过程可以统计软件缺陷主要集中在哪些模块，以便投入重点精力去测试。

5.6.3 软件缺陷分析方法

（1）正交缺陷分类分析方法

正交缺陷分类（Orthogonal Defect Classification，ODC）分析方法是将一个软件缺陷在生命周期的各环节的属性组织起来，从单维度、多维度对软件缺陷进行分析，从不同角度得到各类软件缺陷的缺陷密度和缺陷比率，从而积累得到各类软件缺陷的基线值，用于评估测试活动

（2）Gompertz分析方法

冈珀茨（Gompertz）分析方法是在利用已有数据的基础上，对软件测试过程进行定量分析和预测，对软件产品质量进行定量评估，对是否结束软件测试任务给出判断依据。

5.6.3 软件缺陷分析方法

（3）DRE/DRM分析方法

DRE（Defect Removal Efficiency）分析方法是通过已有项目历史数据，得到软件生命周期各阶段缺陷注入和排除的模型，用于设定各阶段质量目标，评估测试活动。DRE主要针对历史数据，矩阵的每一列代表软件缺陷在何时（什么阶段）引入（产生），每一行代表发现软件缺陷时开展的工作。矩阵中的数值代表已经发现的软件缺陷数量。

5.6.3 软件缺陷分析方法

表5-3所示的矩阵的目标是要分别计算出各个阶段的软件缺陷移除率为后面所用。软件缺陷移除率的定义为当前阶段工作实际发现的软件缺陷数量占当前阶段应该发现的软件缺陷数量的比值。

注入阶段 / 发现阶段	需求	设计	编码	发现总计
需求阶段	8	—	—	8
设计阶段	26	62	—	88
编码阶段	4	11	12	27
功能测试阶段	4	3	112	119
系统测试阶段	0	0	28	28
注入总计	42	76	152	270

5.6.4 软件缺陷分析的流程

（1）确定分析指标

① 反映产品质量的指标：

软件缺陷密度 = 软件缺陷数量 / 软件规模

潜在软件缺陷数 = (100% − 发布前软件缺陷去除率)× 软件缺陷密度

② 反映产品可靠性的指标：

平均失效时间 = 软件持续运行时间 / 软件缺陷数量

③ 反映软件缺陷发现及修复效率的指标：

软件缺陷检出率 = 某阶段当时发现的软件缺陷 / 属该阶段的全部软件缺陷× 100%

发布前软件缺陷去除率 = 发布前发现的软件缺陷 / (发布前发现的软件缺陷+ 软件运行的前3个月发现的软件缺陷)× 100%

软件缺陷修正率 = 修复过程中未引发其他问题的软件缺陷数 / 被修复软件缺陷的总数 ×100%

④ 反映软件缺陷修复成本的指标：

$$\text{平均修复时间} = \frac{\sum \text{软件缺陷修复时间}}{\text{软件缺陷数量}}$$

平均修复成本 = 开发人员的平均人力成本× 平均修复时间

相对返工成本 = 返工的工作量 / 项目总工作量 ×100%

5.6.4 软件缺陷分析的流程

（2）实施软件缺陷分析过程

在确定好相应的指标之后，可以借助相应的方法和工具实施软件缺陷分析的过程，产生分析的结果。

（3）汇总统计

在软件缺陷分析中可以使用统计方法对收集的变更进行分类、汇总。

- 软件缺陷性质统计：变更性质属性一般分为软件缺陷变更和需求变更两种。
- 软件缺陷状态分布：变更状态属性分类很多，但在软件缺陷分析中可以不用分得很细，可以分为关闭、挂起和处理中3种类型。该分析主要反映软件缺陷修改完成情况。
- 软件缺陷按产品分类统计：该分析能显示各软件子系统的软件缺陷分布情况。

5.6.4 软件缺陷分析的流程

- 软件缺陷按原因分类统计：按变更的根本原因属性进行分类统计软件缺陷，统计不包括需求变更。该分析能揭示软件缺陷原因的分布。
- 软件缺陷测试情况统计：统计仅涉及变更的根本原因是系统设计、程序编码、维护和外部问题等软件缺陷变更。该分析能暴露软件测试本身存在的问题。
- 软件缺陷来源统计：该分析主要反映用户或软件代理的地区分布，发现一些客户分布规律。

5.6.5 软件缺陷报告

- 一般情况下，软件缺陷报告包含下列内容。

（1）软件缺陷报告编号：为了便于对软件缺陷进行管理，每个软件缺陷都必须被赋予唯一的编号，编号规则可根据需要和管理要求制订。

（2）标题：软件缺陷报告的标题可以用简明的方式传达软件缺陷的基本信息。软件缺陷报告的标题应该简短并尽量做到唯一，以便在观察缺陷列表时，可以很容易注意到。

（3）报告人：软件缺陷报告的原始作者，有时也可以包括软件缺陷报告的修订者。当负责修复该软件缺陷的开发人员对报告有任何异议或疑义时，可以与报告人进行联系。

（4）报告日期：软件缺陷报告首次报告的日期。让开发人员知道创建软件缺陷报告的日期是很重要的，因为有可能这个软件缺陷在前面版本曾经修改过。

5.6.5 软件缺陷报告

- 一般情况下，软件缺陷报告包含下列内容。

（5）程序（或组件）的名称：程序（或组件）的名称可用来分辨被测试对象。

（6）版本号：测试可能跨越多个软件版本，提供版本信息可以方便地进行软件缺陷管理。

（7）配置：发现软件缺陷时的软件和硬件配置。如操作系统的类型、是否有浏览器载入、处理器的类型和速度、内存大小、可用内存、正在运行的其他程序等。

（8）软件缺陷的类型：软件缺陷的类型包括代码错误、设计问题、文档不匹配等。

（9）严重性：描述所报告的软件缺陷的严重性。

5.6.5 软件缺陷报告

- 一般情况下，软件缺陷报告包含下列内容。

（10）优先级：由开发人员或者管理人员确定软件缺陷的优先级，根据修复这个软件缺陷的重要性而定。

（11）关键词：关键词可在任何时候添加，以便分类查找软件缺陷报告。

（12）软件缺陷描述：对发现的问题进行详细说明，尽管描述要深入，但是简明仍是最重要的。软件缺陷描述的主要目的是说服开发人员决定去修复这个软件缺陷。

（13）重现步骤：重现步骤必须是有限的，并且描述的信息足够使读者知道正确的执行就可以重现这个软件缺陷。

（14）结果对比：在执行了重现软件缺陷步骤后，期望发生什么，实际上又发生了什么。

本章小结

- 软件测试是软件开发中的最后一个阶段。
- 软件测试是使用人工或者自动手段来运行或测试某个系统的过程，通过测试发现软件开发设计的过程中存在的问题，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。
- 软件测试的过程主要描述了软件测试需要做的工作，随着软件测试技术的进步，测试过程也会得到进一步改进。