ISTQB初级认证

第2章 软件生命周期中的测试

作者：郑文强

Email: zwqwwuy@163.com

博客：http://blog.csdn.net/Wenqiang_Zheng

声明

➔本课件的开发基于**ISTQB Foundation Level Syllabus (Version 2007)**。

➔感谢**ISTQB**和大纲作者的努力，对应的大纲可以从**www.istqb.org**下载获得。

➔本课件为个人开发，只能用于个人学习目的，不能用于任何商业活动。

➔更多**ISTQB**初级认证资料，参考：
http://blog.csdn.net/Wenqiang_Zheng/archive/2011/04/09/6311523.aspx

课程内容

1. 开发模型
2. 测试级别
3. 测试类型
4. 维护测试

软件开发模型

ISTQB考试知识点

- ★ 明白在开发生命周期中的软件开发、测试活动和工作产品之间的相互关系，并根据项目和产品的特征以及它们的背景提供相应的例子**(K2)**；
- ★ 知道必须根据项目背景和产品特征来选择软件开发的模型**(K1)**；
- ★ 理解在测试中采用不同测试级别的原因，以及在任何生命周期模型中一个良好的 测试应该具备的特征**(K1)**；

开发模型和测试

软件测试不是孤立存在的，测试活动与开发活动息息相关；

软件测试活动不仅仅包括测试执行，它应该贯穿于整个软件的生命周期之中；

不同的开发生命周期模型需要对应不同的测试阶段、测试活动和测试方法；

瀑布模型

用户需求

需求分析

概要设计

详细设计

编码和实现

测试

运行维护

软件测试作为整个软件生命周期中的一个阶段！

瀑布模型

瀑布模型的阶段

★ 用户需求：用户需求一般由用户提出，系统人员或者产品市场人员从客户或将来的系统用户中收集原始项目的信息和要求；

★ 需求分析：对用户需求进行可行性分析，并对用户需求和要求进行详细描述，并最终得到管理层和客户的批准。通过需求分析，定义了开发系统的目的和需要实现的特性和功能；

★ 概要设计：明确系统的架构、系统的模块数量，以及各个模块之间的接口、数据结构，以及可能的网络环境支持和后台数据库等。简单的说，就是将需求映射到新系统的功能和框图上，从而可以对每个子系统进行独立的开发；

瀑布模型

瀑布模型的阶段（续）

- ★ 详细设计：细化概要设计的框架，定义每个子系统的任务、行为、内部结构以及与其他子系统的接口；
- ★ 编码和实现：通过编程语言实现所有已经定义的单元，比如模块、单元和类等；
- ★ 测试：作为开发周期的一个阶段，主要是指测试执行活动；
- ★ 运行维护：软件系统或者产品发布，在用户中使用，以及根据反馈进行必要的维护活动；

瀑布模型

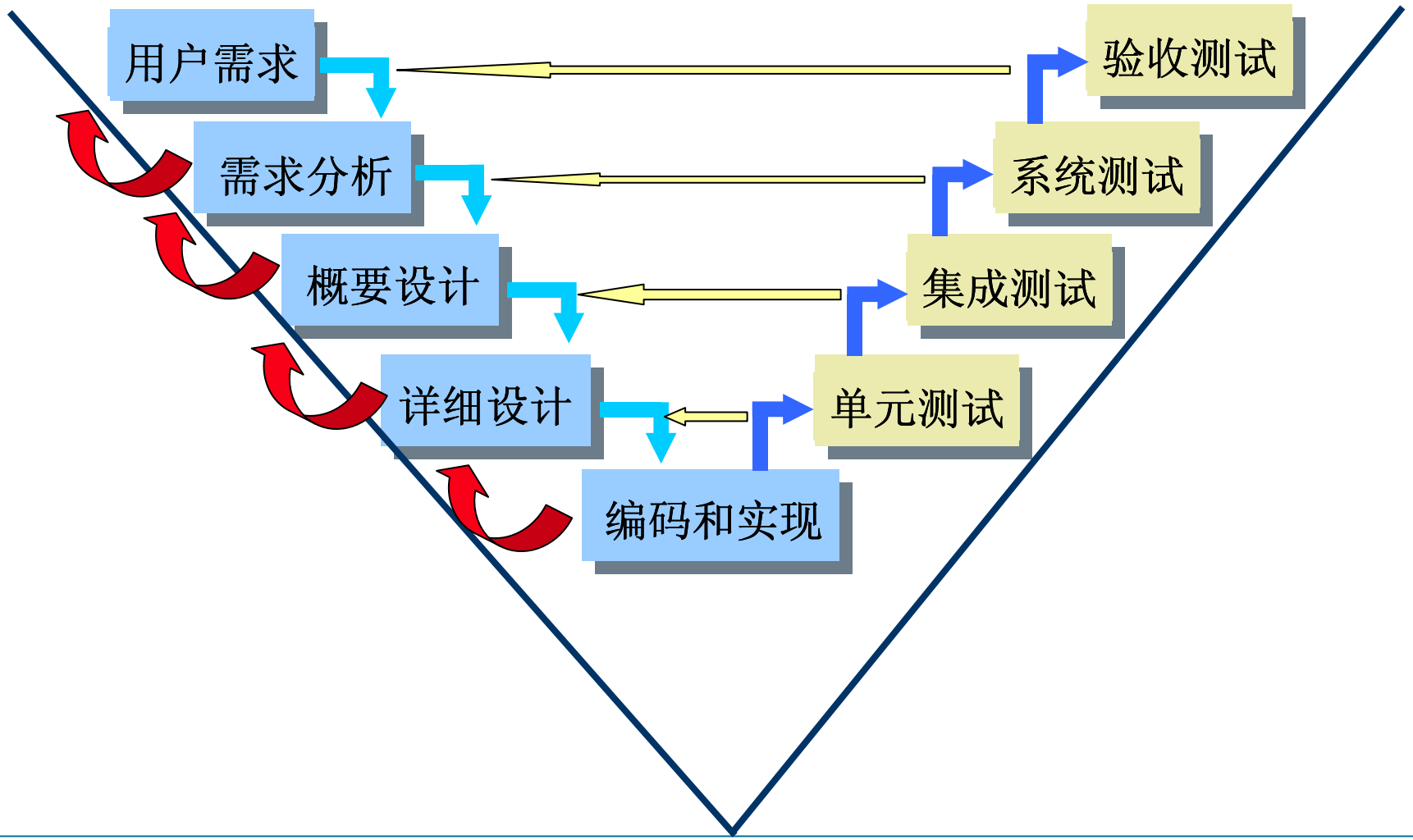
瀑布模型的特点

★ 软件测试是开发过程中的一个阶段，对产品质量进行的最后检查；

★ 在客户需求明确，以及开发过程中没有频繁的需求变更，比较适合瀑布开发模型；

★ 假如需求不明确，或者需求经常发生变更，采用瀑布模型是非常不适合的；

V模型



V模型

V模型的阶段

★除了前面瀑布模型中介绍的用户需求、需求分析、详细设计、概要设计、编码和实现几个阶段之外，还强调了不同的测试级别概念；

★单元测试：验证软件单元是否按照单元规格说明（详细设计说明）正确执行，即保证每个最小的单元能够正常运行。单元测试一般由开发人员来执行，首先设定最小的测试单元，然后通过设计相应的测试用例来验证各个单元功能的正确性；

V模型

V模型的阶段（续）

★ 集成测试：检查多个单元是否按照系统概要设计描述的方式协同工作。集成测试的主要关注点是系统能够成功编译，实现了主要的业务功能，系统各个模块之间数据能够正常通信等；

★ 系统测试：验证整个系统是否满足需求规格说明；

★ 验收测试：从用户的角度检查系统是否满足合同中定义的需求或者用户需求；

V模型

V模型的特点

- ★V模型体现的主要思想是开发和测试同等重要，左侧代表的是开发活动，而右侧代表的是测试活动；
- ★V模型针对每个开发阶段，都有一个测试级别与之想对应；
- ★测试依旧是开发生命周期中的阶段，与瀑布模型不同的是，有多个测试级别与开发阶段对应；
- ★V模型适用于需求明确和需求变更不频繁的情形；

V模型

V模型的验证和确认

验证Verification：通过检查和提供客观证据来证实指定的需求是否满足。也就是说，输入与输出之间的比较；

确认Validation：通过检查和提供客观证据来证实特定目的的功能或应用是否已经实现。在确认时，应考虑使用和应用的条件范围要远远大于输入时确定的范围；

V模型

V模型的验证和确认

验证：是否正确地构建了系统？

VS

确认：是否构建了正确的系统？

实际上每个测试都包括确认测试和验证测试这两个方面。而确认测试的内容随着测试级别的不断提高而增加；

非线性模型

瀑布模型和**V**模型，都是属于线性模型的范畴，它们的重要特点就是线性的，即前置条件是软件系统具有比较明确的需求，且没有频繁的需求变更；

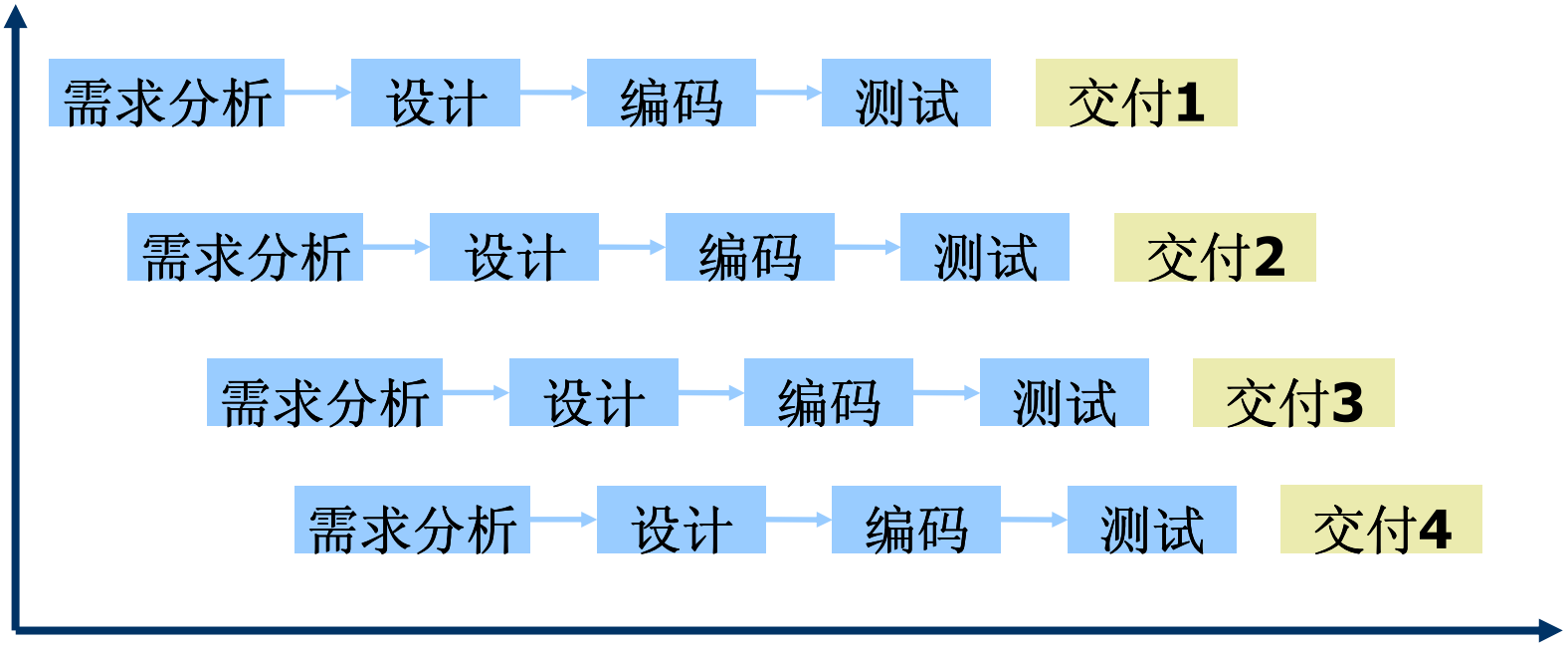
但是，现实生活中，这个条件是很难满足的！

非线性模型

- ★需求是可变的：某些应用软件的需求与外部环境、公司经营策略或经营内容等密切相关，这些都是经常调整和变化的，因此需求也是变化的；
- ★需求是模糊的：许多用户对他们的需求最初只是模糊的概念，想要求一个对需求只有初步设想的人准确无误的说出全部需求，显然是不切实际的；
- ★用户和开发者难于沟通：大多数用户和领域专家不熟悉计算机和软件技术，而软件开发人员也往往不熟悉用户的专业领域，开发人员和用户之间很难做到完全沟通和相互理解，在需求分析阶段做出的用户需求常常是不完整、不正确的；

非线性模型

因此，我们引入了非线性开发模型！增量模型是重要的非线性模型之一！

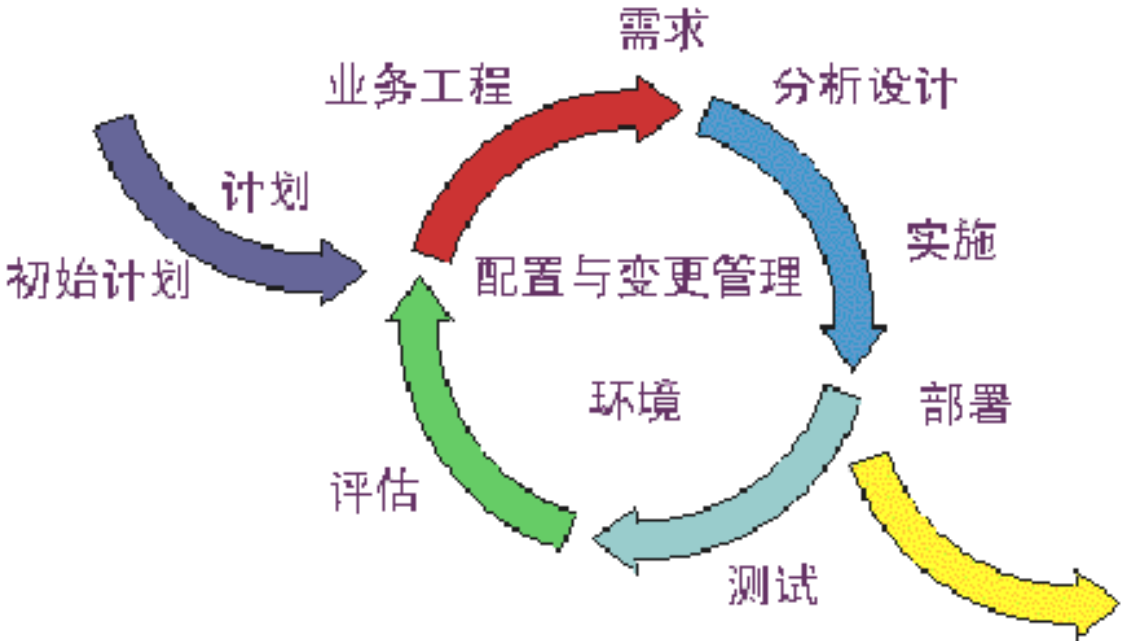


增量模型

增量模型的特点

- ★ 增量模型在每个阶段交付满足客户需求的一个子集的可运行产品。因此可以较好的适应变化，以及控制风险；
- ★ 增量的一个缺点是后面并入的构件不能破坏已构造好的系统部分，这需要软件具备开放式的体系结构；
- ★ 在开发过程中，需求的变化是不可避免的。增量模型的灵活性可以使其适应这种变化的能力大大优于线性模型，但也很容易退化为边做边改模型，从而是软件过程的控制失去整体性；

迭代模型



迭代模型

迭代模型的特点

- ★迭代模型包括了一系列的迭代，每一个迭代都包括了一些或者很多的开发活动（需求、分析、设计、实现等等）；
- ★每个后续的迭代都建立在前一个迭代的基础上以使系统得到发展和细化，直到最终产品被完成；
- ★迭代模型中集成不是在项目的尾声进行的"大动作"，每一次迭代都以集成构建系统各部分结束，这样不断的积累将使日后的返工最小化；

开发模型的选择

- ★在前期需求明确的情况下尽量采用瀑布模型或改进型的瀑布模型；
- ★在用户无信息系统使用经验，需求分析人员技能不足情况下一定要借助原型；
- ★在不确定性因素很多，或者需求不稳定的情况下，无法有效的进行计划的情况下，尽量采用增量迭代和螺旋模型；
- ★资金和成本无法一次到位情况下可以采用增量模型，软件产品分多个版本进行发布；

开发模型的选择（续）

- ★对于完全多个独立功能开发可以在需求阶段就分功能并行，但每个功能内都应该遵循瀑布模型；
- ★对于全新系统的开发必须在总体设计完成后再开始增量或并行；
- ★对于编码人员经验较少情况下建议不要采用敏捷或迭代等生命周期模型；
- ★增量、迭代和原型可以综合使用，但每一次增量或迭代都必须有明确的交付和出口准则；

什么是好的测试

好的测试应该具备

- ★每个开发活动都有相对应的测试活动；
- ★每个测试级别都有其特有的测试目标；
- ★对于每个测试级别，需要在相应的开发活动过程中进行相应的测试分析和设计；
- ★在开发生命周期中，测试人员在文档初稿阶段就应该参与文档的评审；

课程内容

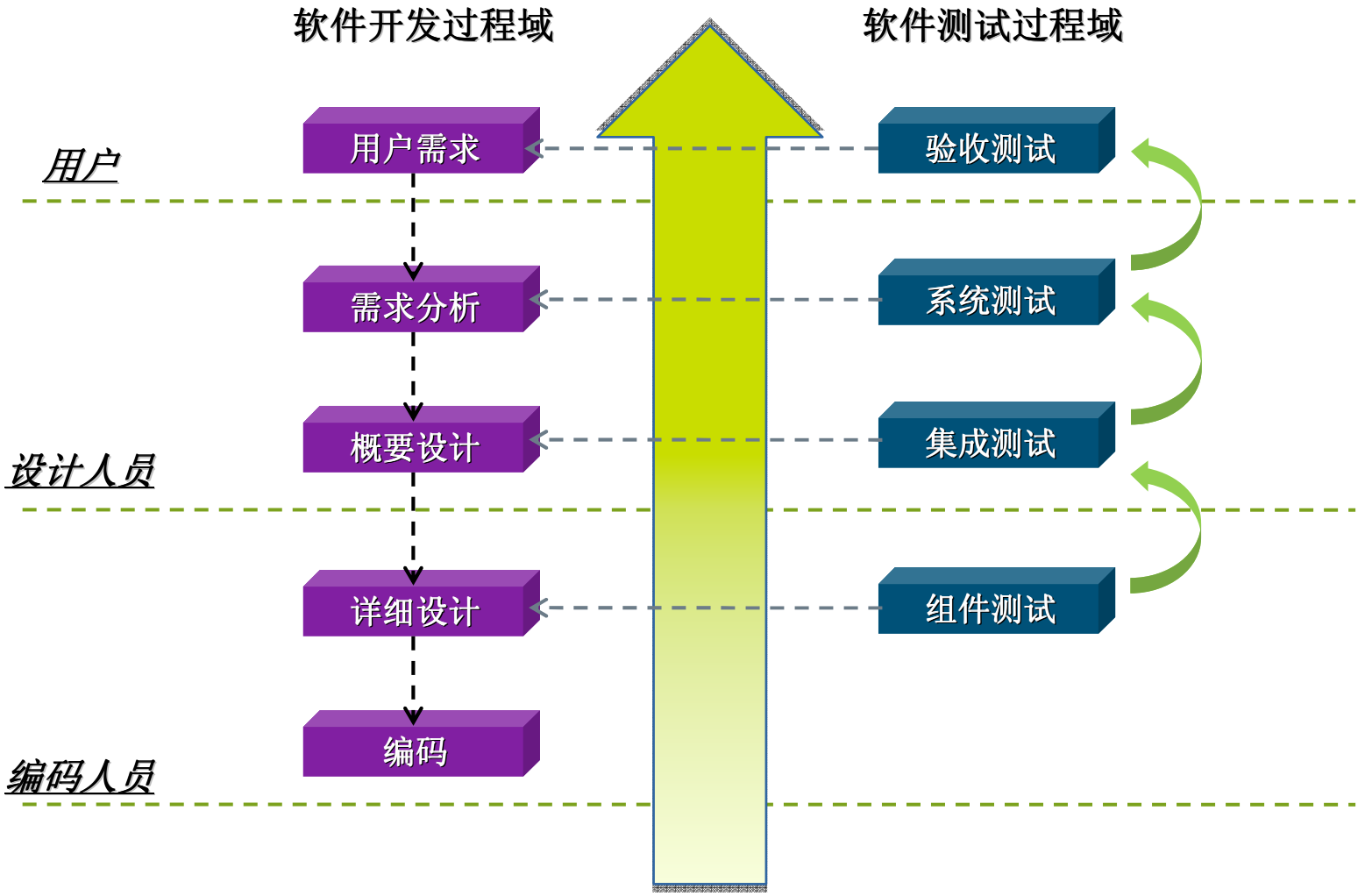
1. 开发模型
2. 测试级别
3. 测试类型
4. 维护测试

测试级别

ISTQB考试知识点

★ 比较不同测试级别之间的区别：测试的主要目的、典型的测试对象、典型的测试目标 （功能性的或结构性的）、相关的工作产品、测试的人员、识别缺陷和失效的种类（**K2**）；

测试级别



测试级别

测试活动贯穿于整个软件生命周期

- ★ 单元测试
- ★ 集成测试
- ★ 系统测试
- ★ 验收测试

测试级别

针对不同的测试级别，我们应该明确

- ★不同的测试的对象；
- ★每个测试级别的测试目的；
- ★测试用例参考的工作产品：测试依据；
- ★发现的典型缺陷和失效；
- ★测试工具的需求和支持；
- ★不同的测试技术和方法；
- ★……

单元测试

基本含义

- ★ 单元测试的对象可以是模块、类、函数和对象等，不同的软件语言来决定；
- ★ 单元测试的主要目的是验证单元是否满足了详细设计规格说明，发现需求和设计中的错误；
- ★ 单元测试设计的主要输入是详细设计规格说明、软件设计和数据模型等；
- ★ 单元测试主要采用白盒测试技术，黑盒测试技术作为单元测试的辅助；

单元测试

基本含义

★ 单元测试应该覆盖功能需求和非功能需求；

★ 单元测试经常会使用测试驱动的方法（测试驱动开发）；

开发驱动测试

VS

测试驱动开发

单元测试

测试环境

- ★ 单元测试处理的对象直接来自开发人员，通常由开发人员来开展单元测试；
- ★ 单元测试可能并不能形成完成的系统，因此需要驱动模块和桩模块的支持：
  - ✦ 桩模块：用以模拟被测模块工作过程中所调用的模块，他们一般只进行很少的数据处理，例如打印入口和返回；
  - ✦ 驱动模块：用以模拟被测模块的上级模块，它接受测试数据，把相关的数据传送给被测模块，启动被测模块，并打印相应的结果；
- ★ 驱动模块和桩模块是测试使用的软件，而不是软件产品的组成部分，但它需要一定的开发费用；

单元测试

单元测试关注点

- ★ 单元模块接口参数；
  - ✦ 实际参数和形式参数的个数是否相同；
  - ✦ 实际参数和形式参数的属性是否匹配；
  - ✦ 调用函数的参数顺序、个数和属性是否匹配；
- ★ 单元模块局部数据结构；
  - ✦ 不合适或者不相容的类型说明；
  - ✦ 变量没有初始化；
  - ✦ 不正确的变量名；

单元测试

单元测试关注点（续）

- ★ 单元模块的独立路径测试；
  - ✦ 误解或者用错了算符优先级；
  - ✦ 混和类型运算；
- ★ 与控制流相关的测试；
  - ✦ 错误的修改了循环变量；
  - ✦ 循环中止条件不可能出现；
- ★ 与异常处理相关的测试；
  - ✦ 输出的错误信息难以理解；
  - ✦ 错误的信息和实际的错误不符；

集成测试

基本含义

- ★集成测试，又叫组装测试、联合测试等；
- ★集成测试是对组件之间的接口进行测试，以及和系统其他部分的相互作用；
- ★最简单的形式是两个已经测试的单元组合成一个组件，来测试它们之间的接口和数据交换；
- ★集成测试的主要工作：把单元测试通过的各个模块逐步集成在一起，来测试数据是否能够正确传递和调用，以及各个模块是否能正确的协同工作；
- ★集成测试可以应用在不同的测试级别，比如单元集成测试、系统集成测试等；

集成测试

集成测试的关注点

- ★ 单元模块是否传输了错误的数据，或者没有传输数据；
- ★ 接受数据的单元不能操作或者崩溃，比如单元功能缺陷、接口格式不兼容、协议不兼容等；
- ★ 单元之间通讯正常，但是使用不同的方法来解析收到的数据，比如规格说明矛盾、理解错误等；
- ★ 数据能正常传输，但是传输时间错误，比如时序问题，或者传输的时间间隔太短，比如吞吐量、负荷、容量等问题；

集成测试策略

自底向上集成测试步骤

★ 明确被测模块并进行先后顺序分层；

★ 按照时间线序关系，将不同单元进行集成；

★ 将不同的模块集成为子系统，或者分系统；

★ 将子系统集成为系统；

自底向上集成测试特点

★ 不需要桩；

★ 需要构造不同的驱动模块；

集成测试策略

自顶向下集成测试步骤

- ★ 以主控模块作为测试驱动，把对主控模块进行单元测试时引入的所有桩模块用实际模块替代；
- ★ 依据所选的集成策略（深度优先或广度优先），每次只替代一个桩模块；
- ★ 每集成一个模块立即测试一遍；
- ★ 只有每组测试完成后，才着手替换下一个桩模块；
- ★ 为避免引入新错误，须不断地进行回归测试（即全部或部分地重复已做过的测试）；

集成测试策略

自顶向下集成测试特点

- ★优点：不需要测试驱动器，或者只需要简单的测试驱动，这是因为经过测试的较高级别单元组成了测试环境的主要部分；
- ★缺点：还没有集成的较低级别的单元必须用桩代替，成本很高；

集成测试策略

核心系统优先集成测试步骤

- ★对核心系统中的模块进行单独的、充分的测试；
- ★核心系统的所有模块一次性集合到被测系统中，在规模相对较大的情况下，也可以按照自底向上的步骤，集成核心系统的各组成模块。；
- ★按照各外围软件部件的重要程度以及模块间的相互制约关系，拟定外围软件部件集成到核心系统中的顺序方案；
- ★完成外围软件部件内部的集成测试；
- ★按顺序不断加入外围软件部件到核心系统；

集成测试策略

随意集成测试步骤

★ 按照单元的完成时间进行集成；

随意集成测试特点

★ 优点：节省时间，因为每个单元可以最快的集成到环境中来；

★ 缺点：桩和测试驱动器都需要；

集成测试策略

大爆炸集成策略：避免

- 本来可以用作测试的时间浪费在等待大爆炸集成上了。由于总是缺乏用于测试的时间，所以不应该浪费可以用于测试的任何时间。
- 所有的失效将会同时发生。有时集成后的系统根本无法运行起来。此外，定位和修正缺陷会很困难，并且消耗很多时间。

系统测试

基本含义

- ★ 系统测试是将已经集成好的软件系统，作为计算机系统的一部分，与计算机硬件、某些支持软件、数据和人员等系统元素结合起来，在实际运行环境下对计算机系统进行一系列严格有效的测试；
- ★ 系统测试关注的是项目或产品范围中定义的整个系统或产品的行为；
- ★ 在系统测试中，测试环境应该尽量和最终使用的目标或产品使用的环境相一致，从而减少和环境相关的失效；

系统测试

测试目标

★ 系统测试的目标是确认整个系统是否满足了规格说明中的功能和非功能需求，以及满足的程度；

★ 系统测试应该发现由于需求不正确、不完整或实现和需求不一致而引起的失效，并识别没有文档化或被忘记的需求；

★ 常见的系统测试包括压力测试、容量测试、性能测试、安全测试、容错测试等；

系统测试

为什么系统测试

- 在较低的测试级别，测试主要是针对技术规格说明的，即从软件开发者的技术观点角度加以考虑。而系统测试从客户或用户① 的观点来考虑整个系统。测试人员确认系统是否完全正确地满足了需求。
- 许多功能和系统属性是从系统的所有组件相互协调的过程中得到的，因而只能在整个系统级别才能看到，也只能在这个时候才能够进行观察并测试。

验收测试

基本含义

★ 验收测试通常是由使用系统的用户来进行，同时系统的其他利益相关者也可能参与其中；

★ 验收测试目的是通过验收测试，对系统功能、系统特定部分或特定的系统非功能特征进行测试；

★ 发现缺陷不是验收测试的主要目标，验收测试也可以用来评估系统是否可以在市场部署、用户使用系统的准备情况等；

验收测试

验收测试类型

- ★合同验收测试
- ★规范验收测试
- ★**Alpha**和**Beta**测试
- ★用户验收测试
- ★运行（验收）测试
- ★**……**

验收测试类型

验收类型	执行者	测试目标	测试阶段
用户验收测试	关键用户	验证软件系统符合商业用户商业需要	软件开发的任何阶段
运行（验收）测试	系统管理员	验证系统符合IT管理需要 • 系统备份/恢复测试 • 灾难恢复测试 • 用户管理测试 • 维护任务测试 • 安全漏洞阶段性检查	软件开发的任何阶段
合同和法规性验收测试	用户、系统管理员	验收软件符合合同中规定的验收准则	软件开发的任何阶段
Alpha和Beta测试	潜在用户	在软件投入市场前，获取潜在用户关于软件的反馈信息	软件正式投入市场前

课程内容

1. 开发模型
2. 测试级别
3. 测试类型
4. 维护测试

测试类型

ISTQB考试知识点

- ★ 举例比较四种不同的测试类型（功能测试、非功能测试、结构测试和与变更 相关的测试）**(K2)**;
- ★ 明白功能测试和结构测试可以应用在任何不同的测试级别**(K1)**;
- ★ 根据非功能需求来识别和描述非功能测试的不同类型**(K2)**;
- ★ 根据对软件系统结构或构架的分析来识别和描述测试的类型**(K2)**;
- ★ 描述确认测试和回归测试的目的**(K2)**;

ISTQB测试分类

- ★ 功能测试
- ★ 非功能测试
- ★ 结构测试
- ★ 变更相关的测试

功能测试

基本含义

- ★ 功能测试指的是软件系统"做什么";
- ★ 功能测试的测试依据包括：需求规格说明、用例、功能规格说明，它们描述了系统必须完成的功能；
- ★ 功能测试主要考虑的是系统的外部表现，即一般采用的是黑盒测试的技术；
- ★ 功能测试可以应用在各个测试级别；

功能测试

功能测试包括什么

- ★合适性（**Suitability**）
- ★准确性（**Accurateness**）
- ★互操作性（**Interoperability**）
- ★安全性（**Security**）

来自**ISO/IEC 9126**质量属性模型

非功能性测试

基本含义

★ 非功能性指的是系统工作的"怎么样"，比如系统有多容易使用等；

★ 非功能测试不是功能的测试，而是对功能行为的测试，或作为整体系统能力的测试；

★ 非功能测试可以应用在任何测试级别上；

非功能测试

非功能测试包括

- ★ 可靠性（**Reliability**）
- ★ 易用性（**Usability**）
- ★ 可维护性（**Maintainability**）
- ★ 可移植性（**Portability**）

来自**ISO/IEC 9126**质量属性模型

非功能测试

非功能测试例子

- 负载测试（load test）：增加系统负载来测量系统的行为（如并发用户数、事务数）。
- 性能测试（performance test）：测量特定用例的处理速度和响应时间，通常依赖于不断增加的负载。
- 容量测试（volume test）：在大容量数据下系统行为的观察（如处理非常大的文件）。
- 压力测试（stress test）：观察超负荷下系统行为。
- 安全性测试（testing of security）：针对未授权的访问，拒绝访问攻击等。

非功能测试

非功能测试例子

- 稳定性（stability）：或可靠性测试：在长时间运行中测试（如失效的平均时间或指定用户配置的失效率）。
- 健壮性测试（robustness test）：测量在系统对于错误操作、错误编程、硬件故障的响应，以及检查系统异常处理和恢复的能力。
- 兼容性和数据转换测试：检查给定系统的兼容性，导入/导出数据等。
- 系统不同配置的测试：例如：操作系统的不同版本，用户接口的语言，硬件平台等等（比对测试（back-to-back testing））。
- 可用性测试（usability test）：检查客户学习系统的容易性、操作的容易性和效率、系统输出的可理解性等，经常与特定用户组的需求相关（[ISO 9241]，[ISO 9126]）。

结构测试

基本含义

- ★ 结构测试，或者白盒测试，使用测试对象的内部代码结构和架构信息（语句或判断、递归调用、菜单结构），也可能使用软件的抽象模型（例如：过程流模型或状态转换模型）等作为输入；
- ★ 结构测试可以在任何测试级别上进行；
- ★ 结构测试技术通常通过评估结构类型测试的覆盖性，来测量测试的完整性；
- ★ 结构测试通常应用在低级别的测试上，比如单元测试和集成测试，但它同样使用其他级别的测试；

结构测试

覆盖类型

- ★语句覆盖
- ★判定覆盖
- ★条件组合覆盖
- ★判定/条件覆盖
- ★路径覆盖
- ★……

具体的覆盖类型可以参考白盒测试设计章节

变更相关的测试

基本含义

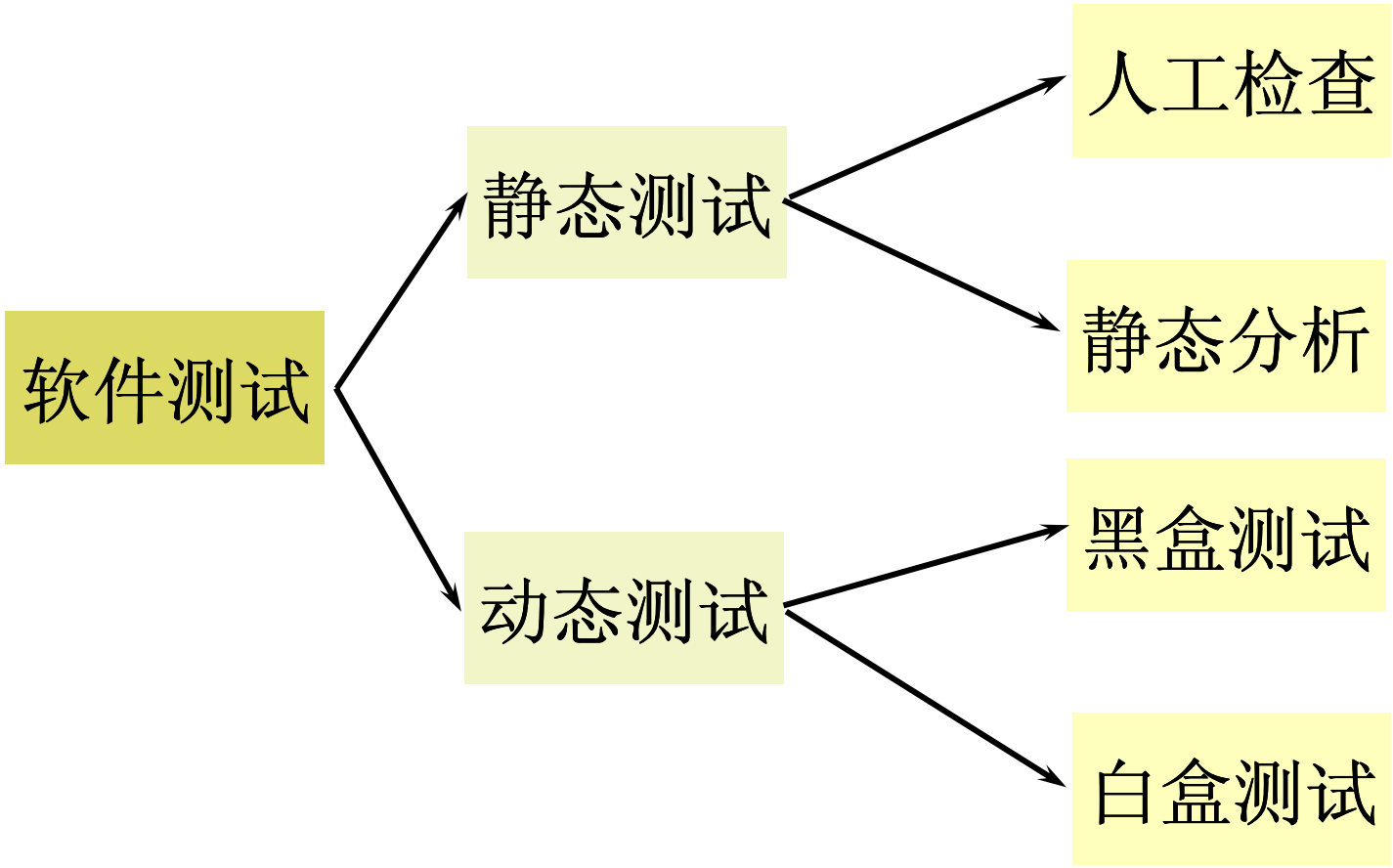
- ★ 当已有的系统发生变化、缺陷被修正或者增加了新功能，变化的部分必须进行重新测试，我们通称为回归测试；
- ★ 目的**1**：通过重新进行测试以确定原来的缺陷已经成功的修改、或者新增的功能已经正确实现；
- ★ 目的**2**：通过测试，确认系统的变更比如新增功能或者缺陷修改，没有影响原来的功能和系统行为；
- ★ 回归测试可以应用到任何级别的测试中；

变更相关的测试

回归测试策略

- ★ 重新运行所有发现缺陷的测试用例，判断新的软件版本是否已经修正了这些缺陷（缺陷再测试、确认测试）；
- ★ 对所有因为修改缺陷而进行修改的程序，以及相关的测试用例，重新执行一遍；
- ★ 对所有因为新增加的功能而受到影响的代码，和相关的测试用例，重新进行执行；
- ★ 对整个系统重新进行一次完整的测试（完全回归测试）；

按形式分类



按测试对象是否运行来进行分类！

静态测试

基本定义

- ★静态测试通过对模块的源代码进行研读，查找错误或收集一些度量数据，并不需要对代码进行编译和仿真运行；
- ★主要目的是从已有的规格说明、已定义的标准或甚至是项目计划中发现缺陷和偏差；
- ★包括人工检查（评审）和自动化检查（静态分析），是一个经常被低估的测试方法，或者在测试过程中经常被忽略的方法；

静态测试

基本类型

- ★ 人工检查：通过人工的方式，对软件产品的设计规格说明的审查，对程序代码的阅读、审查等；
  - ✦ 走查（**Walkthrough)**：
  - ✦ 审查**(Inspection)**：
  - ✦ 评审**(Review)**：
- ★ 静态分析：静态分析往往需要借助测试工具（比如编译器就是一种静态分析工具）来自动检测；

动态测试

基本定义

- ★ 动态测试是通过观察软件运行时的动作，来提供执行跟踪、时间分析，以及测试覆盖度方面的信息；
- ★ 动态测试通过真正运行程序发现错误。通过有效的测试用例，对应的输入/输出关系来分析被测软件的运行情况；

动态测试

基本类型

- ★ 白盒测试
- ★ 黑盒测试
- ★ 基于经验的测试
- ★ ……

课程内容

1. 开发模型
2. 测试级别
3. 测试类型
4. 维护测试

维护测试

ISTQB考试知识点

- ★比较维护测试（一个现存系统的测试）与一个新的应用软件的测试在测试类型、测试的触发和测试规模等方面的区别（**K2**）；
- ★识别维护测试的原因（由于修改、移植或退役等因素）（**K1**）；
- ★描述回归测试和变更的影响度分析在软件维护中的作用（**K2**）；

维护测试

基本含义

- ★ 软件系统在运行过程中，经常需要对软件系统和它运行的环境进行修正、改变或扩展；
- ★ 维护测试是在一个运行的系统上进行，且一旦对软件或系统进行修改、移植或系统被淘汰时，就需要进行维护测试；

维护测试

基本类型

- ★ 修改可以是计划中的功能增强（例如：根据版本发布的计划）；
- ★ 也可以是缺陷修改纠正和应急变更；
- ★ 甚至是环境的变化，比如计划中的操作系统或数据库升级，或由于新发现或暴露的操作系统漏洞而打的补丁等；

维护测试

为什么维护测试

(1) 系统在未预料的和没有计划过的新的运行环境中运行。
(2) 客户表达了新的愿望。
(3) 需要处理不可遇见的很少发生的情况。
(4) 出现很少发生或只在运行时间很长后才发生的程序崩溃。

维护测试

为什么回归测试

★ 通过重新进行测试以确定原来的缺陷已经成功的修改、或者新增的功能已经正确实现；

★ 通过测试，确认系统的变更比如新增功能或者缺陷修改，没有影响原来的功能和系统行为；

我们如何来确定测试的内容和优先级呢？

维护测试

如何回归测试

★ 维护测试的范围取决于变更的风险、现有系统的规模和变更的大小；

★ 确定变更如何影响现有系统的过程，称之为影响分析，它有助于决定实施回归测试的广度和深度；

维护测试

回归测试瀑布模型

- 软件变更分析
- 变更影响分析
- 定义回归测试策略
- 定义回归测试套件
- 执行回归测试套件
- 报告回归测试结果

维护测试

步骤描述

★ 软件变更分析：分析和了解各种不同的软件变更，包括软件版本新功能的增加、需求文档的变更、系统设计的变更、系统实现的变更、软件使用平台和操作系统的变更，以及缺陷的修改等；

★ 软件变更影响分析：分析和了解由于软件变更可能导致对软件系统的影响，比如由于需求文档的变更，会导致后续的设计文档、编码和测试文档的变更，以及测试活动的变更；

维护测试

步骤描述

- ★ 定义回归测试策略：定义回归测试的方法、策略和准则。回归测试策略可以作为指南，帮助测试工程师开展回归测试活动。回归测试策略一般来说是指如何定义回归测试的范围、回归测试的覆盖率准则要求，以及回归测试执行的顺序等等；
- ★ 定义回归测试套件：定义如何选择和重用测试用例来形成一个回归测试套件；
- ★ 执行回归测试套件：执行套件中的测试用例；
- ★ 报告回归测试结果：报告和分析回归测试结果；

维护测试

版本开发和维护测试

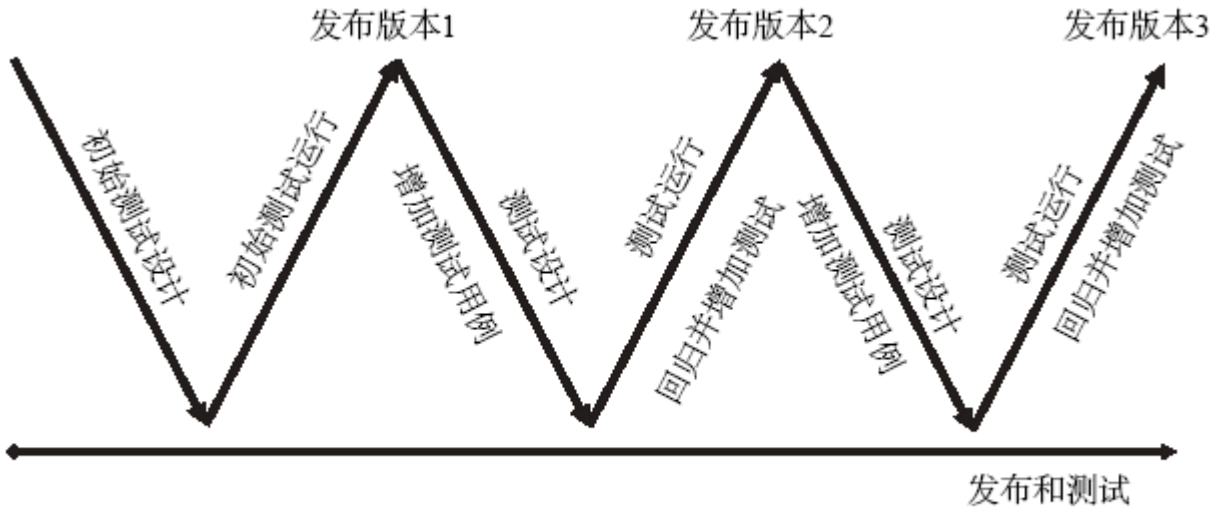


图3-5　增量开发中的测试

答　疑