

Unit Testing

[书上第3.4.5.8章]



单元测试的定义

定义:

单元测试是对软件基本组成单元进行的测试。

时机:

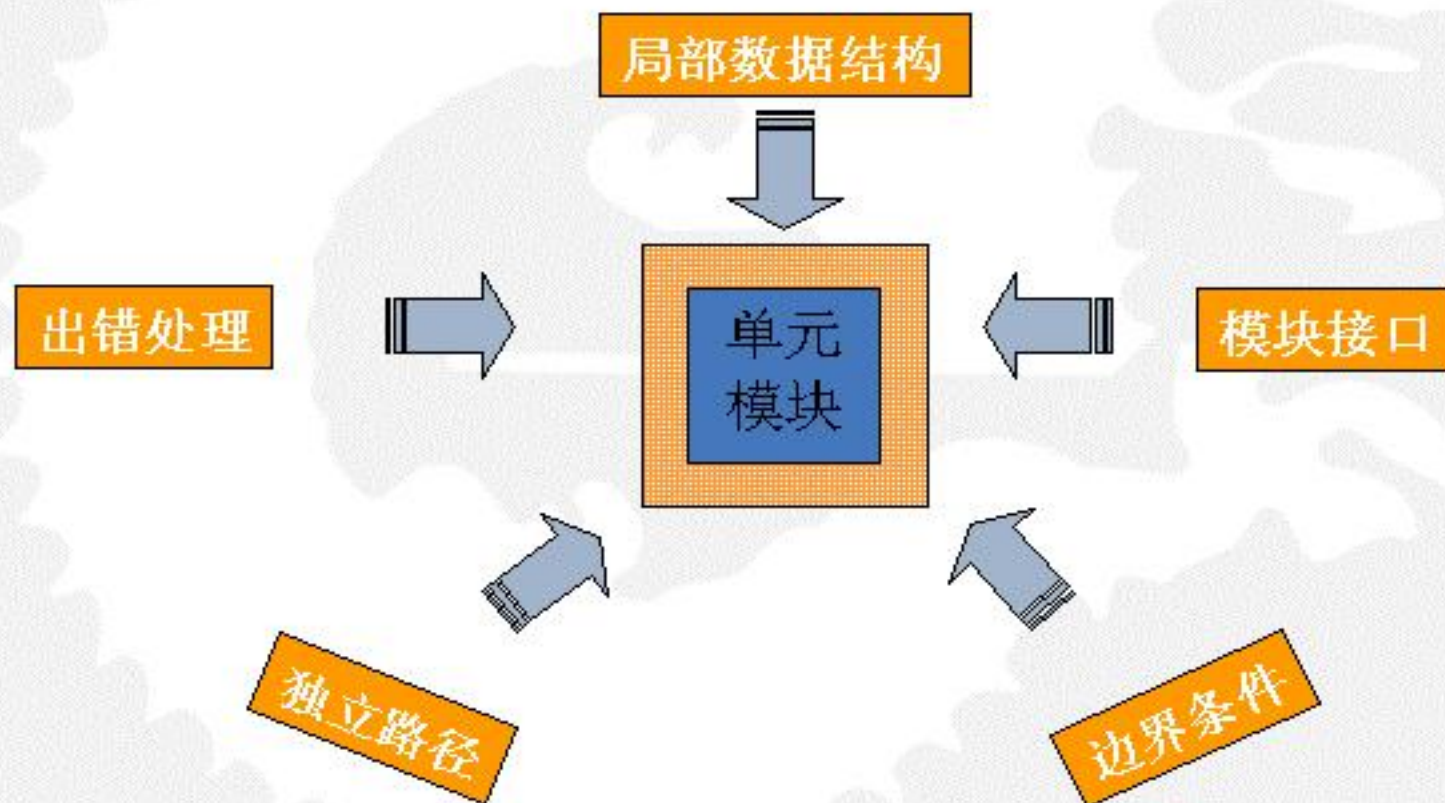
一般在代码完成后由开发人员完成, QA人员辅助.

概念:

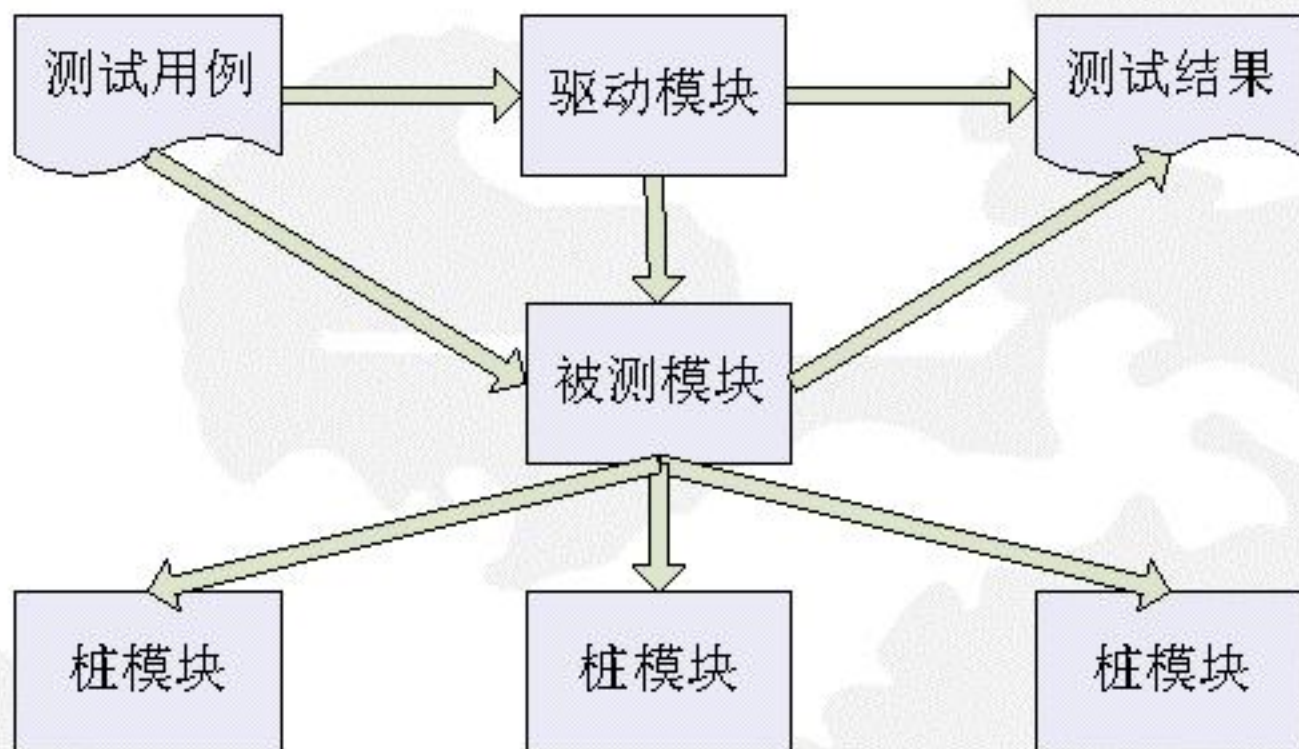
单元: 有明确的功能、性能定义、接口定义的软件设计最小单位——模块。

不是函数或类中的方法! 而是模块或者类

单元测试的内容



单元测试示意图



Strategies for Unit Testing

▲ *Black box*

- ▲ *use only specification of program*
- ▲ *test implementation against its specification*

▲ *White box*

- ▲ *use structure or other properties of a program to generate tests*



Static white-box testing

- ▶ *Static white-box testing is the process of carefully and methodically reviewing the software design, architecture, or code for bugs without executing it.*
- ▶ *Unfortunately, static white-box testing is rarely done in practice (unlike dynamic black-box testing).*



Essential elements of a formal code review

- ▶ *Identify problems:*
 - ▶ *Find problems with the software such as missing items, mistakes, etc.*
- ▶ *Follow rules:*
 - ▶ *Amount of code to be reviewed, how much time will be spent, etc.*
- ▶ *Prepare:*
 - ▶ *Each participant should prepare in order to contribute to the review.*
- ▶ *Write a report:*
 - ▶ *Summarize the results of the review, make report available to the development team.*



Informal 非正式code inspections

▲ *Peer reviews* 同级审查:

- ▲ *An informal small group of programmers and/or testers act as reviewers.*
- ▲ *Participants should follow the 4 essential elements even though the review is informal.*

▲ *Walkthroughs*: 走查

- ▲ *A more formal process in which the author of the code formally presents the code to a small group of programmers and/or testers.*
- ▲ *The author reads the code line by line explaining what it does, reviewers listen and ask questions.*
- ▲ *Participants should follow the 4 essential elements.*



Formal code inspections

- ▶ Code presenter *is not the author of the code.*
- ▶ *The other participants are the inspectors.*
- ▶ *There is a moderator to assure that the rules are followed and the meeting runs smoothly.*
- ▶ *After the inspection a report is composed. The programmer then makes changes and a re-inspection occurs, if necessary.*
- ▶ *Formal code inspections are effective at finding bugs in code and designs and are gaining in popularity*



走查 (*Walk Through*)

定义：采用讲解、讨论和模拟运行的方式进行的查找错误的活动。

注意：

- 引导小组成员在走查前通读设计和编码。
- 限时，避免跑题。
- 发现问题适当记录，避免现场修改。
- 检查要点是代码是否符合标准和规范，是否有逻辑错误。



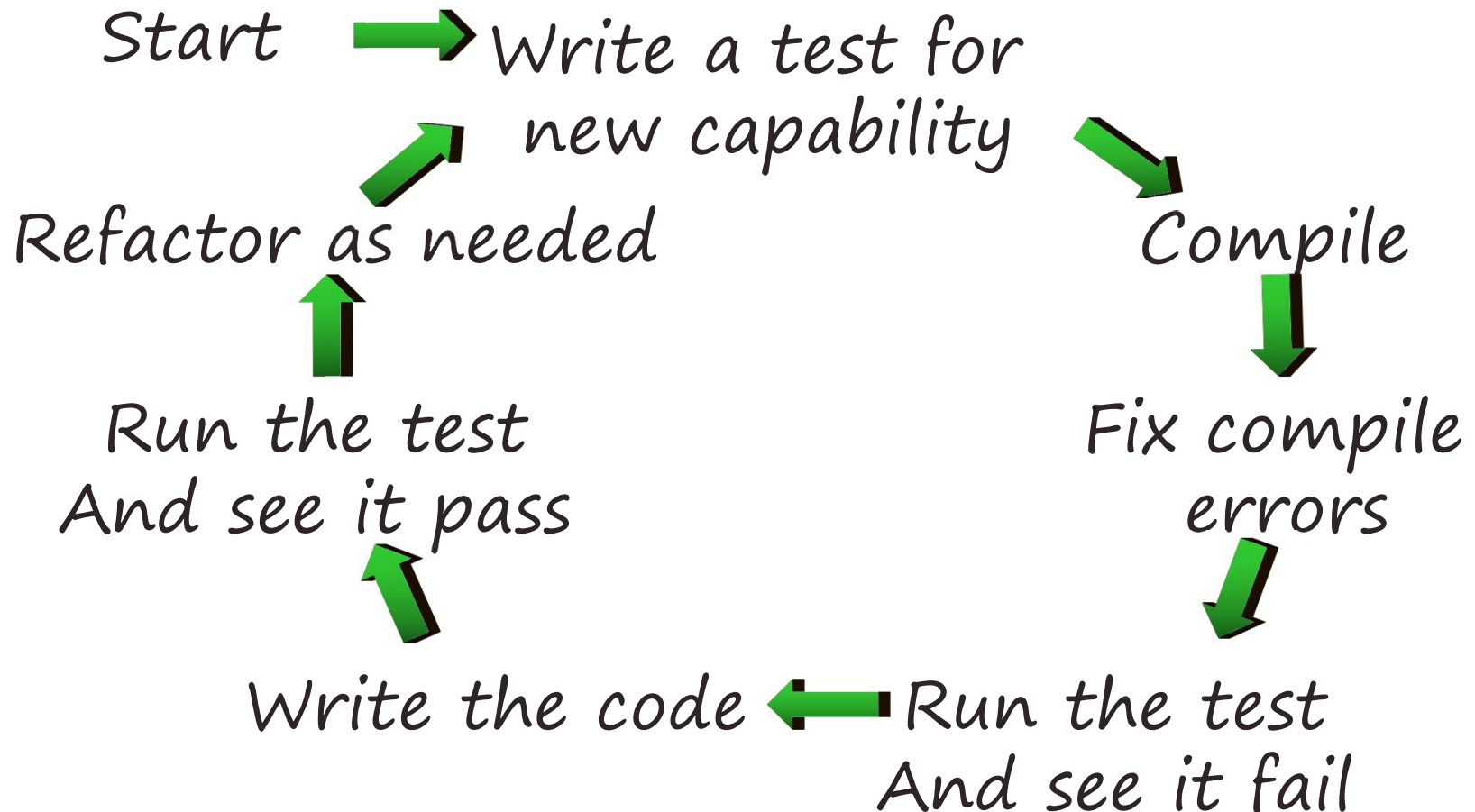
审查 (Review)

定义：采用讲解、提问方式进行，一般有正式的计划、流程和结果。主要方法采用缺陷检查表。

注意：

- 以会议形式，制定会议目标、流程和规则，结束后要编写报告。
- 按缺陷检查表逐项检查。
- 发现问题适当记录，避免现场修改。
- 发现重大缺陷，改正后会议需要重开。
- 检查要点是缺陷检查表，所以该表要根据项目不同不断积累完善。

TDD - Test-Driven Development 测试驱动开发



xUnit:JUnit;CppUnit

- ▲ *xUnit* 框架在 1998 年作为 *eXtreme* 编程的核心概念引入。它提出了一个有效的机制，有助于开发人员将结构化、有效且自动的单元测试添加常规开发活动中。从那以后，该框架演化为针对自动化单元测试框架的实际标准。



xUnit框架

xUnit 框架概念		描述
测试	TestMethod	简单说，这些是您的测试。测试预期结果的逻辑，并报告未取得结果（如果有）。请将它看作您的“方法”。
测试装置	TestClass	针对大量测试的一个逻辑分组。请将它看作您的“类”。
测试套件	测试列表 **	针对大量测试装置的一个逻辑分组。请将它看作您的“类库”。 注不需要一个属性。
测试运行器		

JUnit

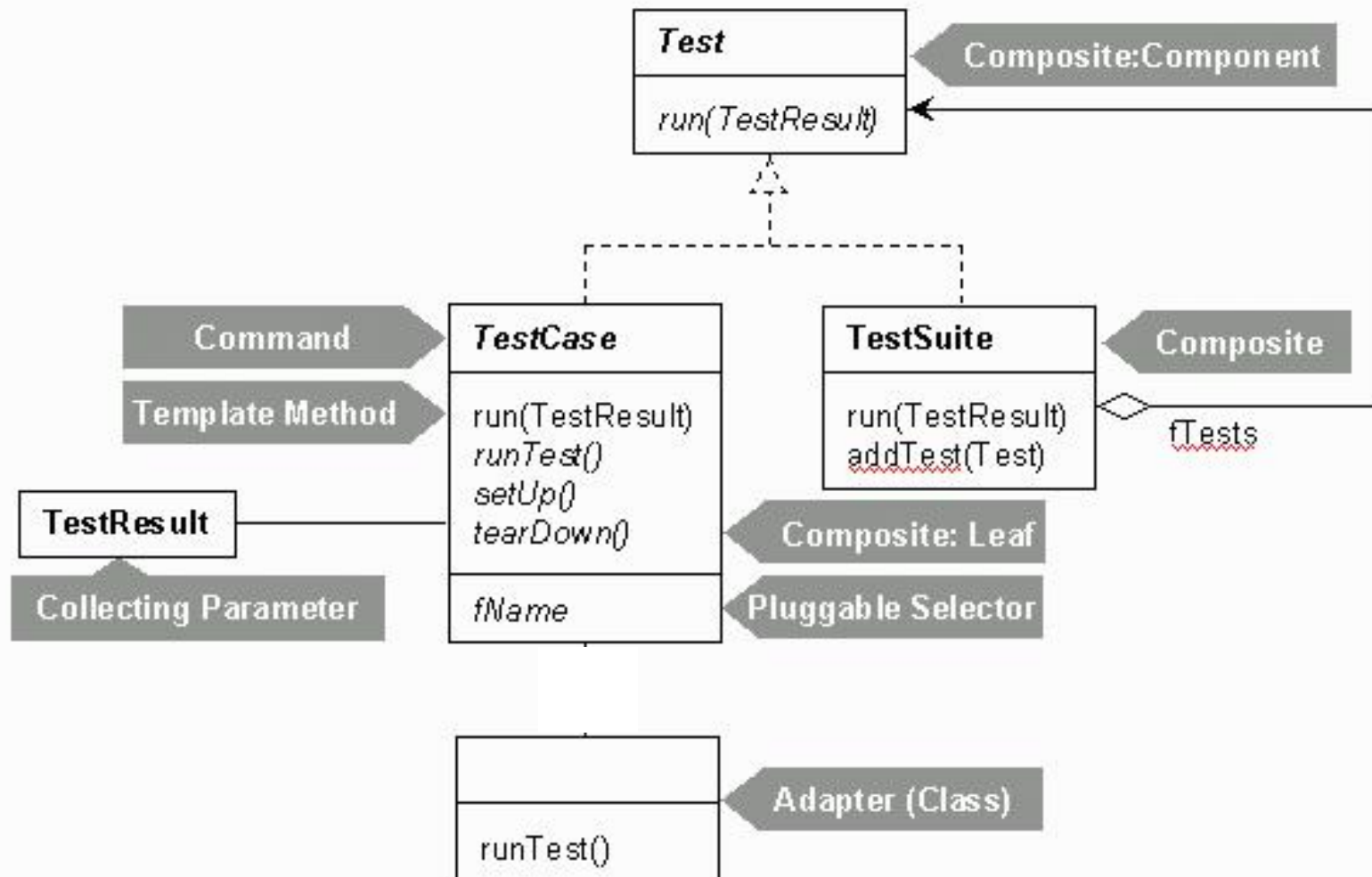
- ▲ JUnit是一个开发源代码的Java测试框架，用于编写和运行可重复的测试。是用于单元测试框架体系xUnit的一个实例（用于java语言）



JUnit

- ▲ 需要说明的是JUnit一般是用来进行单元测试的，因此需要了解被测试代码的内部结构（即所谓的白盒测试），另外JUnit是在xp编程和重构（refactor）中被极力推荐使用的工具，因为在实现自动单元测试的情况下可以大大的提高开发的效率
- ▲ <http://www.junit.org/>
- ▲ *JUnit4 最新 Last Published:2018-02-07 /Version: 4.12*
- ▲ *JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage*
 - ▲ *JUnit Platform是在JVM上启动测试框架的基础。*
 - ▲ *JUnit Jupiter是JUnit5扩展的新的编程模型和扩展模型，用来编写测试用例。Jupiter子项目为在平台上运行Jupiter的测试提供了一个TestEngine（测试引擎）。*
 - ▲ *JUnit Vintage提供了一个在平台上运行JUnit 3和JUnit 4的TestEngine。*

JUnit 框架



JUnit

- ▶ *JUnit* 框架提供测试类 (*TestCase*) 、测试套件 (*TestSuite*) 、测试方法、测试运行器
- ▶ 满足 *xUnit* 框架



JUnit4

- ▶ *JUnit4 使用 Java 5 中的注解 (annotation) , 以下是JUnit4 常用的几个annotation介绍*
 - ▶ *@Before: 初始化方法, 每开始一个新的测试都要执行一次, setup()*
 - ▶ *@After: 释放资源, 每开始一个新的测试都要执行一次, tearDown(),*
 - ▶ *@Test: 测试方法, 对应: testAdd()*
 - ▶ *@Ignore: 忽略的测试方法*
 - ▶ *@BeforeClass: 针对所有测试, 只执行一次*
@AfterClass: 针对所有测试, 只执行一次



JUnit5

- ✦ 以下是在将现有JUnit 4测试迁移到JUnit Jupiter时必须注意的事项。
- ✦ `org.junit.jupiter.api`包中的注解。
- ✦ `org.junit.jupiter.api.Assertions`类中的断言。
- ✦ `org.junit.jupiter.api.Assumptions`类中的假设。
- ✦ `@Before`和`@After`已经不存在；取而代之的是`@BeforeEach`和`@AfterEach`。
- ✦ `@BeforeClass`和`@AfterClass`已经不存在；取而代之的是`@BeforeAll`和`@AfterAll`。
- ✦ `@Ignore`已经不存在：取而代之的是`@Disabled`。
- ✦ `@Category`已经不存在：取而代之的是`@Tag`。
- ✦ `@RunWith`已经不存在：取而代之的是`@ExtendWith`。
- ✦ `@Rule`和`@ClassRule`已经不存在；取而代之的是`@ExtendWith`；关于部分规则的支持请参阅后续章节。



单元测试策略

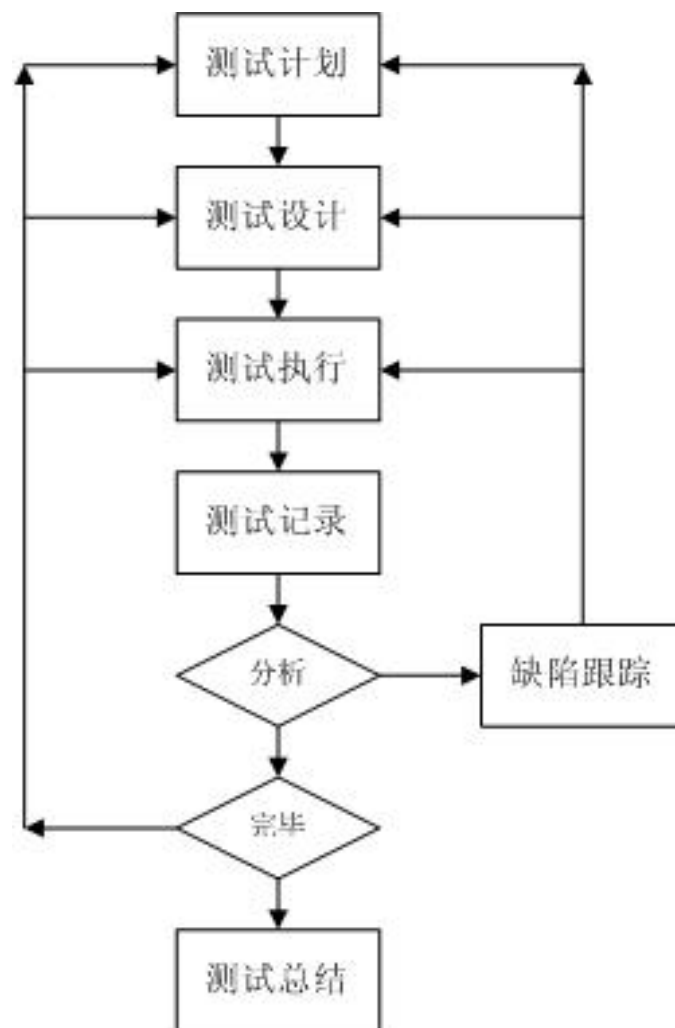
- ▶ 什么时候测试？XP开发理论讲究TDD，即测试驱动开发，先编写测试代码，再进行开发。
- ▶ 先编写产品的框架,是指先编写类、方法空的实现
- ▶ 编译通过然后编写测试类，针对产品类的功能编写测试用例，这时方法名、参数表、返回类型都应该确定下来了
- ▶ 然后编写产品类的代码，每写一个功能点都运行测试，随时补充测试用例。
- ▶ 编写的测试代码以后需修改的可能性比较小。

单元测试的过程和文档管理



过程:

1. 在详细设计阶段完成单元测试计划。
2. 建立单元测试环境，完成测试设计和开发。
3. 执行单元测试用例，并且详细记录测试结果。
4. 判定测试用例是否通过。
5. 提交《单元测试报告》。



单元测试的文档

1. 《软件需求规格说明书》、《软件详细设计说明书》 →
《单元测试计划》
2. 《单元测试计划》、《软件详细设计说明书》 →
《单元测试用例》
3. 《单元测试用例》文档及《软件需求规格说明书》、《软件详细设计说明书》 →
《缺陷跟踪报告》 / 《缺陷检查表》
4. 《单元测试用例》、《缺陷跟踪报告》、《缺陷检查表》 →
评估 →
《单元测试报告》

单元测试常用工具简介



工具分类:

- 静态分析工具
- 代码规范审核工具
- 内存分析工具
- 覆盖率分析工具
- 性能分析工具
- 测试数据生成工具
- 测试框架工具
- 测试结果比较工具
- 测试度量工具
- 测试文档生成和管理工具

单元测试工具

- ▲ JUnit
- ▲ Parasoft Jtest
- ▲ Parasoft C++Test



You now know ...

- ★ ... *static white-box testing*
- ★ ... *code reviews*
- ★ ... *informal code inspections*
- ★ ... *formal code inspections*
- ★ ... *code review checklists*
- ★ *xUnit, JUnit, CppUnit*
- ★ 单元测试工具

