第三节 开发环境、规范与测试

詹文翰

zhanwenhan@163.com

课程安排

1. 软件工程与软件建模
2. 敏捷开发与项目协作
3. 开发环境、规范与测试
4. 需求、设计与程序框架
5. 打包解包及其关键技术
6. 压缩解压及其关键技术
7. 加密解密与系统集成
8. 软件发布与项目收尾

软件开发环境

软件开发环境

软件开发环境是指用于进行软件开发的工具、平台和设置的组合。它提供了一个开发者可以编写、测试和分析代码的环境。其通常包括：

1. 编程语言和框架：编程语言是开发者用来编写软件的规则和语法。框架是一组提供了特定功能和结构的代码库，简化了开发过程。
2. 开发和调试工具：包括集成开发环境、文本编辑器、编译器、连接器、调试器等。
3. 版本控制系统：用于跟踪和管理代码的不同版本。
4. 测试和分析工具：测试工具用于自动化测试和验证软件的功能和性能。分析工具用于评估和优化软件的性能，识别性能瓶颈和优化机会。
5. 集成和部署工具：集成工具用于将不同的开发工具和系统集成在一起，实现自动化的构建、测试和部署流程。部署工具用于将软件应用程序部署到目标环境中，例如服务器、云平台等。

不同的开发者通常会根据自己的偏好和项目要求选择适合的软件开发环境。

软件开发环境

《实验指导书1》给出了一个使用VSCode远程连接Linux服务器进行C++语言开发的软件开发环境搭建教程，可供参考。

软件开发环境各不相同，选择合适的环境即可。但是，对于项目协作来说，软件开发环境**务必统一**，以防止出现不必要的麻烦。

统一开发环境的方法（推荐）：

1. 远程共享同一套软件开发环境
2. 容器/虚拟机镜像复制
3. 基础设施即代码

软件编码规范

除软件开发环境需要统一外，制定并统一软件开发规范同样重要：

1. 统一代码风格，便于代码维护；
2. 降低出现程序Bug的风险；
3. 有助于进行代码审查；
4. 养成良好的编程习惯；

编码规范实例：

Google - 开源项目风格指南

华为 - 软件编程规范

软件测试

软件测试定义

软件测试是为了发现软件开发过程中所发生的错误而执行的工作。

据统计，在软件开发过程中，软件测试的工作量约占软件开发总工作量的40%左右。

软件测试的方法可根据不同的维度被分为若干类。

软件测试分类

按是否运行被测软件分类：可分为静态测试和动态测试。

软件测试分类

按是否运行被测软件分类：可分为静态测试和动态测试。

静态测试：不实际运行被测软件，而只是静态地检查程序代码和文档，来发现软件错误。

通常，代码走查，代码审计均可被看作静态测试。

代码静态分析工具介绍：Lint程序。

软件测试分类

按是否运行被测软件分类：可分为静态测试和动态测试。

动态测试：通过实际运行程序来检验软件的动态行为和运行结果的正确性。

动态测试又可根据不同的维度进行分类。

软件测试分类

按是否运行被测软件分类：可分为静态测试和动态测试。

动态测试：通过实际运行程序来检验软件的动态行为和运行结果的正确性。

动态测试又可根据不同的维度进行分类。

根据**是否查看实现代码**：可将动态测试分为黑盒测试、白盒测试、灰盒测试。

软件测试分类

按是否运行被测软件分类：可分为静态测试和动态测试。

动态测试：通过实际运行程序来检验软件的动态行为和运行结果的正确性。

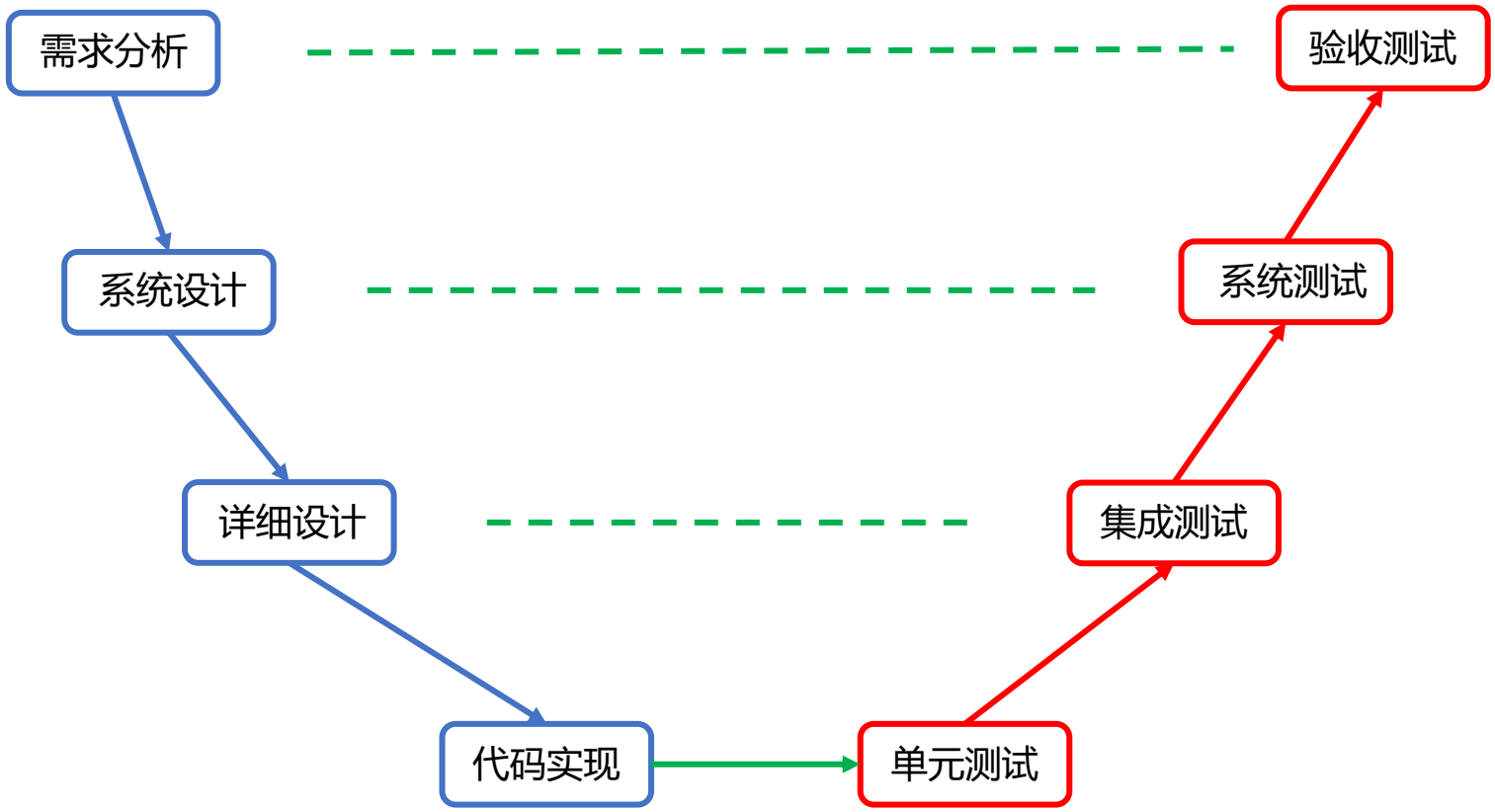
动态测试又可根据不同的维度进行分类。

根据是否查看实现代码：可将动态测试分为黑盒测试、白盒测试、灰盒测试。

根据**测试所对应的软件开发阶段**：可将动态测试分为单元测试、集成测试、系统测试、验收测试。

根据测试所对应的软件开发阶段

V字模型

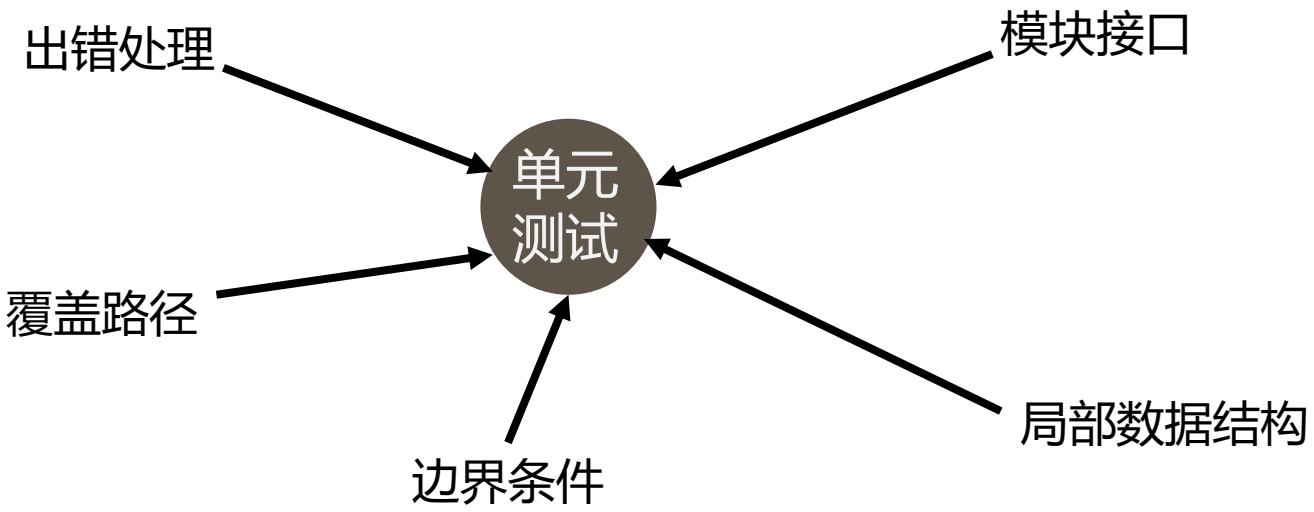


单元测试 验证 模块实现

单元测试又称为模块测试，是针对软件设计/实现的最小单位(模块)进行正确性检验的测试工作。

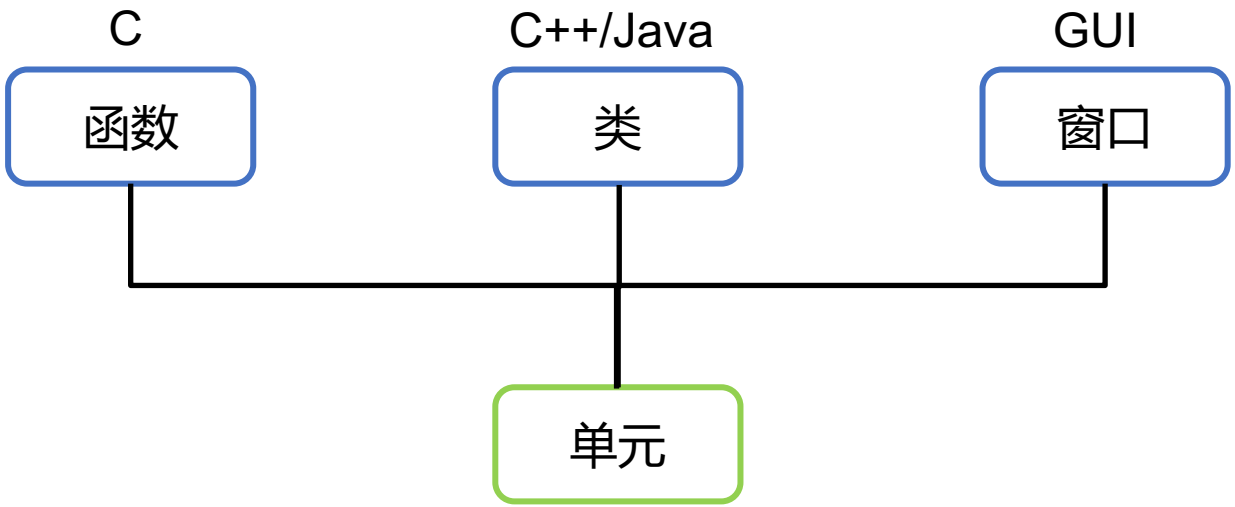
目的：检查每个程序单元能否正确实现详细设计说明中的模块功能、性能、接口和设计约束等要求，以及发现各模块内部可能存在的各种错误。

单元测试一般使用白盒测试方法，测试以下内容：



单元测试 验证 模块实现

对于不同的语言，会有不同的单元，比如，C语言的单元是函数，C++和Java的单元是类，图形化的软件的单元可以指一个窗口或一个菜单等。



集成测试 验证 架构设计

集成测试也称为组装测试、联合测试。它将已通过单元测试的模块集成在一起，主要测试模块之间的衔接、通信和互动。

从组装策略而言，可以分为两类：
1）一次性组装
2）增量式组装(包括自顶向下和自底向上两种方法)

集成测试一般采用黑盒测试或者灰盒测试。

系统测试 验证 系统设计

系统测试将软件作为整个计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行一系列的组装测试，以验证所开发软件是否达到系统所设计的要求或指标。

冒烟测试、性能测试、压力测试都属于系统测试。

系统测试基本上都是黑盒测试。

验收测试 验证 需求分析

验收测试与需求分析相对应，用来验证软件/服务在商业上是否满足了用户要求或市场需求，是否达到了发布的标准。

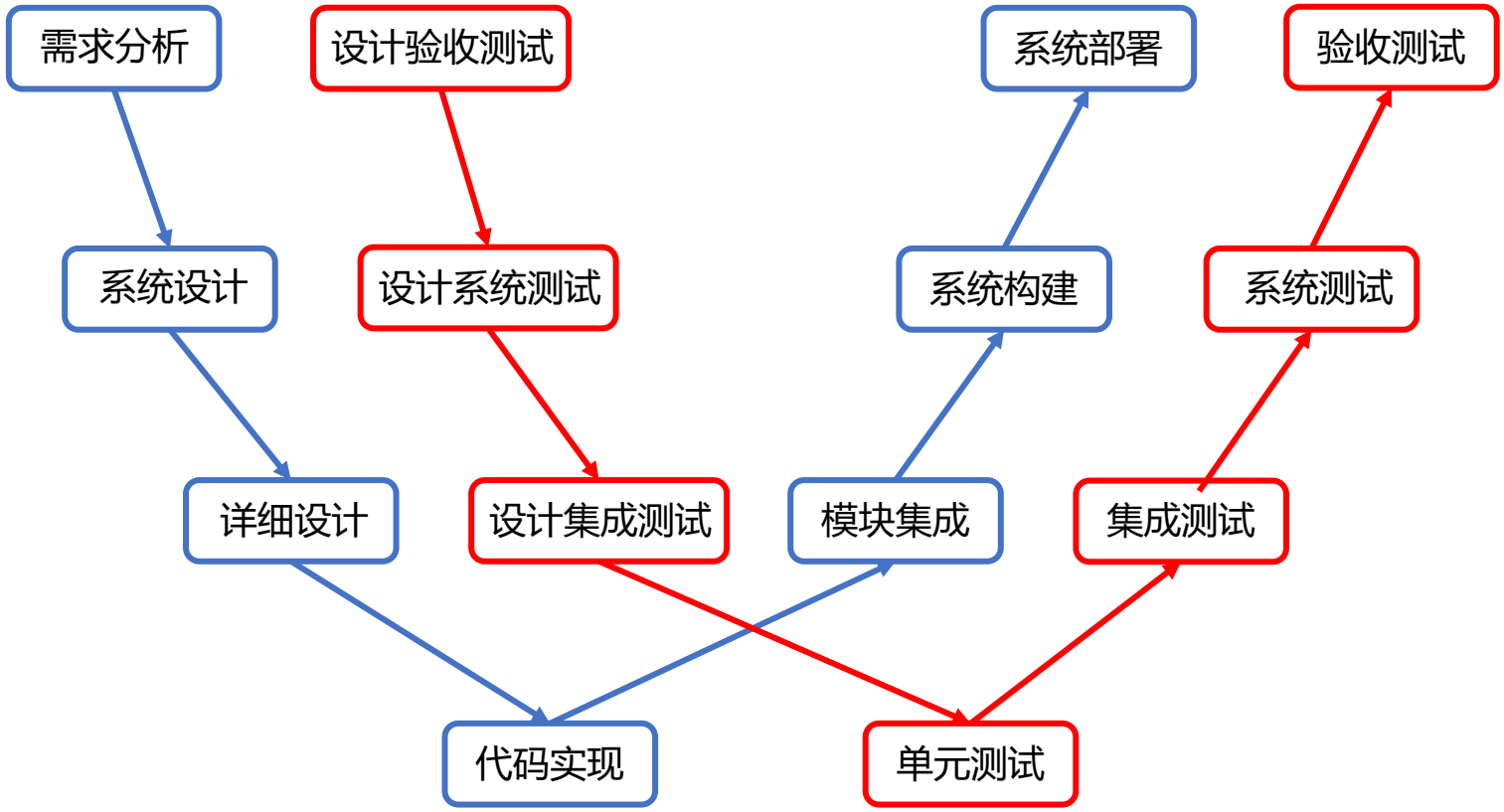
验收测试和系统测试都是针对整个软件/服务的，然而它们的区别在于：系统测试和之前的单元测试和集成测试都是在检验是否正确地制造了东西，而验收测试是在检验是否制造了正确的东西。

Doing things right VS Doing the right things

验收测试基本上都是黑盒测试。

测试驱动开发

W字模型


需求分析
设计验收测试
系统部署
验收测试
系统设计
设计系统测试
系统构建
系统测试
详细设计
设计集成测试
模块集成
集成测试
代码实现
单元测试

测试用例

软件测试是基于测试用例进行的，测试用例由**测试数据**和**预期结果**构成。

为了发现程序中的错误，应竭力设计能暴露错误的测试用例，好的测试用例是发现至今为止尚未发现的错误。

一个合格的测试用例需要包括以下内容：

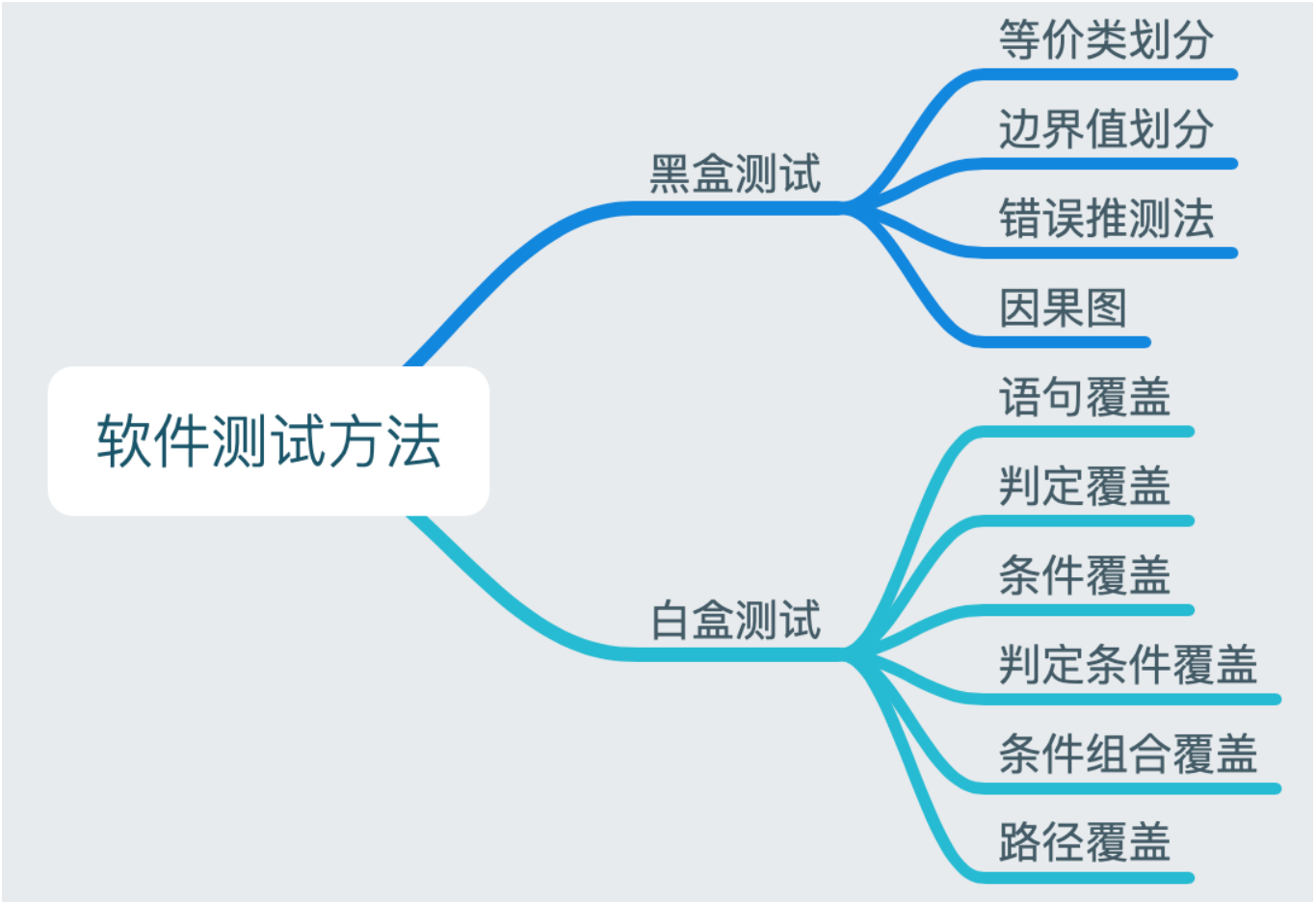
程序版本号	模块名称	用例编号	用例名称	用例级别	预置条件	测试输入	操作步骤	预期结果	实际输出	测试时间	测试人员

测试用例

项目名称	飞机订票系统		项目编号	001	
开发人员	No_1		模块信息	登录页面	
用例作者	No_1		参考信息	需求说明、设计说明	
测试类型	功能测试	设计日期	2012 年 12 月 5 日	测试人员	***
测试方法	手工测试和自动化测试结合			测试日期	2012年12月13日
测试对象	测试用户能否正常登陆				
前置条件	存在正确的用户名：登陆页面正常装载（用户名不固定，密码为 mercury 不区分大小写）				

用例编号	操作	输入数据	预期结果	实际结果	测试状态
01	输入正确的用户名、正确的密码，按 Enter 键	用户名：mingzi，密码：mecury	正常登陆	正常登陆，转入对应的系统页面	Pass
02	输入正确的用户名，正确的密码（没区分大小写），按 Enter 键	用户名：mingzi，密码：MERTURY/mecury	正常登陆	正常登陆，转入对应的系统页面	Pass
03	输入的用户名为空，正确的密码（没区分大小写），按 Enter 键	密码：MERTURY	登陆失败	提示请输入用户名	fail
04	输入正确的用户名，密码为空，按 Enter 键	用户名：mingzi	登陆失败	提示请输入密码	fail
05	输入的用户名和密码均为空，按 Enter 键		登陆失败	提示请输入用户名	fail
06	输入错误的用户名，正确的密码（没区分大小写），按 Enter 键	用户名：zi，密码：MERTURY/mecury	登陆失败	提示请输入大于 4 个字符长度的用户名	fail

软件测试方法


软件测试方法
黑盒测试
等价类划分
边界值划分
错误推测法
因果图
白盒测试
语句覆盖
判定覆盖
条件覆盖
判定条件覆盖
条件组合覆盖
路径覆盖

等价类划分

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

等价类划分

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

所谓等价类，是输入条件的一个子集合，该输入集合中的数据对于揭示程序中的错误是等价的。从每一个子集中选取少数具有代表性的数据，从而生成测试用例。

等价类划分

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

所谓等价类，是输入条件的一个子集合，该输入集合中的数据对于揭示程序中的错误是等价的。从每一个子集中选取少数具有代表性的数据，从而生成测试用例。

等价类又分为**有效等价类**和**无效等价类**。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）。

等价类划分

例：设计一个档案管理系统，要求用户输入以年月表示的日期。假设日期限定在1990年1月-2049年12月，并规定日期由6位数字字符组成，前4位表示年，后2位表示月。

日期检查输入条件等价类

输入条件	有效等价类	无效等价类
日期的类型及长度	（1）6位数字字符	（2）有非数字字符 （3）少于6位数字字符 （4）多于6位数字字符
年份范围	（5）在1990-2049之间	（6）小于1990 （7）大于2049
月份范围	（8）在01-12之间	（9）等于00 （10）大于12

等价类划分

设计测试用例，覆盖所有有效等价类；
再针对每一个无效等价类，设计一个测试用例。

日期检查测试用例

测试数据	期望结果	覆盖等价类
200211	有效输入	（1）（5）（8）
95June	无效输入	（2）
20036	无效输入	（3）
2001006	无效输入	（4）
198912	无效输入	（6）
200401	无效输入	（7）
200100	无效输入	（9）
200113	无效输入	（10）

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。**通常边界值分析法是作为对等价类划分法的补充**，这种情况下，其测试用例来自等价类的边界。

采用边界值分析法设计的日期检查测试用例

测试数据	期望结果	选取理由
199001	有效输入	最小年份最小日期
199012	有效输入	最小年份最大日期
199000	无效输入	刚好小于最小日期
199013	无效输入	刚好大于最大日期
204901	有效输入	最大年份最小日期
204912	有效输入	最大年份最大日期
204900	无效输入	刚好小于最小日期
204913	无效输入	刚好大于最大日期

白盒测试

根据测试的强度由低到高：语句覆盖、判定覆盖、条件覆盖、判定条件覆盖、条件组合覆盖、路径覆盖。

六种覆盖标准比较

语句覆盖	程序中每个语句至少都能被执行一次
判定覆盖	程序中的每一个分支至少都通过一次
条件覆盖	程序中每个判定的每个条件的各种可能值至少出现一次
判定/条件覆盖	同时满足判定覆盖标准和条件覆盖标准
条件组合覆盖	每个判定中各条件的每种组合至少出现一次
路径覆盖	程序中的每条可能路径至少执行一次

参考：https://blog.csdn.net/huangli414/article/details/77876044

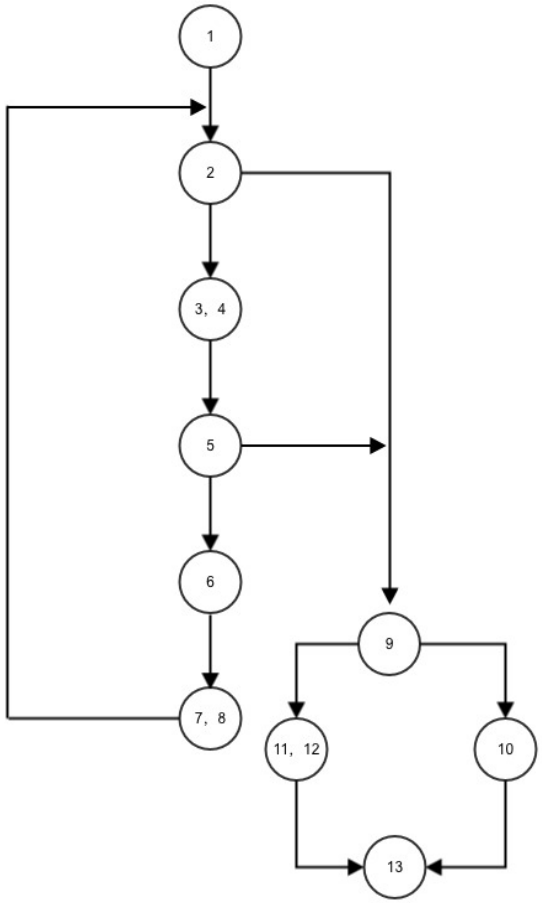
白盒测试

路径覆盖的例子。选定一个系统模块，绘制流程图，并采用路径覆盖的测试方法编码测试。

流程图（右图），基本路径包括：

path1：1 - 2 - 3,4 - 5 - 6 -7,8 - 2 - 9 - 10 - 13
path2：1 - 2 - 3,4 - 5 - 6 -7,8 - 2 - 9 - 11,12 - 13
path3：1 - 2 - 3,4 - 5 - 9 - 10 - 13
path4：1 - 2 - 9 - 10 – 13
……

白盒测试的复杂度由环路复杂度（圈复杂度）决定。



常见测试工具

白盒测试框架：gtest、boosttest

Googletest（通常简称为gtest）是由Google开发的一个C++单元测试框架，用于编写和运行自动化测试。它支持各种测试方法，包括单元测试、集成测试和功能测试等，并提供了丰富的测试工具和测试报告，以帮助开发人员快速、准确地测试和验证代码的正确性和健壮性。

内存测试工具：valgrind

Valgrind是一套Linux下，开放源代码的仿真调试工具的集合。Valgrind由内核以及基于内核的其他调试工具组成。内核模拟了一个CPU环境，并提供服务给其他工具；而其他工具类似于插件，利用内核提供的服务完成各种特定的内存调试任务。

性能测试工具：gprof、perf

gprof(GNU profiler)是GNU binutils工具集中的一个工具，Linux系统当中会自带这个工具。它可以分析程序的性能，能给出函数调用时间、调用次数和调用关系，找出程序的瓶颈所在。在编译和链接选项中都加入-pg之后，gcc会在每个函数中插入代码片段，用于记录函数间的调用关系和调用次数，并采集函数的调用时间。

实验环节

对于选择Linux C++平台进行项目开发的同学，完成本课程实验一：

- 实验内容见《实验指导书1》
- 要求掌握基本的程序编译和调试方法
- 要求各小组构建好自己的开发环境，组内开发环境统一
- 不需要提交实验报告

对于选择Linux C++平台进行项目开发的同学，完成本课程实验四：

- 实验内容见《实验指导书4》
- 要求掌握大型程序的构建方法：make
- 要求学会使用gtest、valgrind、gprof、perf等测试框架和工具
- 不需要提交实验报告

项目要求：

- （推荐）采用gtest测试框架，完成项目中任一模块的白盒单元测试。测试代码留在项目源代码中。
- （推荐）在项目开发过程中，使用valgrind、gprof、perf等工具。在项目报告中体现出来。
- 项目报告中，至少包含两种以上测试类型（单元测试、集成测试、性能测试、稳定性测试等）。
- 项目报告中，至少包含15个以上测试用例。