测试基础

测试方法：

白盒测试覆盖分析

黑盒测试

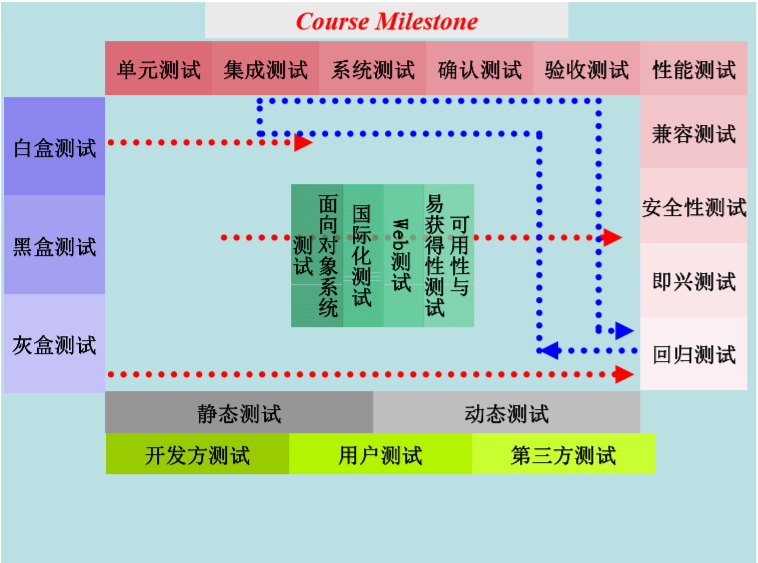
测试策略

单元，集成测试

性能，安全测试

系统，确认，回归测试

5实验报告，每个十几页+作业+课堂测验



1. 软件测试

测试背景：

二十世纪六十年代 “软件危机“，原因：缺乏规范化工程约束，缺陷积累放大

解决方法：

软件工程：质量，过程，方法，工具

软件缺陷原因：**需求说明书**，**设计**，编码，其它

意义：保证软件质量，深入每个开发阶段确保产品符合用户需求

测试概念

定义，目的，**原理**，类型

IEEE ：运行测试**动态**软件，分析**静态**软件项

目的：确保软件产品、过程质量

十大原理：用户需求至上、测试是有计划的活动（贯穿整个开发）、缺陷有**集群性**、测试应走向大规模、不可能完全测试、第三方独立进行测试、测试无法找到所有缺陷、缺陷对已进行测试有免疫力、需注意最优测试量（有风险）、并非所有缺陷都需修复（bug, feature）

V字模型：需求-验收、概要设计-系统测试、详细设计-集成测试、编码-单元测试

**W模型**：将静态分析与测试准备提前，形成测试驱动开发（设计VAV与测试准备）

前置测试模型：将W模型过程与产品细化

H模型：测试与其它流程并行，准备-就绪点-执行

测试流程：拟定计划、编制大纲、生成用例（前提-输入-期望输出 代表性，可判定性，可再现性）、实施测试、分析结果

白盒：基于代码测试，测试逻辑行为（91%结构测试/逻辑驱动/基于结构）

静态：桌面检查、同行评审、走查、代码审查（Inspection后续基石）

不执行软件，审查软件设计、结构、代码（结构化分析）

走查：组为单位以**会议**形式进行，审查人员模拟运行（非仔细筹备）

审查：正式高度组织化的过程，讲解者非开发员

动态：程序（逻辑）图、测试用例、执行测试、分析覆盖、判定结果

依据源代码生成测试用例并执行

程序流程图：单入单出口

逻辑覆盖准则：

语句覆盖：每条语句执行

判定覆盖：执行判定所有可能结果（经过所有分支&语句）

条件覆盖：执行判定中每个条件取值可能（**短路符**&&、||会忽略执行条件）

判定条件覆盖：条件与由其所组成的判定所有可能值（条件正误交错）

条件组合覆盖：保证判定内部**组合取值**至少一次（ n:判定中最多条件数）

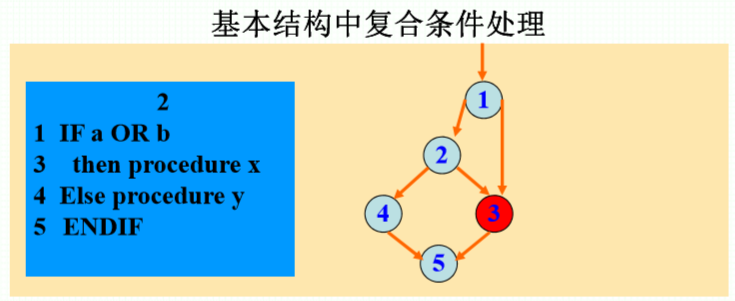
路径覆盖：保证经过可能执行路径/环路组合（部分路径可能不可达）

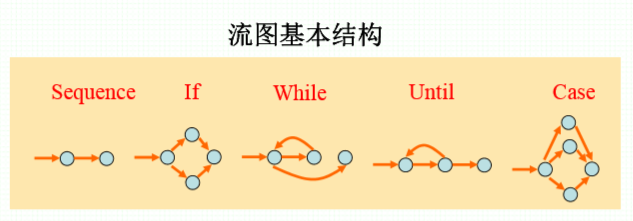
基于数据流测试：程序是对数据加工处理

通过变量定义、使用位置来选择测试路径

Du-path：从变量定义节点到使用节点的一条路径

Dc-path：路上只能有一定义





黑盒：基于需求测试

脚本：自动化的黑盒测试（83%可测逻辑）

域：条件数+1，围成区域数+1

基本路径：贯穿程序的、至少引入一组**新语句/条件/边**的程序通道（有顺序的）

环复杂度：基本路径数目、测试用例最小量

基本路径测试：寻找基本控制流，根据基本路径设计用例

开放域：基本路径理论要求**强连通图**、需引入尾置头的**虚拟边**

循环测试：

简单循环：跳过、执行一次或多次循环

嵌套循环：先测最内层、设为典型值、逐步测外层

串接循环：分别简单循环（循环独立）、采用嵌套循环（循环相关）

等价类：一组输入值、程序行为相同、则此组输入值为一个划分

有效等价类：满足程序输入、用例可覆盖多条等价类

无效等价类：不满足程序输入要求、只违反一条规则、测试用例仅覆盖一条

边界值：各种变量值的极限、存在边界值的参数数n、边界值条件数m

测定每一个参数边界、选择决定因素、固定其它参数为典型值、系统整体边界

4n+1：取值在值域范围内边界

6n+1：选取边界外挨着边界的值、测试异常情况健壮性

3m：-1/自身/+1

因果分析：多个输入条件相互关联

因果图：划分导致问题的原因描述原因间关系（注意原因数）

关系：恒等、非、或、与（因果间）

原因间：互斥(最多一个1)、包含(最多一个0)、唯一(只有一个1)、要求、屏蔽

原因组合管理，列决策表、读懂因果图

合并相似规则：两条或以上相似条件，具有相同动作

基于图/模型：

状态机：事件导致状态变化、工作流：是否通过导致对象变化、有限自动机：遍历图

正交数组：通过正交表选择合理的测试数据（应用图形界面）

保证任意两个输入所有组合被包含，每个因子的水平出现次数、任意两列水平搭配相同

测试步骤：确定因子和水平、选择正交表（因子数接近但略大）、变量映射形成用例、**增补特殊用例**

黑盒测试：被测软件内部逻辑未知的测试方法总称

单元测试：

目标：体现预期**功能**，覆盖设定**逻辑**，内部**数据完整性**，接受**正确/非法输入**，**性能安全性**

Driver-Stub：使用driver调用E（被测者），再编写stub替代G被E调用

Driver：获取E所需数据、调用E、输出E返回值

Stub：接受E输入、检查正确性、返回E所需值、输出提示

集成测试：发现与**接口连接**相关的错误

自底向上：无需编写stub、需编写大量drive（需求设计变化）

自顶向下：需大量stub、较早发现设计错误、对底层错误发现较晚（需求清晰）