1. 测试概述

危机原因：缺乏规范化工程约束，缺陷积累放大

软件缺陷原因：需求说明书，设计，编码，其它

意义：保证软件质量的重要手段，深入每个开发阶段确保产品符合用户需求

Bill Hetzelt: 测试是建立一种信心，认为程序能够按预期执行

Glenford J.Myers: 测试是为发现错误而执行的过程

Myers: 测试是为了证明程序有错，测试用例与成功的测试均为了发现未知错误

测试目的：确保软件质量，确保开发过程方向正确性

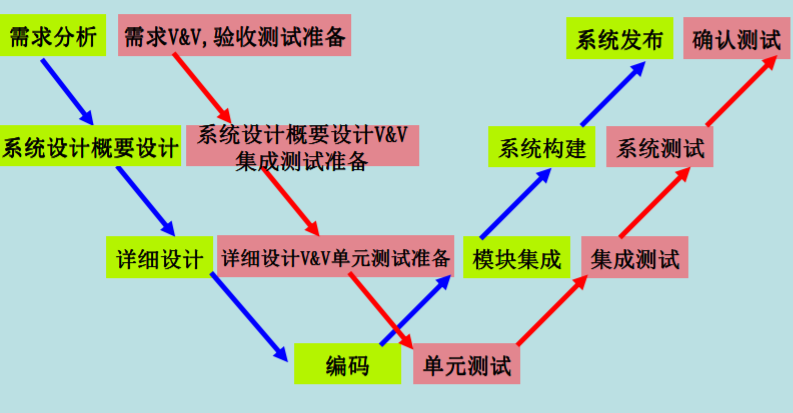
**十大原理**：用户需求至上、测试是有计划的活动（贯穿整个开发）、缺陷有**集群性**、测试应走向大规模、不可能完全测试、第三方独立进行测试、测试无法找到所有缺陷、缺陷对已进行测试有免疫力、测试是有风险的活动（最优测试量）、并非所有缺陷都需修复（bug, feature）

测试用例：代表性，可判定性，可再现性

测试类型：

|  |
| --- |
| 测试技术：白盒，黑盒，灰盒 |
| 开发阶段：单元，集成，系统，确认，回归，性能，验收 |
| 执行状态：静态，动态 |
| 执行主体：开发，用户，第三方 |
| 特殊：国际化，兼容性，安全性 |

W模型：将静态分析与测试准备提前，形成测试驱动开发



测试工作量：45%

1. 白盒测试

定义：基于代码测试，根据程序逻辑行为，生成用例

静态白盒：不执行软件，审查软件设计、结构、代码（结构化分析）

桌面检查（代码编写者）：优点（编码者易理解自己的代码、开销小无进度限制、尽早发现缺陷），缺点（开发人员不是测试最佳人选、有效性难以保证），特点（团体小、非正式）

代码走查（团队会议）：特点（由团队通过会议完成、与会者充当计算机执行测试用例、开发人员及时回答问题）

代码审查（ALL）：优点（发现缺陷、提高质量、尽早定位缺陷集群位置），缺点（费时、涉及人员多、参与者不一定全部理解程序）

动态白盒：提供源代码与可执行程序，根据代码设计执行测试用例

|  |  |  |
| --- | --- | --- |
| 语句覆盖 | 执行每条语句 | 100%语句覆盖困难（处理错误，小概率，不可达），无法发现NULLPTR等问题 |
| 判定覆盖 | 每个判定TF均包含 | 优点（简单，包含语句覆盖）  缺点（忽略了条件表达式内部） |
| 条件覆盖 | 判定内每个条件TF均包含 | 不保证执行所有分支  可能比**语句**或**判定**覆盖**弱** |
| 判定条件覆盖 | 判定与其内的条件TF均包含 | 错误屏蔽：原子条件取值对整体不可见  无法检查运算符错误 |
| 条件组合覆盖 | 判定内条件组合均出现 | 代价昂贵，为原子条件数目  **短路符**用于寻找最少测试用例集  某些条件组合内部互斥 |
| 路径覆盖 | 所有可能路径与环路 | 优点（相对彻底测试）  缺点（路径分支指数级增加，不可达路径，未测试分支中条件 |

流图：抽象程序流程，通过对图问题的求解得到基本路径进行测试

基本路径：贯穿程序的，至少引入一组**新语句**、**新判断**的程序通道

数学定义：其它路径的向量，可以通过基本路径的向量**线性运算组合**出来（**基本路径不唯一**）

环复杂度：度量基本路径数，是语句覆盖所需测试用例上线（域数，E边-N节点+2，判断数+1）

