**《软件测试技术》复习大纲**

**第一章**

**1 软件测试学科的发展**

1957年之前，调试为主（Debugging Oriented）

1957-1978，以**功能验证**为导向，测试是证明软件是正确的（正向思维）

1978-1983年，以**破坏性检测**为导向，测试是为了找到软件中的错误（逆向思维）

1983-1987年，以**质量评估**为导向，测试是提供产品的评估和质量度量

1988年起，以**缺陷预防**为导向，测试是为了展示软件符合设计要求，发现缺陷、预防缺陷。

2 正向测试与反向测试的定义，关系

正向：测试是为了验证软件是否符合用户需求，即验证软件产品是否能正常工作。

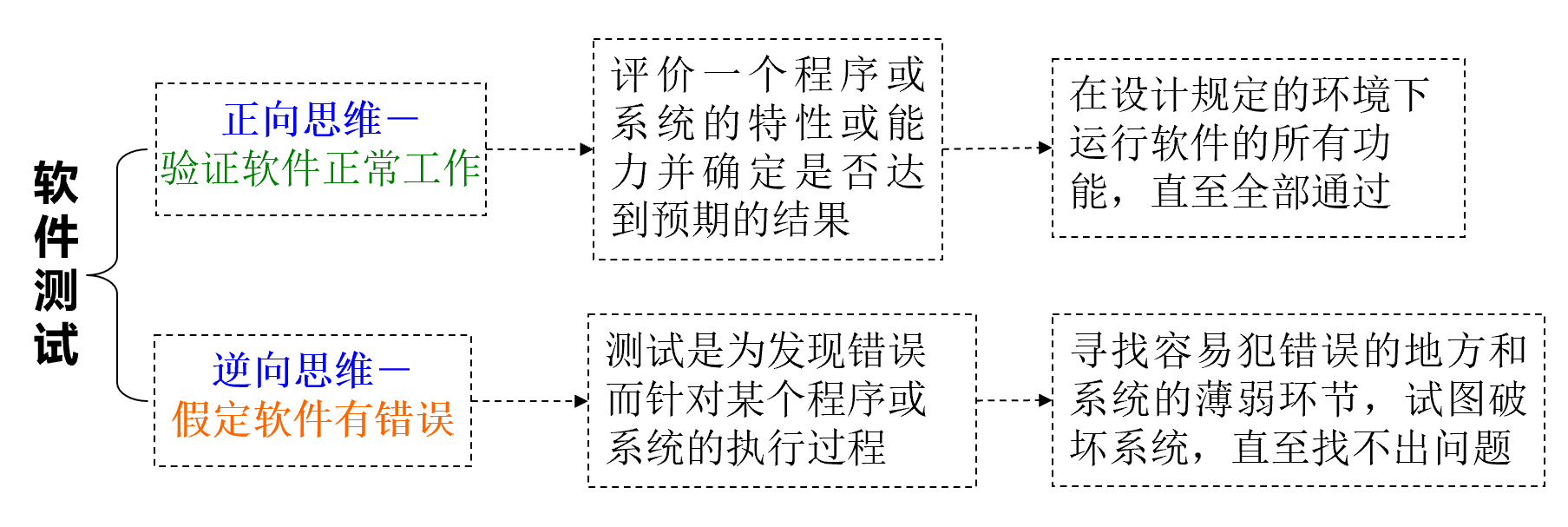
反向：测试是为了证明程序有错，而不是证明程序无错误。

关系： 认知决定行为。

正向测试思维是验证软件是否能正常工作，在设计规定的环境下，运行软件的所有功能，直至全部通过；

反向测试思维是假定软件有错误，寻找容易犯错误的地方和系统的薄弱缓解，试图破坏系统，直至找不出问题。

这正反两方面的观点是从不同的角度看问题,一方面通过测试来保证质量,另一方面又要改进测试方法和提高软件测试的效率,两者应该相辅相成。因为测试不能证明软件没有丝毫错误、不能确认所有的功能可以正常工作,所以测试要尽可能找出那些不能正常工作,不一致性的问题。软件测试就是在这两者之间获得平衡,但对于不同的应用领域,两者的比重是不一。



3 从**从经济视角认知软件测试**

以最小的代价获得最高的软件产品质量。要求软件测试今早开展工作，发现缺陷越早，返工的工作量越小，造成的损失越小。测试成本<缺陷造成的损失，测试才有意义。

**4** SQA，与软件测试关系

软件质量保证（Software Quality Assurance）是通过对软件产品有计划的进行评审和审计来验证软件是否合乎标准的系统工程，通过协调、审查和跟踪以获取有用的信息，形成分析结果以指导软件过程。

SQA指导、监督软件测试的计划和执行，督促测试工作的结果客观、准确和有效，并协助测试流程的改进；软件测试是SQA重要手段之一，为SQA提供所需的数据，作为质量评价的客观依据。

SQA是管理工作，侧重于对流程的评审和监控。

测试是技术工作，侧重对产品进行评估和验证。

第一章问题：

1. 在日常使用软件过程中，遇到哪些软件质量问题？
2. 软件测试的正反两方面观点会如何影响测试工作的？
3. 软件测试和软件开发的关系是怎样的？如何更好利用这种的关系？
4. 软件测试和质量保证之间的联系和区别？

第二章

1 缺陷定义，现象，判定准则

**定义**：任何程序、系统中的问题，和产品设计书的不一致性，不能满足用户的需求

IEEE（1983 729）：

从产品内部看，软件缺陷是软件产品开发或维护过程中所存在的错误、毛病等各种问题。

从外部看，软件缺陷是系统所需要实现的某种功能的失效或违背。

**现象**：

功能、特性没有实现或部分实现

设计不合理

实际结果和预期结果不一致

运行出错（包括运行中断、系统崩溃、界面混乱）

数据结果不正确、精度不够

用户不能接受的其他问题，如存取时间过长、界面不美观

**判定准则**：将被测试系统的实际输出与所期望的输出进行比较，从而判断是否有差异，即是否为缺陷。（作比较的对象：需求规格说明书、其他需求设计文档、竞争对手的产品)

2 软件缺陷产生的原因有哪？

技术问题：算法错误、计算和精度问题、接口参数传递不匹配

团队工作：沟通不充分、误解

软件本身：文档错误、用户使用场合、时间不协调或不一致，系统自我恢复或数据备份，灾难性恢复等问题。

3 产品质量的内容，内部，外部，使用质量

软件质量：软件产品具有满足规定的或隐含要求能力要求有关的特征与特征综合。

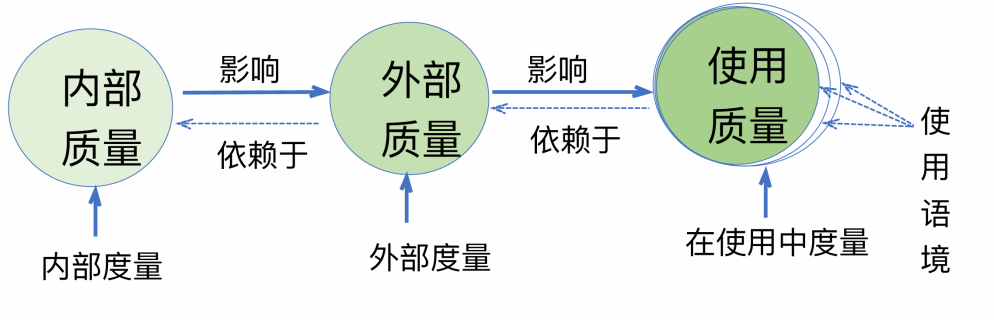
产品质量：人们实践产物的属性和行为，是可以认识，可以科学地描述的。并且可以通过一些方法和人类活动来改进质量。

产品质量标准：功能性、可用性、可靠性、性能、容量、可伸缩性、可维护性、兼容性、可扩展性

内部质量：涉及软件开发团队内部各部门工作质量的集中体现，涉及需求规格质量，设计水平，代码质量，以及测试人员质量和QA质量人员的监督作用，软件未交付，属于内部评价范畴。

外部质量：是指软件交付给客户时，除软件本身的其他附件，包括用户使用说明书，帮助文档，以及完整的软件，软件已交付，属于外部评价范畴。

使用质量：是指客户在使用软件时的感受和评价。软件已被运行，属于外部评价范畴，使用质量还掺杂诸多主客观因素。包括，有效性、效率、满意程度（是否有用、可被新人、愉悦、舒服）、免于风险程度（经济风险、健康和安全风险、环境风险）、上下文覆盖（上下文/场景的完整性）



4 如何理解**软件规格说明书缺陷**

最容易产生缺陷的部分：

1. 开发与用户沟通存在困难，对产品理解不一样
2. 靠想象描述系统的实现结果，有些特性不清楚
3. 需求变化的不一致性，用户的需求不断变化，不能及时在需求中得到正确描述
4. 对规格说明书不够重视，对其投入人力，时间不足
5. 沟通不够

5 verification,validation

Verification验证: 是否正确地构造了软件，即验证开发过程是否遵守已定义好的内容。验证产品满足规格设计说明书所定义的系统功能和特性。(开发者)

Validation有效性确认: 是否构造了用户需要的软件，即验证产品所实现的功能是否满足用户的需求。(用户)

6 软件测试不同层次测试的对象和任务

V模型。以下自底向上。

单元测试： 编码

对象：组件/模块/类/函数

任务：组件功能、健壮性、效率

集成测试： 详细设计

对象：单元之间接口

任务：组件之间的接口

系统测试： 系统/软件设计（系统功能测试是从用户角度）

对象：由单元构成的系统

任务：系统功能、安全性、健壮性、效率

验收测试： 需求功能设计

对象：系统承载的用户业务

任务：功能及用户界面、安全性、效率、用户的可接受性

7 静态测试的内容包括什么？开展相关活动时采用的形式有哪些？

内容：对软件产品的需求和设计文档、代码的评审（技术评审、文档评审）以及对代码的静态分析。（需求评审、设计评审、代码评审和代码分析）

评审的主要形式：互相评审(peer review)、走查(walk-through)、会议评审(inspection)

需求评审：审查需求规格说明书是否完整正确清晰。

产品评审：

评审对象：需求文档、设计和代码

设计评审：

保证需求能在设计中得到准确和完整的表示，即保证系统架构设计和产品功能规格说明书的质量。

借助UML等建模工具，对系统架构、可测试性、系统部署、设计规格说明书进行评审

静态分析：

人工检测：发现逻辑设计和编码错误

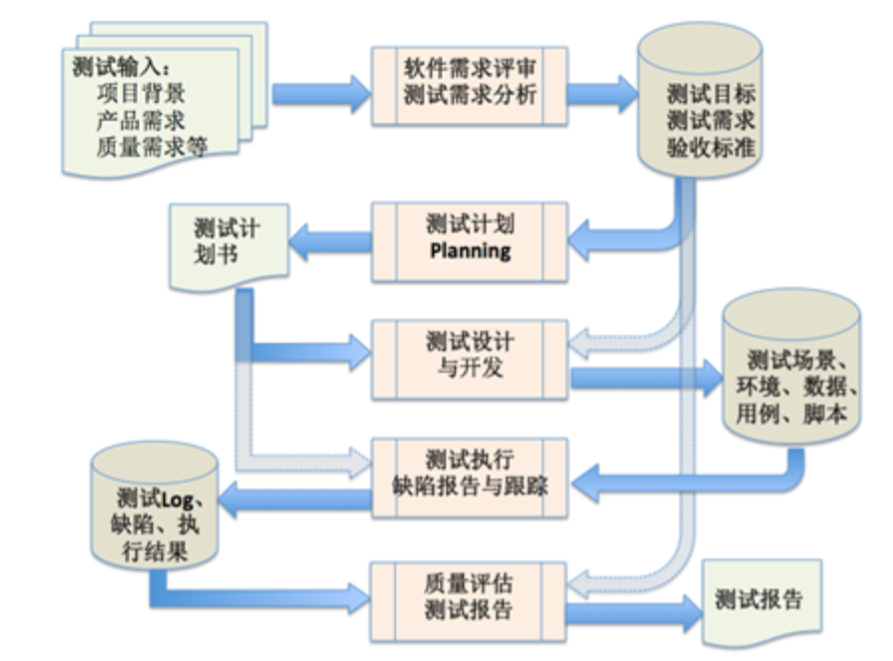
计算机辅助静态分析：检查程序逻辑的各种缺陷和可疑的程序构造。

代码静态分析：

通过词法分析、语法分析、控制流分析、数据流分析对程序代码进行扫描

采用模拟程序执行的技术，如符号执行、抽象解释、值依赖分析

8 测试工作的流程



9 软件测试的工作范畴

1. 测试需求分析：
   1. 明确测试范围，了解哪些功能点需要测试，哪些功能点不需要测试
   2. 知道哪些测试目标优先级高、哪些目标优先级低
   3. 要完成哪些相映的测试任务才能确保目标的实现
2. 软件测试策略：
   1. 测试方式：手工/自动化，静态/动态比重，ET/ST，自己团队测/众测/外包
   2. 测试方法：黑盒/白盒，基于数据流/控制流，完全组合测试/组合优化测试
   3. 测试过程：先测什么，后测什么
3. 测试计划：
   1. 目标/范围
   2. 测试项及优先级
   3. 测试风险识别与分析
   4. 指定测试策略
   5. 进度安排
   6. 资源配置
   7. 跟踪和控制机制
4. 测试设计：
   1. 总体设计：测试方案、结构的设计
   2. 详细设计：测试用例设计
5. 测试用例：
6. 测试执行：
   1. 手工执行
   2. 自动化执行
7. 测试结果和过程评估
   1. 结果评估：对测试结果分析，分析测试覆盖率、基于缺陷的趋势和分布分析、基于缺陷评估当前被测试的版本的质量
   2. 过程评估：结合测试计划评审，把计划的测试活动和实际执行活动进行比较，了解测试计划执行的情况和效果。

思考题：

1. 为何说缺陷是质量的对⽴⾯？
2. 为什么静态测试和 动态测试是⼀对对⽴统⼀体？
3. 为什么说测试需求分析是测试计划、测试设计的基础？

第三章

1 等价类

有效等价类、无效等价类

2 边界值法

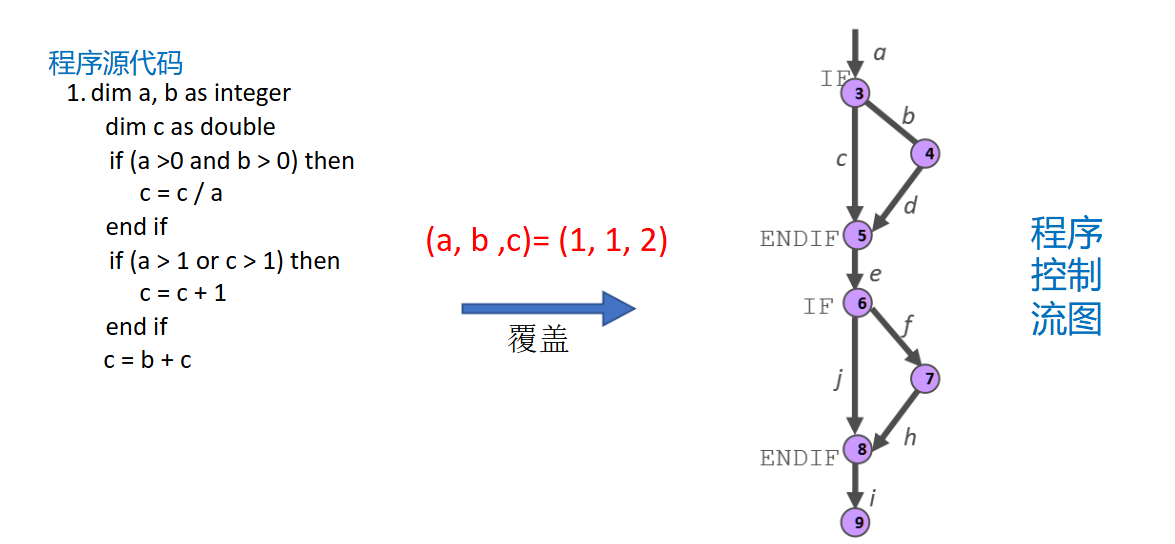
3 决策表，因果图法

判定表：条件/活动

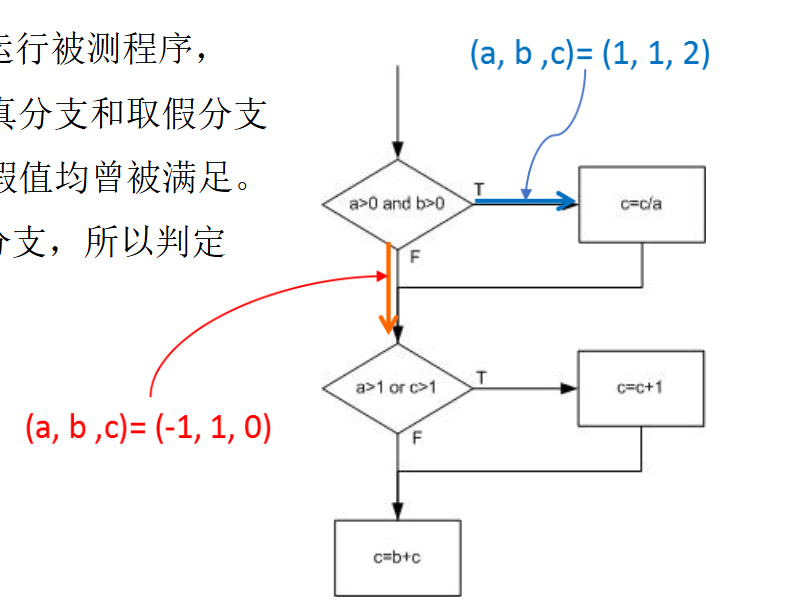
条件桩、动作桩、条件项、动作项、规则（条件组合的取值以及对应的动作）

4 各种逻辑覆盖法

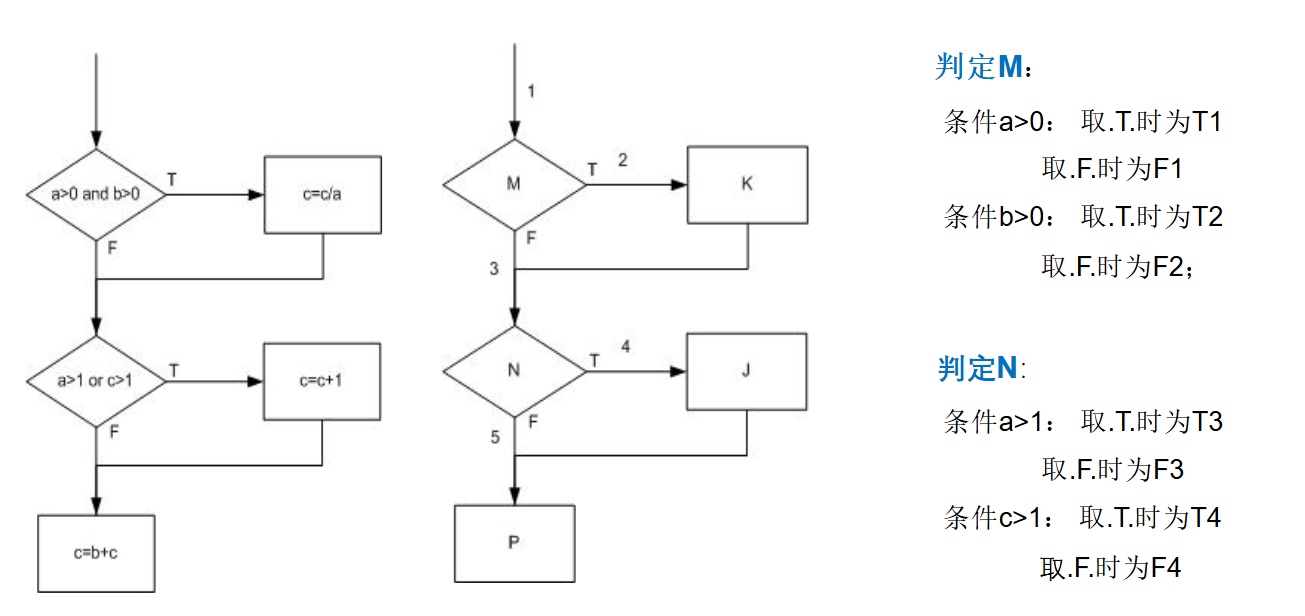
语句覆盖：程序中每个可执行语句至少被执行一次



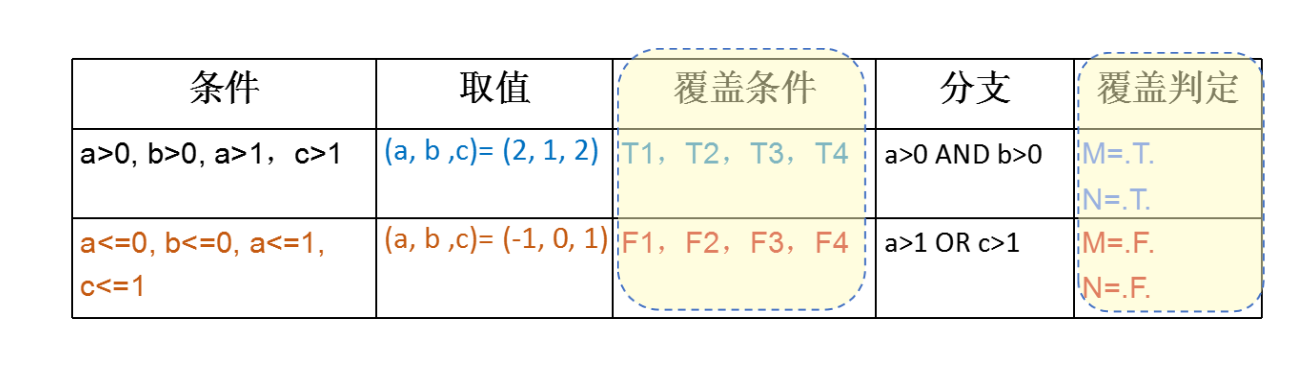
判定(分支)覆盖（DC）：程序中每个判断的取真分支和取假分支至少经历一次



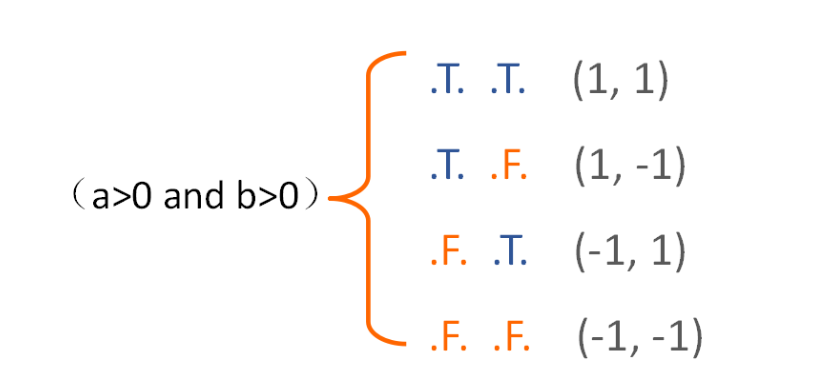
条件覆盖（CC）：每个判断中每个条件的可能取值至少满足一次



判定/条件覆盖（DC/CC）：判定覆盖和条件覆盖的交集，使判断条件中的所有条件可能取值至少执行一次，同时，所有判断的可能结果至少执行一次。



条件组合覆盖（MCC）：判断中的每个条件的所有可能组合至少出现一次，并且每个判断本身的判定结果也至少出现一次。



MC/DC覆盖：

1. 每个判定的所有可能结果至少能取值一次
2. 判定中的每个条件的所有可能结果至少取值一次
3. 一个判定中的每个条件独立地对结果产生影响
4. 每个入口和出口至少执行一次

（对于n个条件的输入，有>=n+1个测试用例)

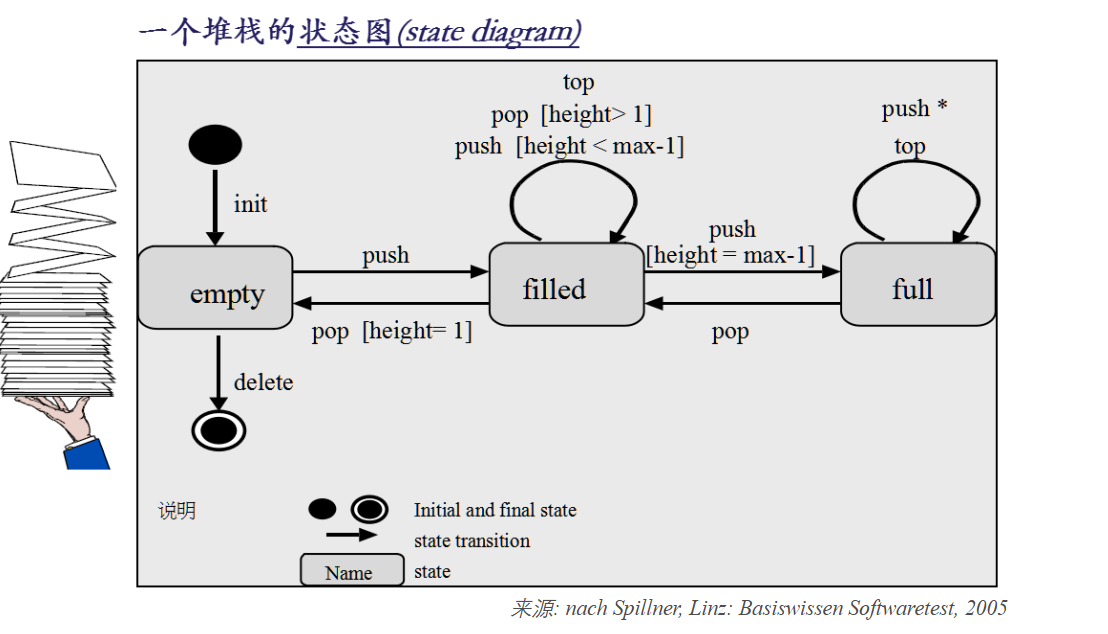
5 路径覆盖法（BPC）

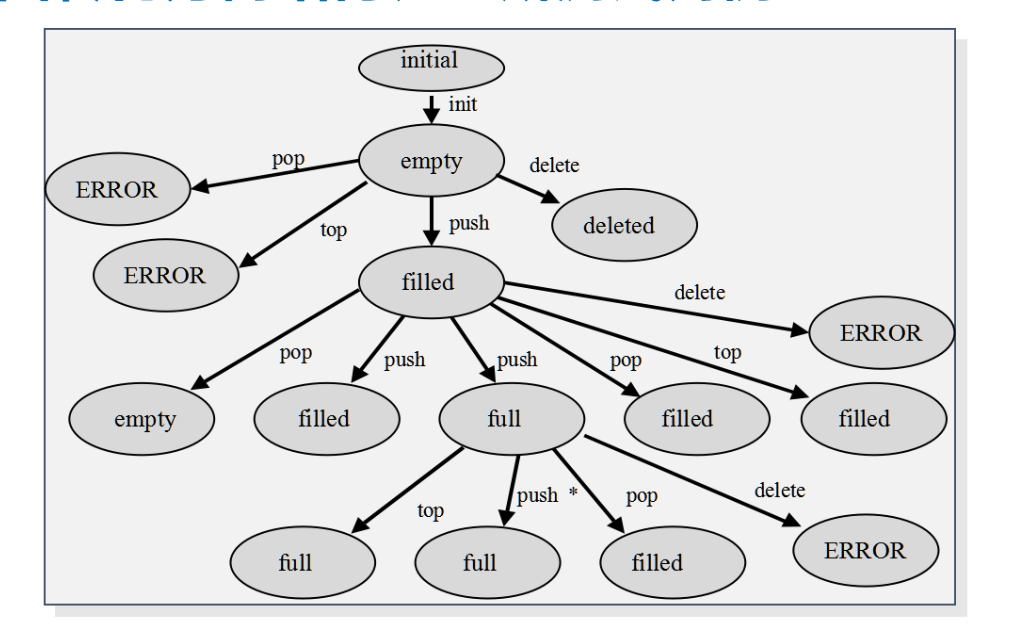
圈复杂度V(G) = 区域数量 = 连线数量-节点数量+2 = 简单可预测节点数量+1

6 功能图法，EFMS

功能图：状态迁移图和逻辑功能模型构成

EFMS（扩展的有限状态机）





第四章

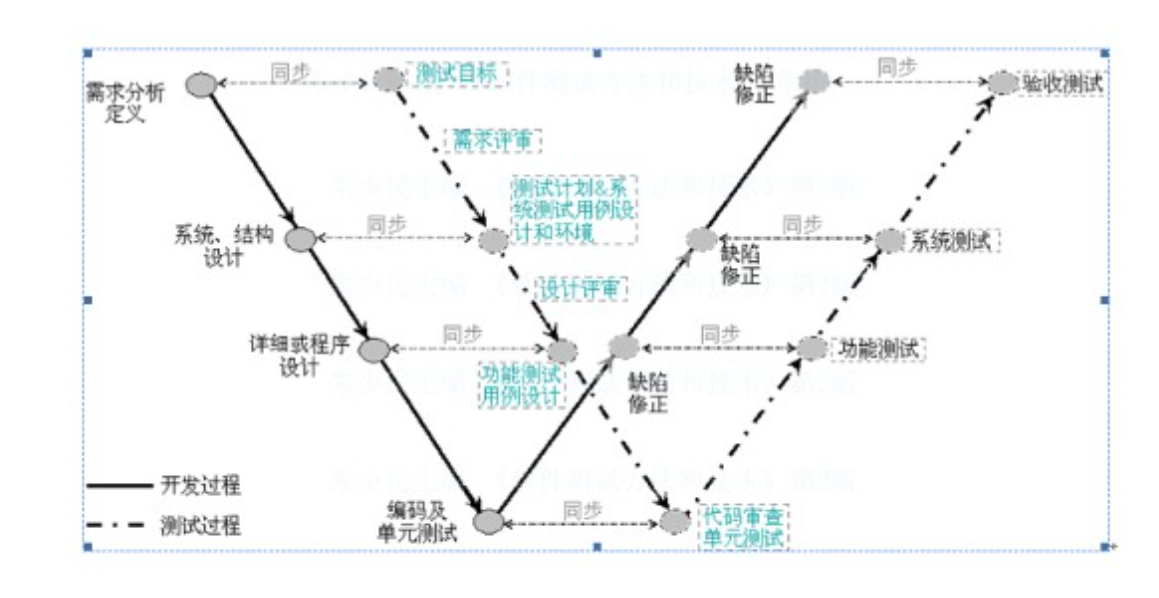
1 测试左移和右移，贯穿全生命周期的测试思想

测试左移：不仅让开发人员做更多的测试，而且需要做需求评审、设计评审，以及验收测试驱动开发（ATDD）

测试右移：在线测试（Test in Production,Tip)，包括在线性能监控与分析、A/B测试和日志分析等

软件测试贯穿于软件开发全过程,不仅可以在第一时间内发现缺陷,降低缺陷带来的成本(劣质成本),而且能有效地预防缺陷的产生,构建更好的软件产品质量。所以,软件测试不再是事后检查,而是缺陷预防和检查的统一,将软件测试扩展到软件质量保证的全过程中,从而大大减少软件缺陷的数量、提高软件质量。更有价值的是,它可以极大地缩短开发周期、降低软件开发的成本。

2 w模型



增加了软件开发阶段中应同步进行的验证和确认活动。

1.测试过程和开发过程是同时开始、同时结束的，两者保持同步关系。

2.测试过程是对开发过程中阶段性成果和最终的产品进行验证的过程。

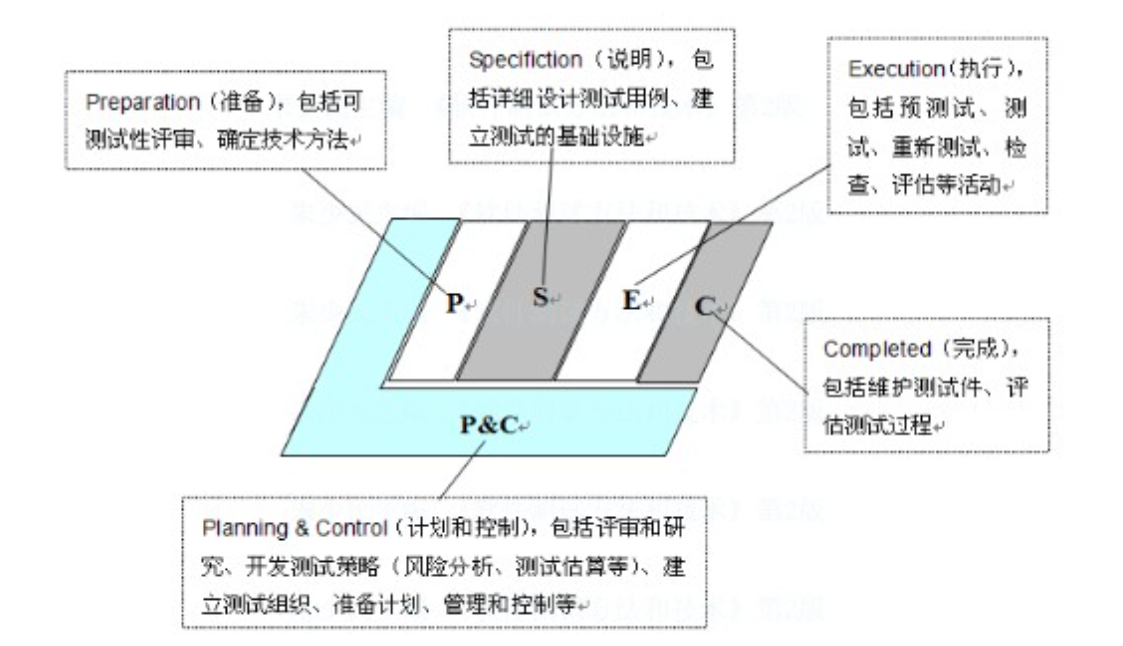
前期测试过程更多依赖开发过程，后期开发过程更多依赖测试过程。

3.测试过程中的⼯作重点和开发⼯作的重点可能是不⼀样的,两者有各⾃的特点，⽆论在资源管理,还是在⻛险管理,两者都存在着差异

3 TMAP定义，几个阶段，模型基石及关系

Test Management Approach，测试管理方法。结构化的、基于风险策略的测试方法体系。目的是能更早地发现缺陷，以最小的成本有效地、彻底地完成测试任务，以减少软件发布后的支持成本。

测试生命周期：计划和控制、准备、说明、执行、完成



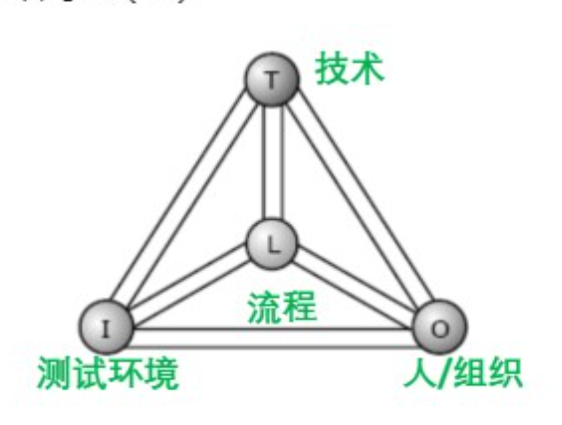
TMap三大基石：（三⼤基⽯⽀持整个⽣命周期 ）

（L）软件开发⽣命周期⼀致的测试活动⽣命周期，描述了在开发的过程中的某些特殊阶段需要的活动。

（O）坚实的组织融合，强调测试⼩组必须融合到项⽬组织中，⽽且每个测试成员都必须被分配 任务和承担责任。

（I）正确的基础设施和⼯具，它说明了为了获得最优化的结果需要适当的基础设施和⼯具。其中“测试环境”必须是稳定的、可控制和有必要通过⼯具的使⽤提⾼测试的有效性。

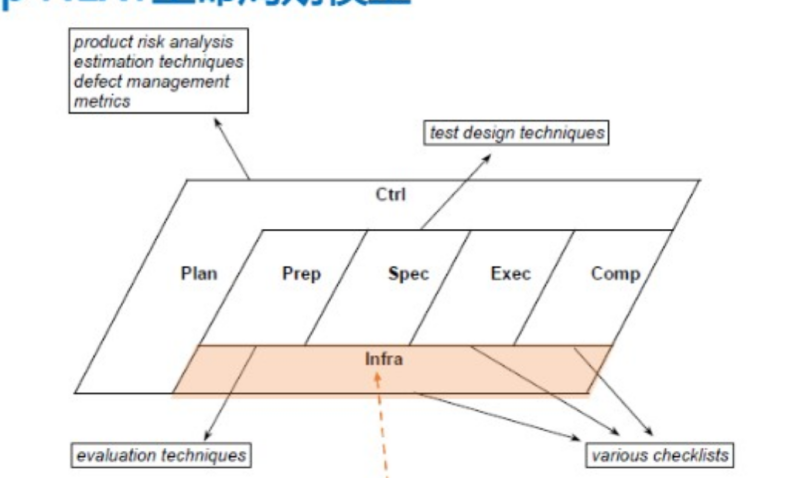
（T）⽀持测试过程的技术，这些技术⽤于定义基于⻛险的测试策略，⽀持有计划的测试过程，研究和审查测试基准，详细说明测试⽤例以及如何提交报告



下表课上考过选择：

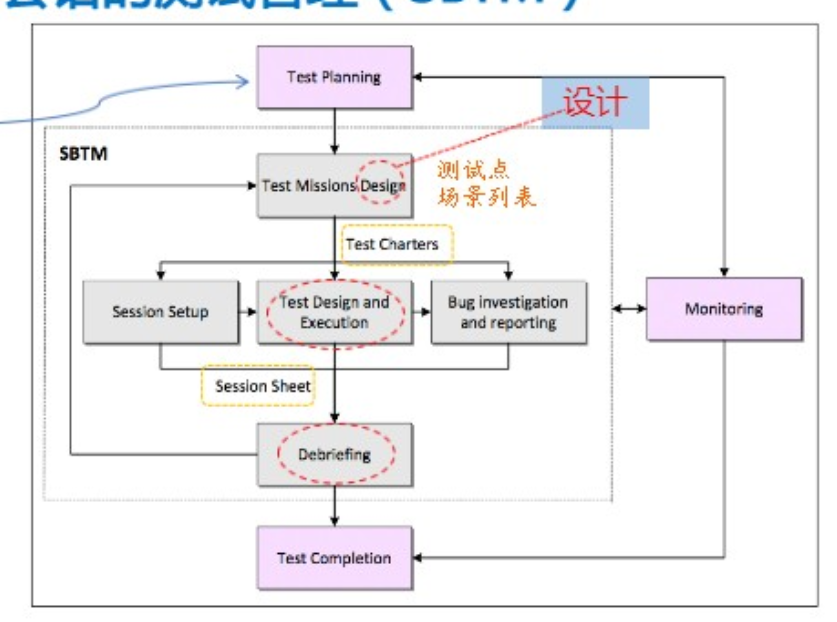


TMap NEXT 基于风险分析，可量化分析



4 SBTM基本要素，结果，原理？

基于会话的测试管理（Session Based Test Management)



Session（会话）是一段不受打扰的测试时间，测试管理的最小单元。

每个Session关联一个特定的、目标明确的测试任务（mission）

Charter（章程/测试指导）：对每个session如何执行进行简要的描述，相当于每个session需要一个简要的计划（提纲）

Session sheet（测试报告）：供第三方（如测试经理）进行检查的材料。最好能被工具扫描、解析。

Debriefing（口头报告）：测试人和leader之间的对话

1. Past(做了哪些测试)
2. Results(测试结果)
3. Obstacles(障碍)
4. Outlook(未来要做哪些测试)
5. Feelings(感觉)

5 测试几个学派的特点

1. 分析流派：认为软件是有逻辑性的，将测试看作计算机科学和数学的一部分，认为测试工作是技术性很强的工作，侧重于使用类似UML工具进行分析和建模（代码覆盖率、结构化测试）【认为测试是严格的和技术性的，在学术界有许多支持者】
2. 质量流派：软件质量需要规范，测试时过程的质量控制、解释项目质量风险的活动，确定开发人员是否遵守规范，测试人员扮演产品质量的守门员角色。【强调过程规范性】
3. 标准流派：把测试看作侧重劣质成本控制并具有可重复标准的、旨在衡量项目进度的工作，测试是对产品需求的确认，每个需求都要得到确认。【：将测试视为衡量进度的一种方法，强调成本和可重复的标准】
4. 上下文驱动流派：认为软件是人创造的，测试所发现的每一个缺陷都和相关利益者密切相关；测试是一种有技巧的心理活动；强调人的能动性和启发是测试思维（ET）【强调人的价值，寻找涉众关心的bug】
5. 敏捷流派：认为软件就是+-持续不断的对话，而测试就是验证开发工作是否完成，强调自动化测试（TDD）【使用测试来快速验证开发是完整的】

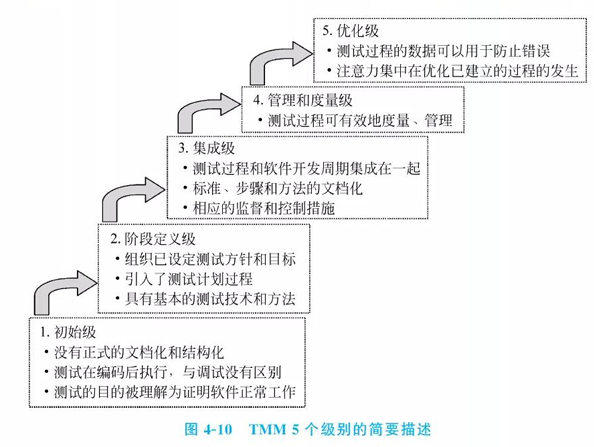
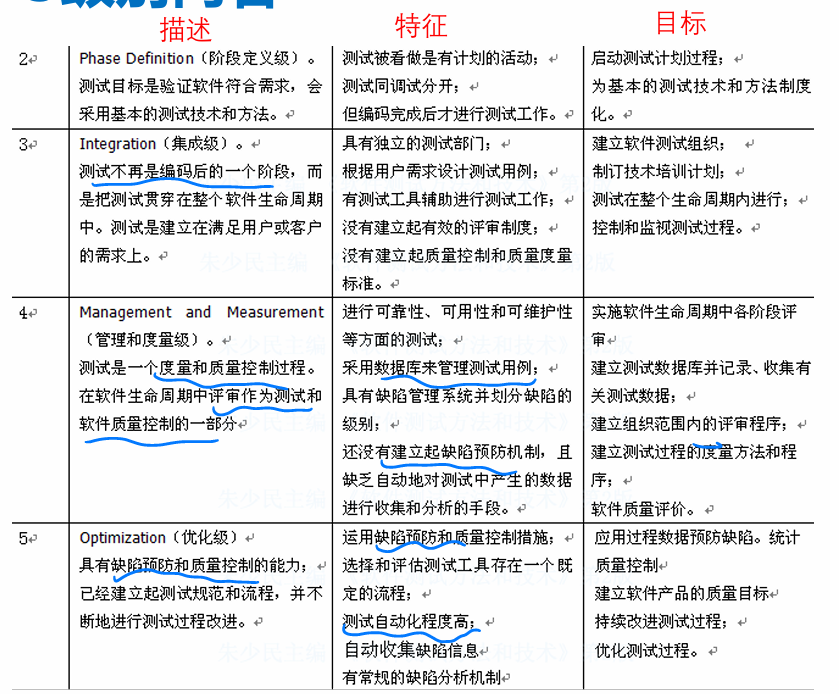
6 TMM，TPI，CTP, STEP定义，特点

都是软件测试能力成熟度模型

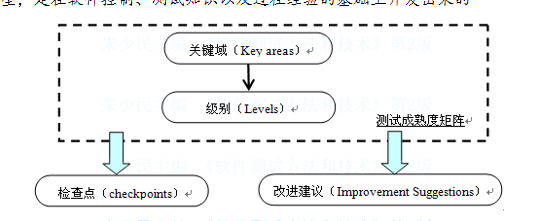
TMM（Testing Maturity Model）:过程能力描述了遵循一个软件测试过程可能达到的预期结果的范围。

TMMi 5个级别的⼀系列测试能⼒成熟度的定义，每个级别的组成包括到期⽬标、到期⼦⽬标活动、任务和职责等。

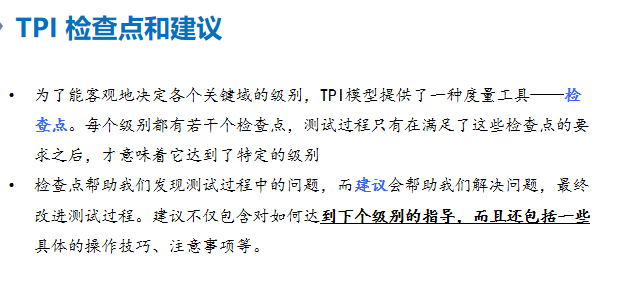
⼀套评价模型，包括⼀个成熟度问卷、评估程序和团队选拔培训指南。

TPI（Testing Process Improvement）：基于连续性表示法的测试过程改进的参考模型，在软件控制、测试知识以及过往经验的基础上开发出来的。 A-D四个级别





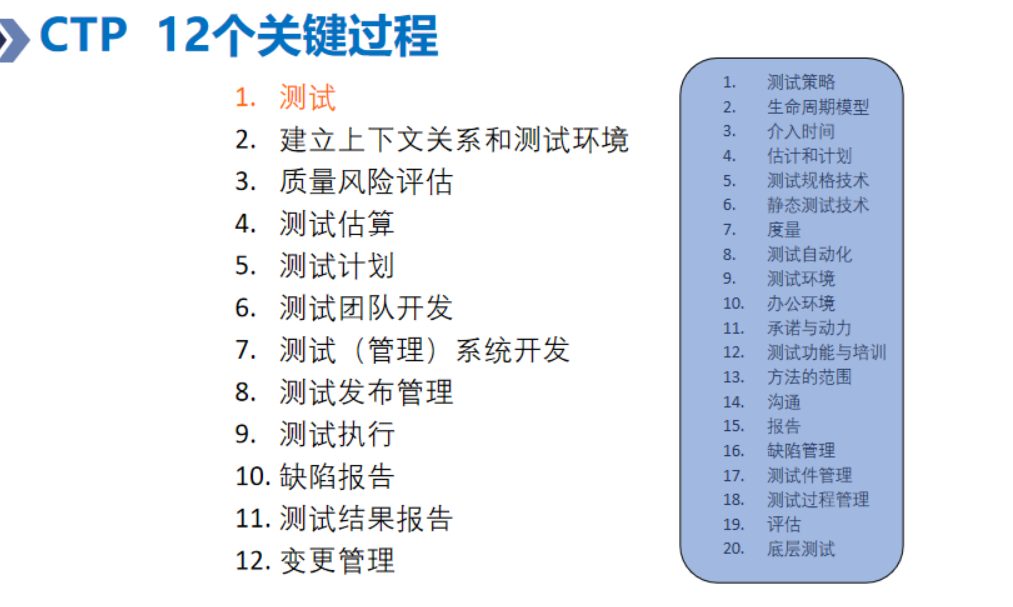




CTP：关键测试过程（Critical Test Process）内容参考模型、上下文相关的方法，并能对模型进行裁剪。

使用CTP的过程改进：始于对现有测试过程的评估，通过评估以识别过程的强弱，并结合组织的需要提供改进的意见。

4P:计划、准备、执行、完善（计划、完善管理工作；准备执行实践工作）



STEP（Systematic Test and Evaluation Process，系统化测试和评估过程），内容参考模型。

1. 基于需求的测试策略
2. 在生命周期初始开始进行测试
3. 测试用作需求和使用模型
4. 由测试件设计导出软件设计（测试驱动开发）
5. 及早发现缺陷或完全的缺陷预防
6. 对缺陷进行系统分析
7. 测试人员和开发人员一起工作

强调度量：

1. 已定义的测试过程使用
2. 客户满意度
3. 不同时期的测试状态
4. 测试需求和风险覆盖
5. 缺陷趋势，包括发现、等级和分类分项数据。
6. 缺陷密度、缺陷移除效率、缺陷发现率
7. 缺陷引进、发现和移除等阶段
8. 测试成本，包括时间、工作量、资金

第五章

1 代码评审的形式及各自特点

代码规范

代码互查（peer review）：

可以采用一些协作工具，例如GitHub PR的diff功能

开发人员走上台介绍其代码

代码走查:

**采用讲解、讨论和模拟运行的方式进行的查找错误的活动。**

会议评审:  
 以会议形式，制定目标、流程和规则

按缺陷检查表（不断完善）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。

|  |  |  |
| --- | --- | --- |
|  | **走 查** | **审 查** |
| **准备** | 通读设计和编码 | 事先准备Spec、程序设计文档、源代码清单、代码缺陷检查表等 |
| **形式** | 非正式会议 | 正式会议 |
| **参加人员** | 开发人员为主 | 项目组成员包括测试人员 |
| **主要技术方法** | 无 | 缺陷检查表 |
| **生成文档** | 会议记录 | 静态分析错误报告 |
| **目标** | 代码标准规范  无逻辑错误 | 代码标准规范  无逻辑错误 |

其他：

关键代码的集体评审

代码评审作为一种文化

借助工具更有效

流程定义

培训

2 单元测试定义，作用，目标

单元测试定义：对软件中的最小可测试单元进行检查和验证

作用：

1. 尽早发现错误
2. 检查代码是否符合设计和规范，有利于将来代码的维护
3. 支持扩展
4. 引导更好的设计
5. 支持变化
6. 避免回归缺陷
7. 保证工作的平稳步调
8. 节省测试时间
9. 规范行为和记录代码

目标：

1. 确保单元模块被正确实现（编码层面），包括功能、性能、安全性
2. 检查输入是否正确传递和得到保护，输出是否正常
3. 内部数据能否保持其完整性，包括变量的正确定义和引用、内存及时释放、全局变量的正确处理和影响最低
4. 代码行、分支覆盖或MC/DC达到要求

3 桩程序，驱动程序

驱动程序:代替上级模块传递测试用例的程序

桩程序：代替下级模块的仿真程序

4 集成测试的概念

将已分别通过测试的单元按设计要求集成起来再进行的测试，以检查这些单元之间的接口是否存在问题，包括接口参数的一致性引用、业务流程端到端的正确性等。

5 单一系统的集成测试的集成模式及优缺点

非渐增式：（先分别测试每个模块，再把所有模块按设计要求放在一起结合成所要的程序，如大棒模式。）

优点：简单直接

缺点：集成中容易出现混乱，因为测试时可能发现一大堆错误，为每个错误定位和纠正很困难，并且在改正一个错误的同时又可能引入新的错误，更难定位错误的原因和位置。

常见模式：

大棒模式（Big Boom）

渐增式：（把下一个要测试的模块同已经测试好的模块结合起来进行测试，测试完以后再把下一个应该测试的模块结合进来测试。）

缺点：流程步骤比较多

优点：程序一段段扩展，每次测试的接口非常有限，错误易于定位和纠正，界面的测试也可做到完全彻底。

自顶向下：

自底向上：

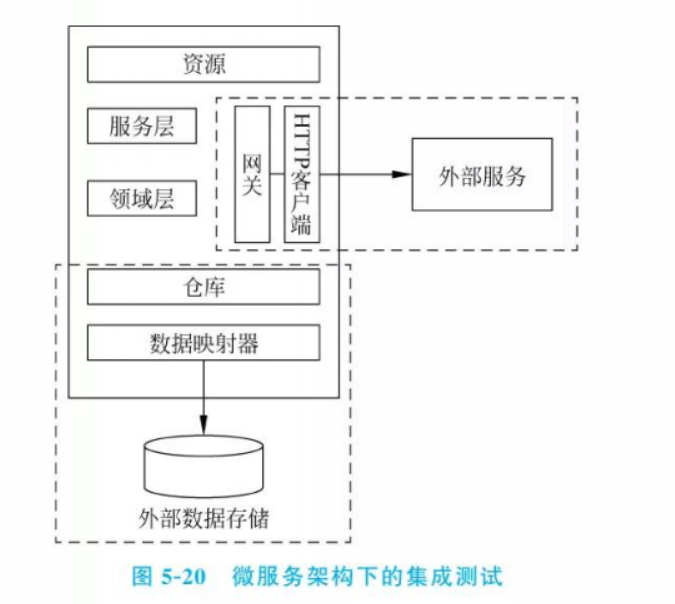
三明治集成：

6 微服务特点，测试目标？测试基本步骤？

微服务特点：

1. 传统的单体应用拆分为多个微服务，形成了多个松耦合的微服务组件模块
2. 原来单个系统内部的API接口调用变成了微服务之间的接口调用
3. 不同的微服务很可能是由不同的开发团队负责开发
4. 包括6部分具体组件：
   1. 资源组件
   2. 服务层
   3. 领域层
   4. 仓库层
   5. 数据映射器
   6. 网关和HTTP客户端

测试目标：验证一个子系统或功能模块和外部组件之间的正常通信（外部组件包括其他微服务或者外部数据存储系统）



测试步骤：

和外部服务：

1. 启动被测微服务和外部微服务的实例
2. 调用被测微服务，该服务会通过外部服务提供的API读取响应数据
3. 检查被测微服务是否能正确解析返回结果

和外部数据存储：

1. 启动外部数据库
2. 连接被测应用到数据库
3. 调用被测微服务，该服务会往数据库写数据
4. 读取数据库，查看期望的数据是否被写到了数据库里

7 集成测试的目标

[集成测试](http://zhidao.baidu.com/search?word=%E9%9B%86%E6%88%90%E6%B5%8B%E8%AF%95&fr=qb_search_exp&ie=utf8" \t "_blank)，也叫[组装测试](http://zhidao.baidu.com/search?word=%E7%BB%84%E8%A3%85%E6%B5%8B%E8%AF%95&fr=qb_search_exp&ie=utf8" \t "_blank)或联合测试。在[单元测试](http://zhidao.baidu.com/search?word=%E5%8D%95%E5%85%83%E6%B5%8B%E8%AF%95&fr=qb_search_exp&ie=utf8" \t "_blank)的基础上，将所有模块按照设计要求（如根据[结构图](http://zhidao.baidu.com/search?word=%E7%BB%93%E6%9E%84%E5%9B%BE&fr=qb_search_exp&ie=utf8" \t "_blank)〕组装成为子系统或系统，进行[集成测试](http://zhidao.baidu.com/search?word=%E9%9B%86%E6%88%90%E6%B5%8B%E8%AF%95&fr=qb_search_exp&ie=utf8" \t "_blank)。实践表明，一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。程序在某些局部反映不出来的问题，在全局上很可能暴露出来，影响功能的实现。

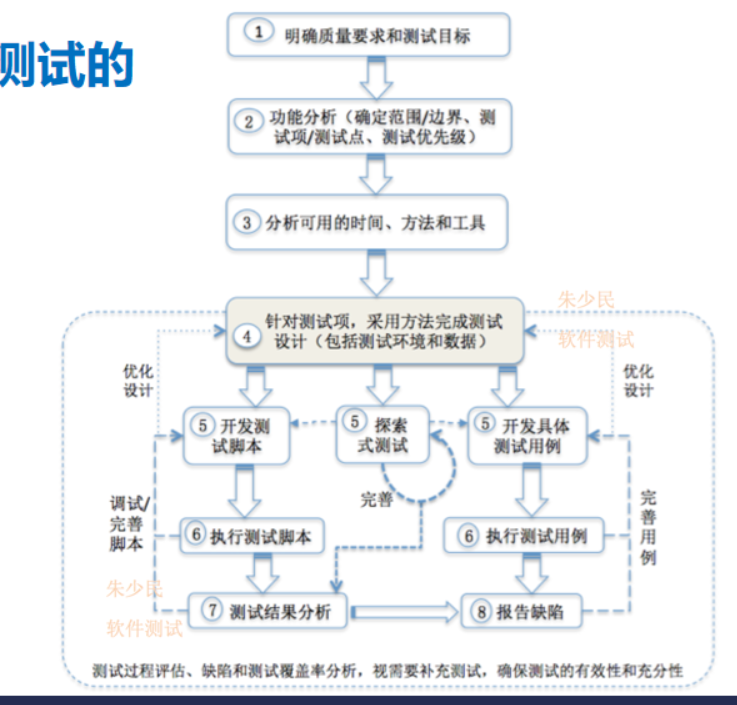
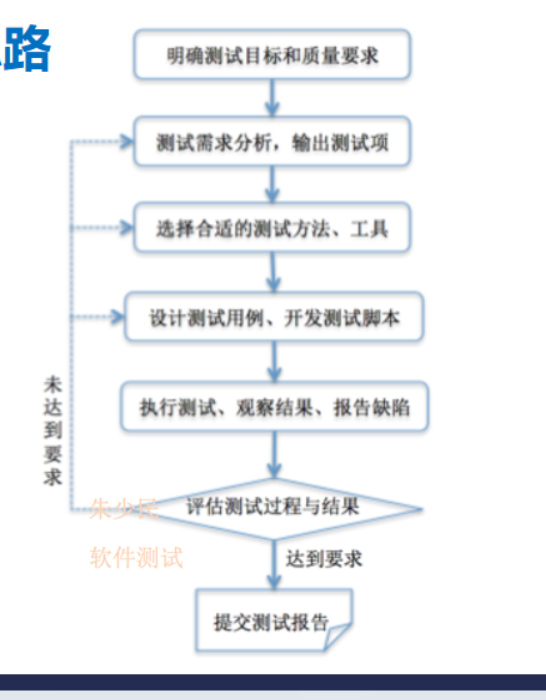
目标在于检验与软件设计相关的程序结构问题。如数据穿过接口时可能丢失；一个模块与另一个模块可能有由于疏忽的问题而造成有害影响； 把子功能组合起来可能不产生预期的主功能；个别看起来是可以接受的误差可能积累到不能接受的程度；全程数据结构可能有错误等。

8 集成测试的依据

集成测试的测试依据；概要设计书，详细设计说明书，主要是概要设计说明书

第六章

1 功能测试的基本思路



2 回归测试需要解决的问题。回归测试的策略和方法

回归缺陷：原来正常工作的功能，没有发生需求变化，而由于受其它改动影响而产生的问题。

回归测试就是为了发现回归缺陷而进行的测试。

回归测试的目的：在程序有修改的情况下保证原有功能正常的一种测试策略和方法。

回归测试的策略和方法：

1. 再测试全部用例
2. 基于风险选择测试（选择容易发生回归缺陷的部分）
3. 基于操作剖面选择测试（软件操作剖面：软件功能点被用户的使用程度）
4. 再测试修改的部分（相当于没进行回归测试，效率高，风险大）

第七章

1 性能测试目标

1. 获取系统性能某些指标数据
2. 为了验证系统是否达到用户提出的性能指标
3. 发现系统中存在的性能瓶颈，优化系统的性能

2 什么是性能测试？

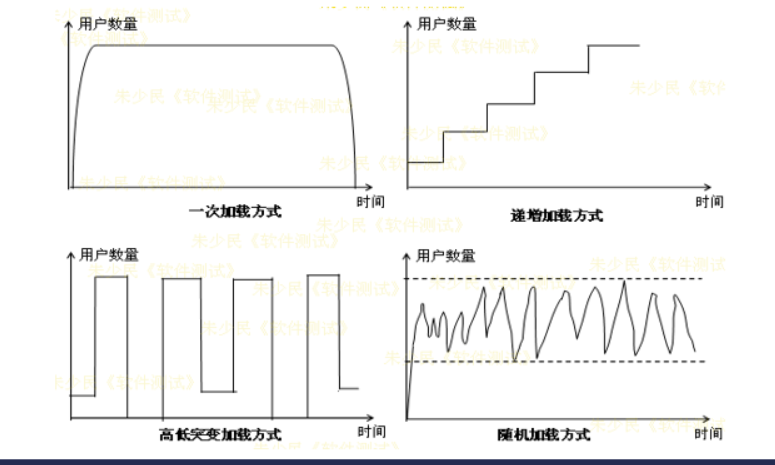
为了发现系统性能问题或获取系统性能相关指标而进行的测试。一般在**真实环境、特定负载**条件下，通过工具模拟实际软件系统的运行及其操作，同时监控性能各项指标，最后对测试结果进行分析以确定系统的性能状况。

3 系统性能表现

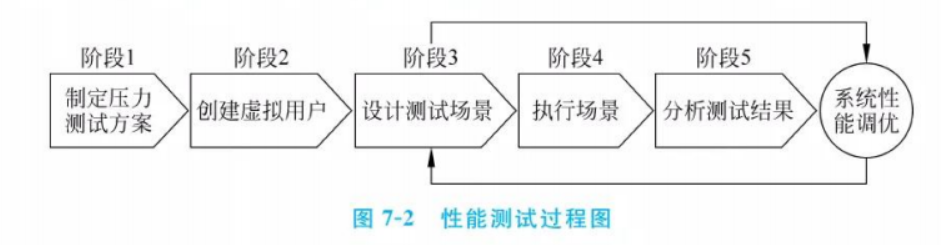
1. 请求响应时间
2. 事务响应时间
3. 数据吞吐量（web服务器成功处理的HTTP页面或HTTP请求数量）
4. RPS：每次请求发送的数据量，Request Per Load
5. 并发连接数：Simultaneous Connections
6. 思考时间：Thinking Time，用户发出请求之间的间隔时间

4 按照测试目的分，性能测试类型

1. 性能基准测试：在系统标准配置下获得有关的性能指标数据，作为将来性能改进的基线。
2. 性能验证测试：验证系统是否达到事先已定义的系统性能指标、能否满足系统的性能需求
3. 性能规划测试：在多种特定的环境下，获得不同配置的系统性能指标，从而决定系统部署的软硬件配置选型。
4. 容量测试：系统的容量可以看作是系统性能指标之一。
5. 压力测试：模拟实际应用的软硬件环境及用户使用过程的高负载、异常负载、超长时间运行，从而加速系统崩溃，以检查程序对异常情况的抵抗能力，找出性能瓶颈、不稳定等问题。
6. 渗入测试(soak test)：通过长时间运行，使问题逐渐渗透出来，从而发现内存泄漏、垃圾回收（GC）或系统的其他问题，以检验系统的健壮性。
7. 峰谷测试(peek-rest test):采用高低突变加载方式进行，先加载到高水平的负载，然后急剧降低负载，稍微平息一段时间，再加载到高水平的负载，重复这样过程，容易发现问题的蛛丝马迹，最终找到问题的根源。



5 性能测试的基本过程



1. 确定性能测试需求
2. 根据测试需求，选择测试工具和开发相应的测试脚本。
3. 建立性能测试负载模型（确定并发用户数量、RPS、思考时间、加载方式、持续加载时间）
4. 执行性能测试
5. 提交性能测试报告（包括性能测试方法、负载模型和实际执行的性能测试、性能测试结果及其分析）

6 什么是安全性测试

全面检验软件在需求规格说明中规定的防止危险状态措施的有效性和在每一个危险状态下的反应，对软件设计中用于提高安全性的结构、算法、容错、冗余、中断处理等方案进行针对性测试，并对安全性关键的软件单元和软件部件单独进行加强的测试，以确认其满足安全性需求。

7 渗透测试实施策略

1. 全程监控
2. 择要监控
3. 主机监控
4. 指定攻击源
5. 对关键系统，采用对目标的副本进行渗透测试

8 软件安全性测试有哪两种？有什么关系和区别？

1. 安全功能性测试（Security Functional Testing）：数据机密性、完整性、可用性、不可否认性、身份认证、授权、访问控制、审计跟踪、委托、隐私保护、安全管理等。
2. 安全漏洞测试（Security Vulnerability Testing）：从攻击者的角度，以发现软件的安全漏洞为目的。安全漏洞是指系统在设计、实现、操作、管理上可能存在的可被利用的缺陷或弱点。

9 安全性测试的任务

1. 全面检验软件在需求规格说明中规定的防止危险状态措施的有效性和在每一个危险状态下的反应
2. 对软件设计中用于提高安全性的结构、算法、容错、冗余、中断处理等方案进行针对性测试
3. 在异常条件下测试软件，以表明不会因可能的单个或多个输入错误而导致不安全状态。
4. 对安全性关键的软件单元、组件，单独进行加强的测试，以确认其满足安全性需求

10 安全性测试方法按内外部分为哪两种？

基于威胁的方法

从软件外部考察其安全性，识别软件面临的安全威胁并测试其是否能够发生

基于漏洞的方法

从软件内部考虑其安全性，识别软件的安全漏洞，如借助特定的漏洞扫描工具

11 什么是XSS攻击和sql注入攻击，如何进行测试和防范

XSS：允许恶意用户将代码植入到“供其他用户使用的web页面”中。

测试：可以通过尝试在输入字段中插入恶意JavaScript代码，然后查看这些代码是否被执行来测试XSS漏洞。专业的安全测试人员可能会使用自动化工具来检测潜在的XSS漏洞

防范：

* + **输入验证与净化**：对所有用户输入进行严格的验证和过滤，使用白名单方法允许特定格式和字符2。
  + **输出编码**：在将用户输入的数据展示到页面上时，进行HTML编码或其他形式的编码，以防止恶意脚本执行。
  + **使用安全框架**：现代Web开发框架通常提供了防止XSS的机制，应确保使用最新版本的框架

SQL注入：根据SQL语句编写的规则，附加一个永远为真的条件，使系统中某个认证条件总是成立，从而欺骗系统、躲过认证，进而侵入系统。

测试：测试SQL注入漏洞通常涉及在应用程序的输入字段中尝试输入特殊的SQL语法，如单引号(')、分号(;)等，看是否能够改变数据库的正常行为或获取额外的数据

**防范**：

* **参数化查询**：使用参数化查询或预编译语句来避免将用户输入直接嵌入SQL查询中。
* **输入验证和过滤**：对所有输入数据进行严格的验证和过滤，确保只接受预期的输入类型，并转义特殊字符。
* **最小权限原则**：确保应用程序以最小的权限运行，减少攻击者可能获得的控制范围。
* **使用安全编码规范**：推行严格的安全编码规范，并通过自动化测试工具检测潜在漏洞。

12 web安全性测试可从哪些方法开展

1. 数据加密
2. 登录或身份验证
3. 输入验证
4. SQL注入漏洞检测
5. 超时限制验证
6. 目录安全性
7. 操作留痕（日志文件）

13 什么是软件可靠性？可从哪几个指标度量？各自的定义

软件可靠性是软件系统在规定的时间内及规定的环境条件下，完成规定功能的能力。

度量指标：

1. 成熟性：为避免软件自身存在的故障而导致软件失效的能力
2. 容错性：在出现故障或违反规定接口情况下软件维持规定性能级别的能力
3. 易恢复性：在失效发生的情况下软件重建性能级别恢复直接受影响的数据的能力

14 容错测试的要点？

容错测试：一种对抗性的测试过程。

1. 在这种测试中，通过各种手段让软件强制性地发生故障，或将把应用程序或系统置于（模拟的）异常条件下，以产生故障。例如设备输入/输出故障或无效的指针或关键字等。
2. 然后调用恢复进程并监测、检查应用程序和系统，核实系统和数据已得到了恢复。
3. 对于自动恢复需验证重新初始化、检查点、数据恢复和重新启动等机制地正确性。
4. 对于人工干预地恢复系统，还需评估平均修复时间，确定其是否在可接受地范围内。

15 什么是A/B测试？有什么特点

A/B测试（ABTest） 是将用户分成不同的组，同时在线试验产品的不同版本，通过用户反馈的真实数据来确定哪一个方案更好

特点：

1. 先验性：采用流量分割与小流量测试的方式，先让线上部分小流量用户使用以验证设计，再根据数据反馈来推广到全流量，减少产品损失
2. 并行性：同时运行两个或两个以上版本的试验完成对比分析，而且保证每个版本所处的环境一致的，避免流程周期长的问题，节省验证时间
3. 科学性：基于统计的数据来做出决策，避免主观或经验的错误决策

第八章

1 软件本地化，国际化，全球化，相互关系

1. I18N(internationalization):软件国际化,为保证所开发的软件能适应全球市场的本地化工作而不需要对程序做任何系统性或结构性变化的特性，这种特性通过特定的系统设计、程序设计、编码方法实现。任何实现I18N的产品，在对其进行本地化工作时，只要进行一些配置和翻译工作，而不需要修改软件的程序代码。
2. L10N(localization):软件本地化,将一款软件产品按特定国家或语言市场的需要进行全面定制的过程。（L10N是将一个软件产品按特定国家/地区或语言市场的需要进行加工，使之满足特定市场上的用户对语言和文化的特殊要求的软件生产活动。）
3. G11N(globalization)：软件全球化，是一个概念化产品的过程，基于全球市场考虑，以便一个产品只做较小的改动就可以在世界各地出售。全球化可以看作I18N和L10N的结合。**G11N (**globalization**)= I18N + L10N**

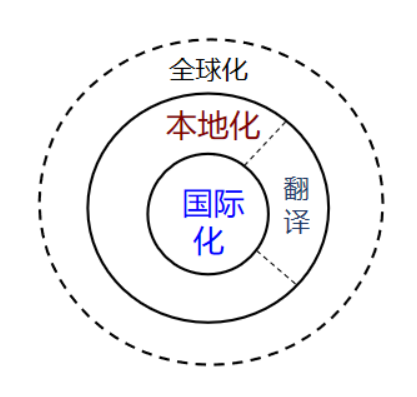
关系：

I18N是L10N的基础和前提，为L10N做准备

L10N是I18N向特定本地语言环境的转换

I18N 是软件产品源语言开发的一部分，属于Engineering

L10N 可以独立于Engineering，可由第三方完成



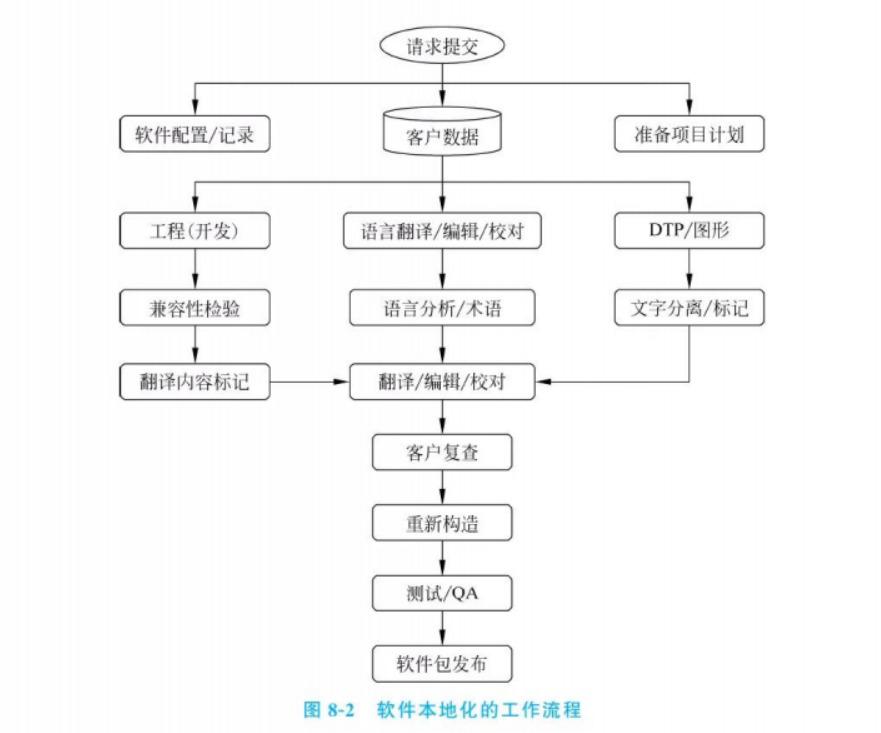
2 unicode与utf-x关系，特点

Unicode是一个国际标准，采用双字节对字符进行编码，提供了在世界主要语言中通用的字符，所以也称为基本多文种平面。Unicode以明确的方式表述文本数据，简化了混合平台环境中的数据共享。Unicode简称为UCS，常用的是UCS-2双字节编码体系。

Utf-x是Unicode的编码体系，如UTF-8/16/32，UTF-8解决了unicode在不同的计算机之间的传输保存，使得双字节的Unicode编码能够在现存的处理单字节的系统上正确传输。

3 软件本地化基本步骤

1. 建立一个配置管理体系，跟踪目标语言各个版本的源代码
2. 从源语言代码中分离资源文件或提取需要本地化的文件
3. 把分离或提取的文本、图片等翻译成目标语言
4. 把翻译好的文本、图片重新插入目标语言的源代码版本中
5. 如果需要，编译目标语言的源代码
6. 测试翻译后的软件，调整UI以适应翻译后的文本
7. 测试本地化后的软件，确保格式和内容都正确



4 本地化测试主要有哪些工作

1. 功能性测试：所有基本功能、安装、升级等测试
2. 翻译测试，包括语言完整性、术语准确性等检查
3. 可用性测试：包括用户界面、度量衡和时区等适合当地的要求
4. 兼容性测试：包括软硬件本身、第三方软件兼容性等的测试
5. 文化、宗教、喜好等适用性测试
6. 手册验证，包括联机文件、在线帮助、PDF文件等测试

5 软件本地化测试完整路线

## 第九章 测试自动化及框架

1 自动化测试与测试自动化

自动化测试（Automated Test）：相对于手工测试（Manual Test）存在的概念，由手工逐个地运行测试用例地操作过程被测试工具或系统自动执行地过程所代替，包括输入数据自动生成、结果地验证、自动发送测试报告等。主要通过所开发的软件测试工具、脚本等来实现，具有良好的可操作性、可重复性和高效率等特点。

测试自动化：是软件测试中提高测试效率、覆盖率和可靠性的重要手段，测试自动化是软件测试不可分割的一部分。

2 如何理解测试自动化

测试自动化是软件测试中提高测试效率、覆盖率和可靠性的重要手段。测试自动化是软件测试不可分割的一部分，意味着测试全过程的自动化和测试管理工作的自动化。

3 测试自动化实现的原理，几种技术

原理：

·代码分析: 类似于高级编译系统，在工具中定义类/对象/函数/变量等定义规则、语法规则等，在分析时对代码进行语法扫描，找出不符合编码规范的地方。

·捕获和回放: 代码分析是一种白盒测试的自动化方法，捕获和回放则是一种黑盒测试的自动化方法。

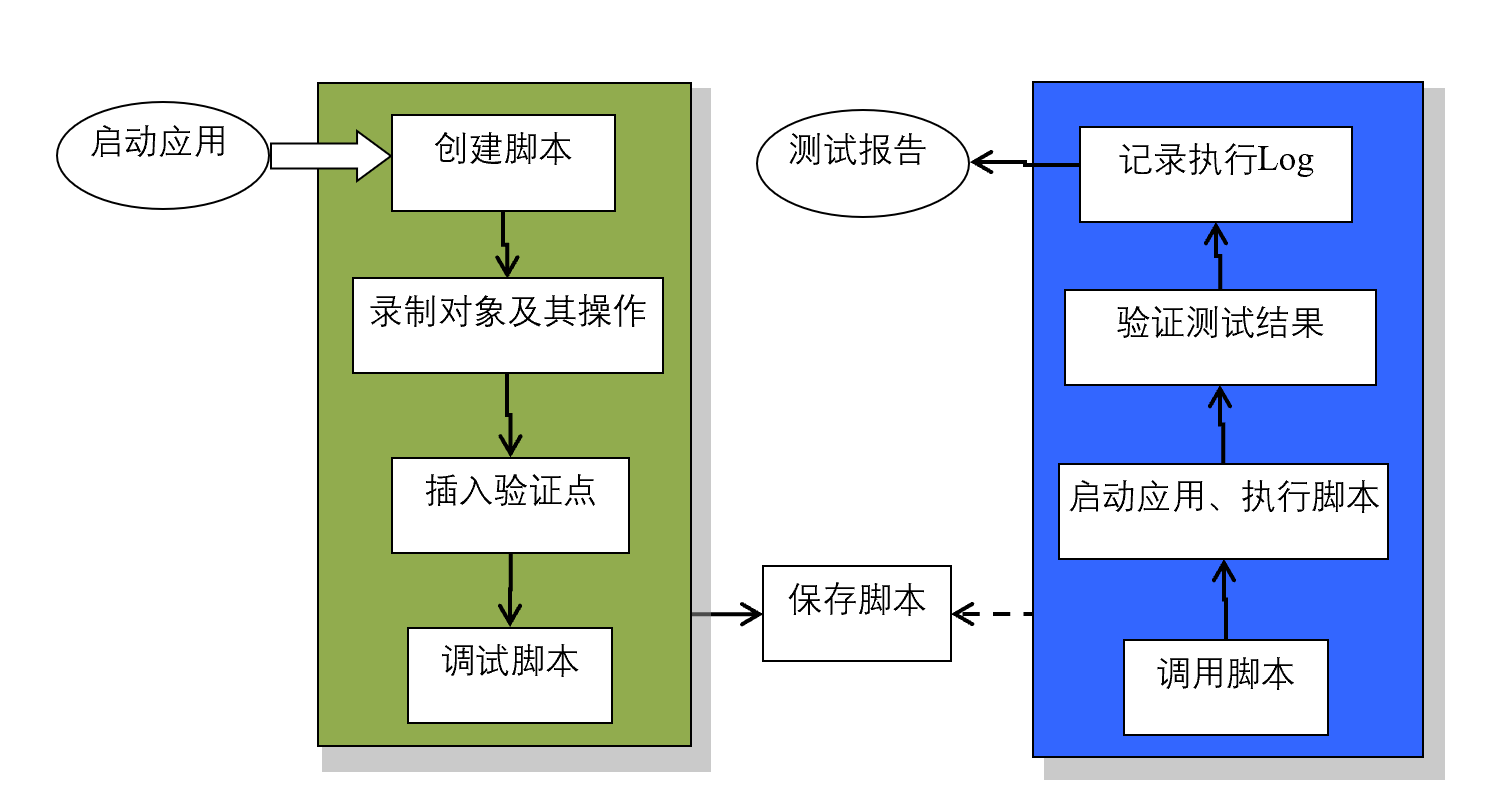
·直接编写脚本来操作、控制、验证对象：包括对象识别、脚本技术、对运行结果进行比较

实现基础：可以通过特定的程序（脚本、指令）对软件应用的代码进行测试。

测试自动化包括4种类型：

1. UI自动化测试（动态测试）
   1. 实现原理：对象识别（如按照屏幕的实际像素坐标定位，寻找UI控件/控件识别、图像识别算法、OCR技术等）
2. API自动化测试（动态测试）
   1. 实现原理：脚本测试+接口调用
3. 自动化单元测试（动态测试）
4. 自动化代码分析（静态测试）
   1. 实现原理：代码静态检测工具（编译器、Lint、IDE的代码检查）

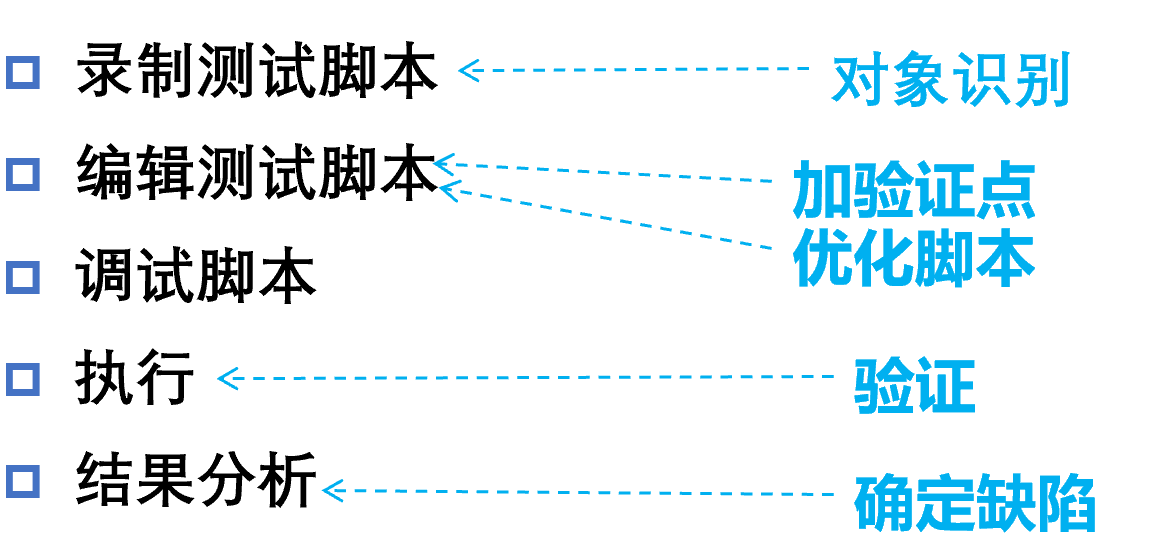
4 自动化测试的流程



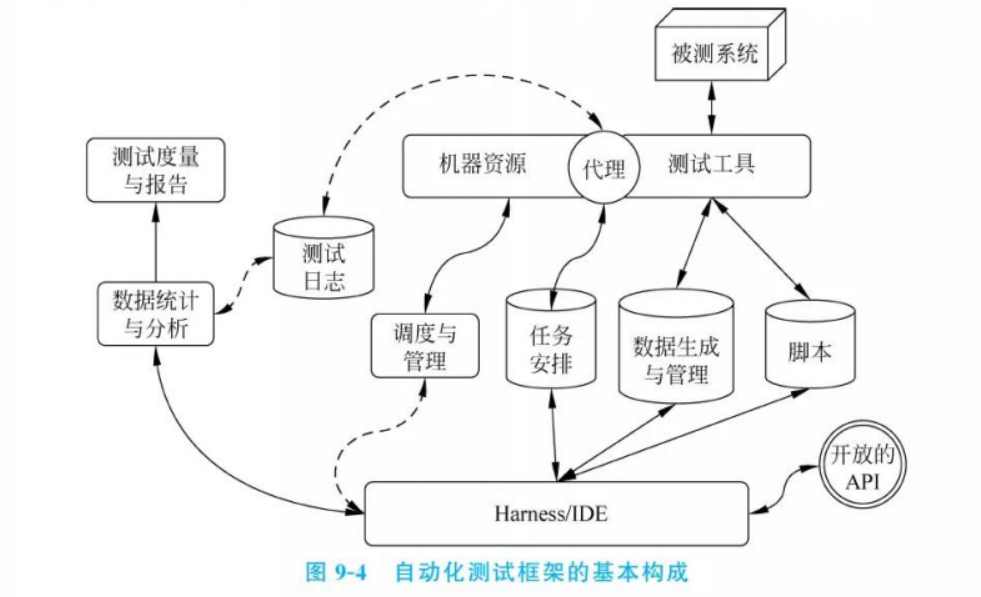
5 几种脚本技术

1. 线性脚本：录制手工执行的测试用例得到的脚本，包括击键、移动、输入数据
2. 结构化脚本：类似于结构化程序设计，具有各种逻辑结构(选择、分支、循环迭代等），且具有函数调用功能。
3. 数据驱动脚本：将测试脚本和数据分离开来，测试输入数据存储在独立的数据文件中。一个测试用例对应一种输入组合。
4. 关键字驱动脚本：数据驱动脚本的逻辑扩张，封装了各种基本的操作，由相应的函数实现。开发脚本时，可以使用已经定义好的关键字，提升编写测试脚本的效率，并且便于维护。

6 自动化功能测试基本构成



7 TA框架的构成及各部分特点



1. Harness/IDE：TA框架核心，其他TA框架的组成部分都能作为插件与之集成
2. TA脚本管理：公共脚本库、项目脚本库
3. 测试资源管理：增加、删除配置相应的测试设备（软硬件部分），根据他们的使用状态来分配测试资源
4. 测试数据管理：测试数据的自动生成、存储、备份和恢复等。
5. 开放的接口：提供给其他CI环境、dev环境或其他测试环境的继承接口
6. 代理：负责Harness与工具的通信，控制测试工具的运行
7. 任务安排：安排和提交定时任务、事件触发任务等，以便实现无人值守的自动化测试执行
8. 数据统计分析：针对测试结果（含测试工具运行时产生的日志），生成可读性良好的测试报告（例如HTML格式的测试结果）

