目录

[3.产品质量的内容，内部/外部/使用质量 2](#_Toc169285653)

[8.测试工作的流程 3](#_Toc169285654)

[6 功能图法，EFMS 5](#_Toc169285655)

[6.TMM，TPI，CTP, STEP定义，特点 9](#_Toc169285656)

[STEP（Systematic Test and Evaluation Process，系统化测试和评估过程），内容参考模型。 10](#_Toc169285657)

**《软件测试技术》复习大纲**

**第一章**

1、软件测试学科的发展

1957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，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，以质量**评估**为导向，测试是提供产品的评估和质量度量

1988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。

调试→证明（正向思维）→破坏（逆向思维）→评估→预防

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环节，试图破坏系统，直至找不出问题。

3、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**。要求软件测试尽早开展工作，发现缺陷越早，返工的工作量越小，造成的损失越小。测试成本小于缺陷造成的损失，测试才有意义。

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，或者与产品设计书的不一致，不能满足用户的需求

从**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：**

①功能/特性没有实现或部分实现

②设计不合理

③实际结果和预期结果不一致

④运行出错（包括运行中断、系统崩溃、界面混乱）

⑤数据结果不正确、精度不够

⑥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**预期输出**进行比较，从而判断是否有缺陷。

（作比较的对象：①需求规格说明书、②其他需求设计文档、③竞争对手的产品)

2.软件缺陷产生的**原因**有哪些？

①技术问题：算法错误、计算和精度问题、接口参数传递不匹配

②团队工作：沟通不充分、有误解

③软件本身：文档错误、用户使用时间/场合不协调，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**满足规定或隐含要求、能力要求**有关的特征与特征总和。

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：**（功用靠性容 伸维兼扩展）**

①内部质量：涉及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**。

②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**。

③使用质量：是指**客户在使用软件时的感受和评价**。**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、免于风险程度、上下文覆盖

6.软件测试不同层次测试的对象和任务（单元、集成、系统、验证）

背景是V模型，以下自底向上。

①单元测试：

对象：**组件/模块/类/函数**

任务：组件功能、健壮性、效率

②集成测试：

对象：单元之间的**接口**

任务：组件之间的**接口**

③系统测试：

对象：由单元构成的**系统**

任务：系统功能、安全性、健壮性、效率

④验证测试：

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？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**的评审以及对代码的静态分析。

评审的主要形式：**互相评审、走查、会议评审**

△产品评审：

评审对象：**需求和设计文档、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

①评审软件需求/分析测试需求

②制定测试目标和测试需求验收标准

③编写测试计划

④设计和开发测试场景、环境、数据、用例和脚本

⑤执行测试并记录缺陷报告，进行缺陷跟踪

⑥记录测试日志和测试执行结果

⑦评估软件质量

⑧撰写测试报告

9.软件测试的工作范畴

1. 测试需求分析:

- 明确测试范围和优先级

- 确定需要完成的测试任务

2. 软件测试策略:

- 确定测试方式(手工/自动化、静态/动态等)

- 选择测试方法(黑盒/白盒、基于数据流/控制流等)

- 确定测试过程(先后顺序)

3. 测试计划:

- 定义目标、范围、优先级

- 识别风险并制定应对措施

- 安排进度并配置资源

- 制定跟踪和控制机制

4. 测试设计:

- 设计总体测试方案和结构

- 设计详细的测试用例

5. 测试执行:

- 手工执行测试

- 自动化执行测试

6. 测试结果和过程评估:

- 分析测试覆盖率和缺陷情况

- 评估测试计划的执行情况和效果

第三章

1、等价类（黑盒）

有效等价类、无效等价类

2、边界值法

3、判定表，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（白盒）

按照测试用例覆盖路径分类

①语句覆盖：程序中**每个可执行语句至少**被执行一次（较弱很难检测出其他错误）

②判定(分支)覆盖（DC）：设计足够多的测试用例，保证程序中**每个判断的取真分支和取假分支**至少经历一次 如两个判断，有真有假，用两个测试用例满足真真、假假（真假、假真）即可

③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

④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。

⑤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。

⑥MC/DC覆盖：

1. **一个判定中的每个条件独立地对结果产生影响**

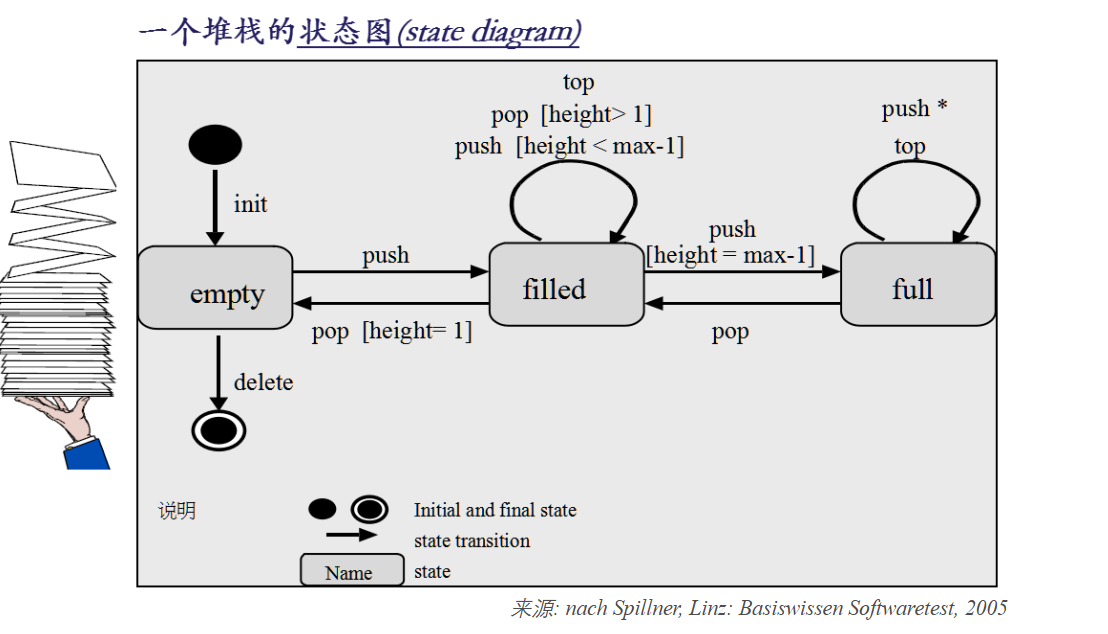
5 路径覆盖法（BPC）

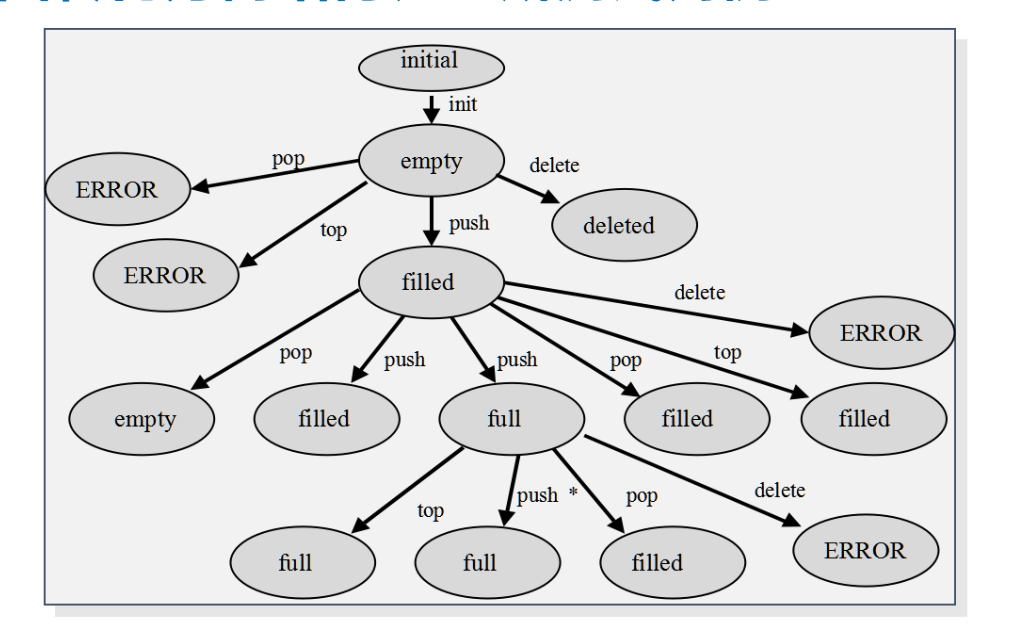
圈复杂度V(G) = 区域数量 = 连线数量-节点数量+2 = 简单可预测节点数量+1

# 6 功能图法，EFMS

功能图：状态迁移图和逻辑功能模型构成

EFSM（扩展的有限状态机）





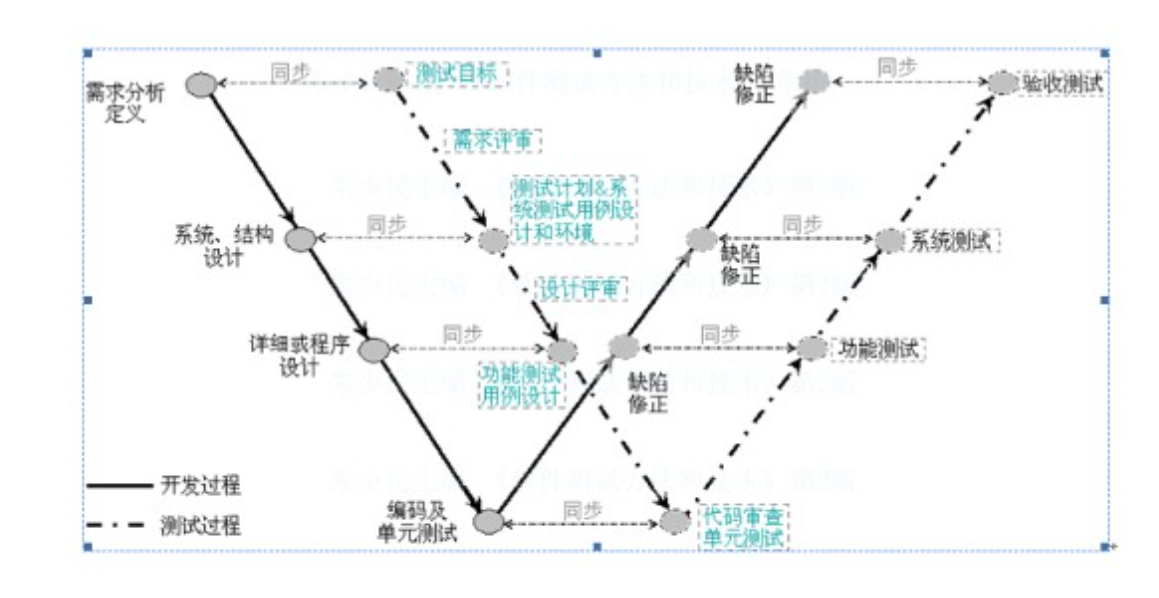
第四章

1 测试左移和右移，贯穿全生命周期的测试思想

测试左移：**让开发人员**做更多的测试，以及需求评审、设计评审、验收测试和驱动开发

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

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。**

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。

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。

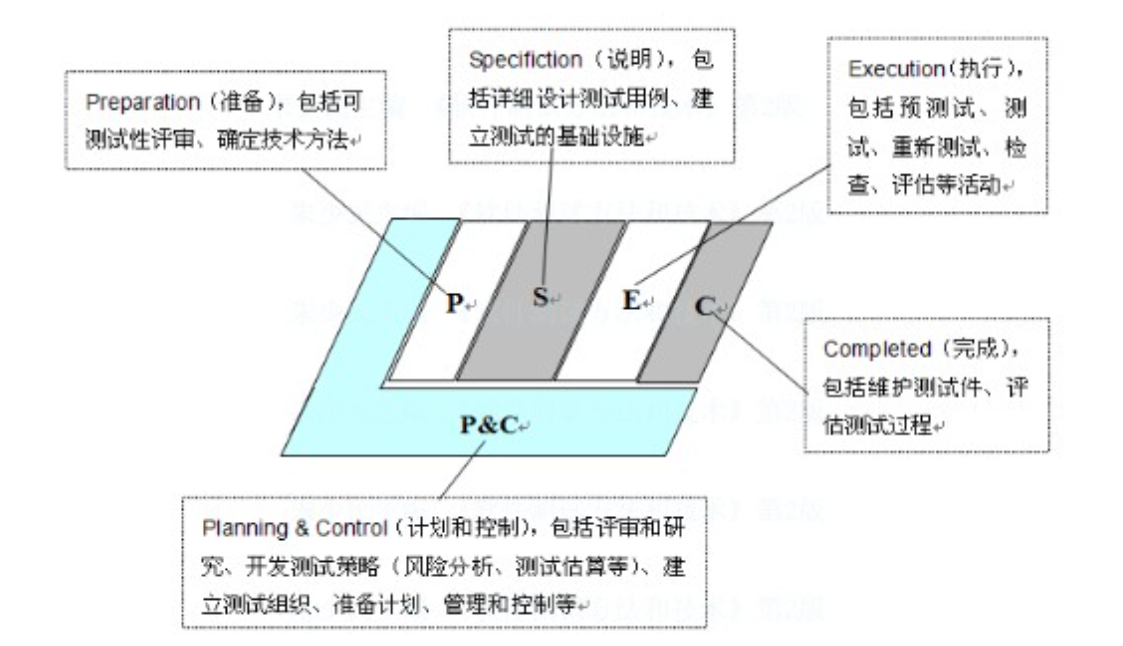
**前期测试过程更多依赖开发过程，后期开发过程更多依赖测试过程。**

3 TMAP定义，几个阶段，模型基石及关系

TMAp测试管理方法：结构化的、基于风险策略的测试方法体系。

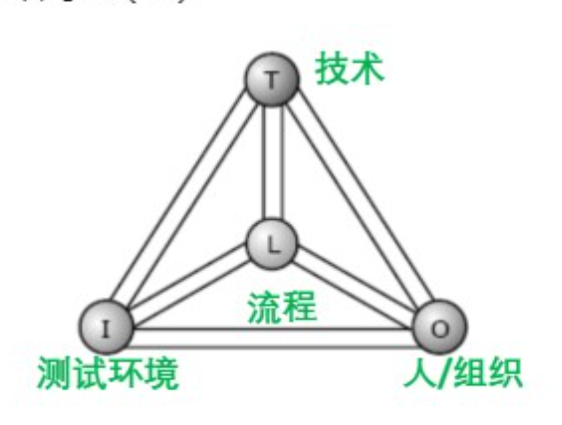
目的是能**更早地发现缺陷**，以最小的成本完成测试任务，减少软件发布后的支持和维护成本。

测试生命周期：计划和控制、准备、说明、执行、完成



TMap三大基石：

1. 与软件开发周期一致的测试活动生命周期（L）
2. 坚实的组织融合（O）
3. 正确的基础设施和工具（I）
4. 可用的技术（T）





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，是测试管理的最小单元。

每个会话关联一个特定的、目标明确的测试任务

△Charter备忘录：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.测试五个学派的特点

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
2. 质量流派：软件质量需要规范，确定开发人员是否遵守规范，扮演产品质量的守门员角色。
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。
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）
5. 敏捷流派：认为软件就似乎持续不断的对话，而测试就是验证开发工作是否完成，强调自动化测试（TDD）

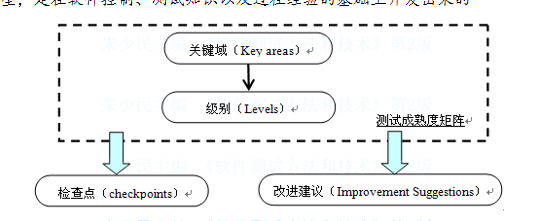
# 6.TMM，TPI，CTP, STEP定义，特点

都是软件测试能力成熟度模型

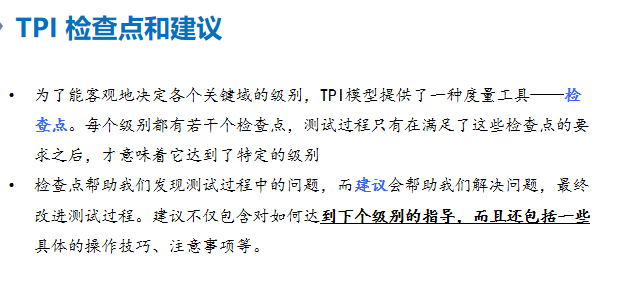
TMM:过程能力描述了遵循一个软件测试过程可能达到的预期结果的范围。

1. 初始级
2. 管理级
3. 定义级
4. 测量级
5. 优化级

TPI：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。



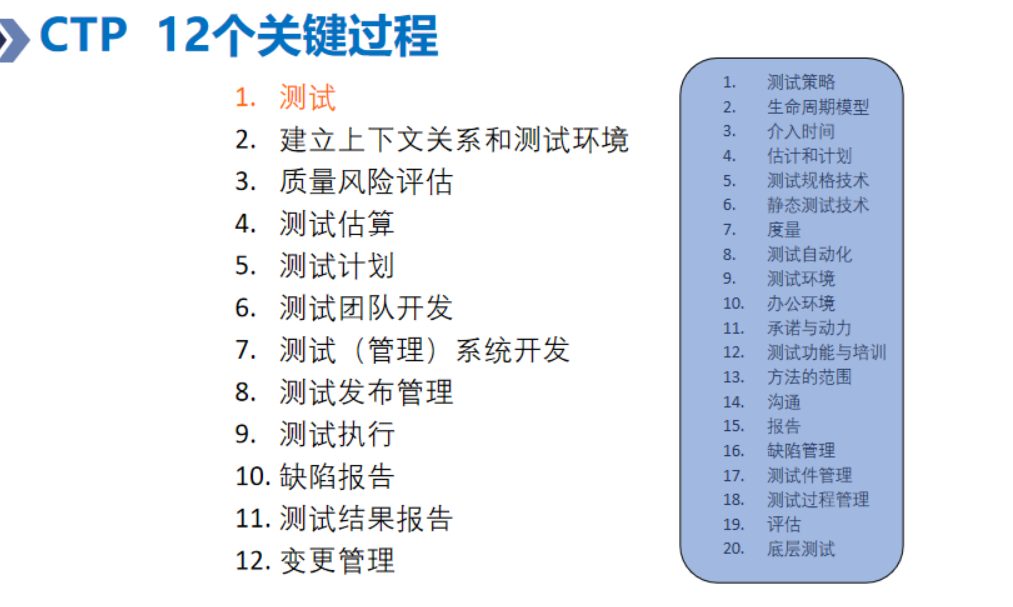




CTP：关键测试过程（Critical Test Process）内容参考模型、上下文相关的方法，并能对模型进行裁剪。

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。

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

△代码互查(Peer Review):

这是最常见的形式,开发人员之间相互审查彼此的代码变更。可以利用GitHub等协作工具,通过PR(Pull Request)的diff功能来进行评审。优点是及时发现问题,提高代码质量,也有助于知识共享。

△代码走查(Code Walkthrough):

由代码作者主动邀请其他开发人员,一起走读审查代码。作者对代码进行详细介绍和解释,大家一起讨论分析。有助于发现隐藏的问题,增强代码的可理解性。

△会议评审(Review Meeting):

定期组织专门的代码评审会议。一般会集中审查某个功能模块或者重点关注的代码。可以充分讨论,集思广益,得出改进意见。

△关键代码的集体评审:

针对系统关键组件或模块,组织更加严格和全面的评审。可以邀请多方面专家一起参与评审。确保核心部分的代码质量和安全性。

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. 支持扩展
2. 引导更好的设计
3. 支持变化
4. 避免回归缺陷
5. 保证工作的平稳步调
6. 节省测试时间
7. 规范行为和记录代码

目标：

1. 确保单元模块被正确实现，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

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.集成测试的集成模式及优缺点

△非渐增式：

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：

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**。

缺点：流程步骤比较多

常见模式：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

△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)，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)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)的基础上，将所有模块按照设计要求组装成子系统或系统，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)。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；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

①**明确**测试目标和质量**要求**；

②测试**需求分析**，输出测试项；

③**选择**合适的测试**方法和工具**；

④**设**计测试**用例**、开发测试**脚本**；

⑤**执行**测试、**观察**结果、**报告**缺陷；

⑥**评估**测试过程与结果；

⑦若达到要求则**提交**测试报告，若未达到要求则返回到相应步骤重新进行。

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

④RPS：每次请求发送的数据量

⑤并发连接数

⑥思考时间：用户发出请求之间的间隔时间

4.按照测试**目的**分，性能测试类型分为

①性能**基准**测试：在**系统标准配置**下获得的性能指标数据，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

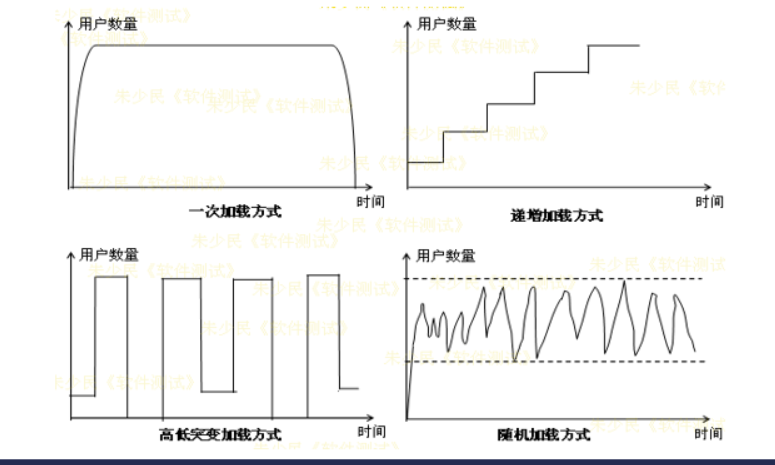
③性能**规划**测试：在多种特定的环境下，获得**不同配置**的系统性能指标，从而决定系统部署的软硬件配置选型。

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：通过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：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**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：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：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？

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。

**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

④SQL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**规定时间**内及**规定环境条件**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，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

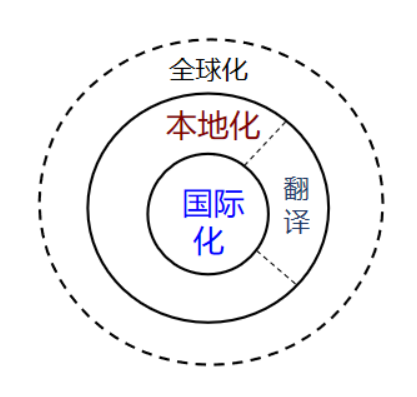
第八章

1 软件本地化，国际化，全球化，相互关系

I18N: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。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。

L10N: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。

G11N：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。**



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

1.自动化测试与测试自动化

自动化测试：**相对于手工测试**存在的概念，由手工逐个运行测试用例被测试工具或系统自动执行所代替，包括输入数据自动生成、结果验证、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

实现基础：可以通过特定的程序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）

实现原理：对象识别

2.API自动化测试（动态测试）

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）

实现原理：代码静态检测工具

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

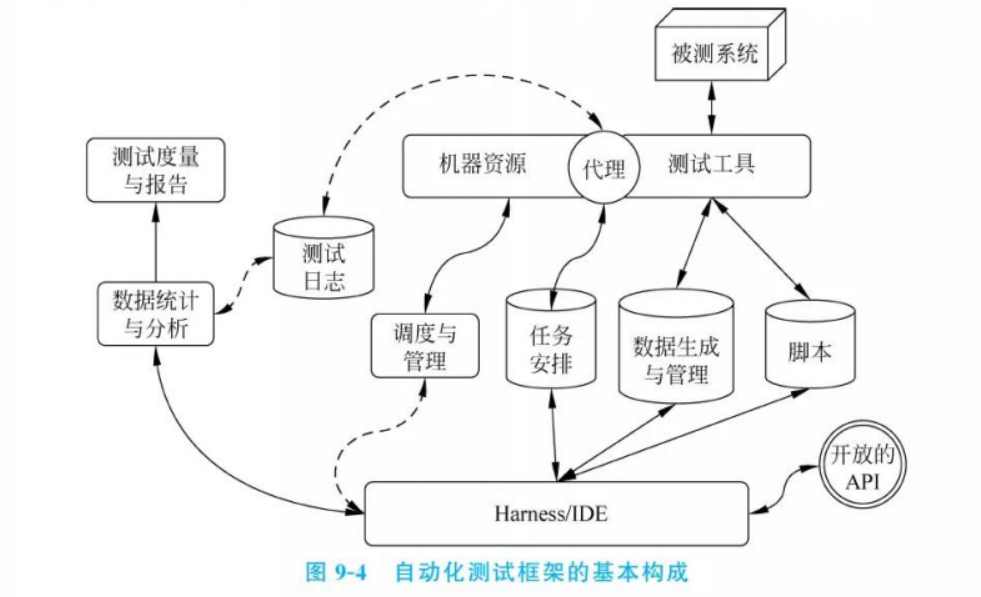
②结构化脚本：类似于结构化程序设计，具有各种逻辑结构且具有函数调用功能。

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）