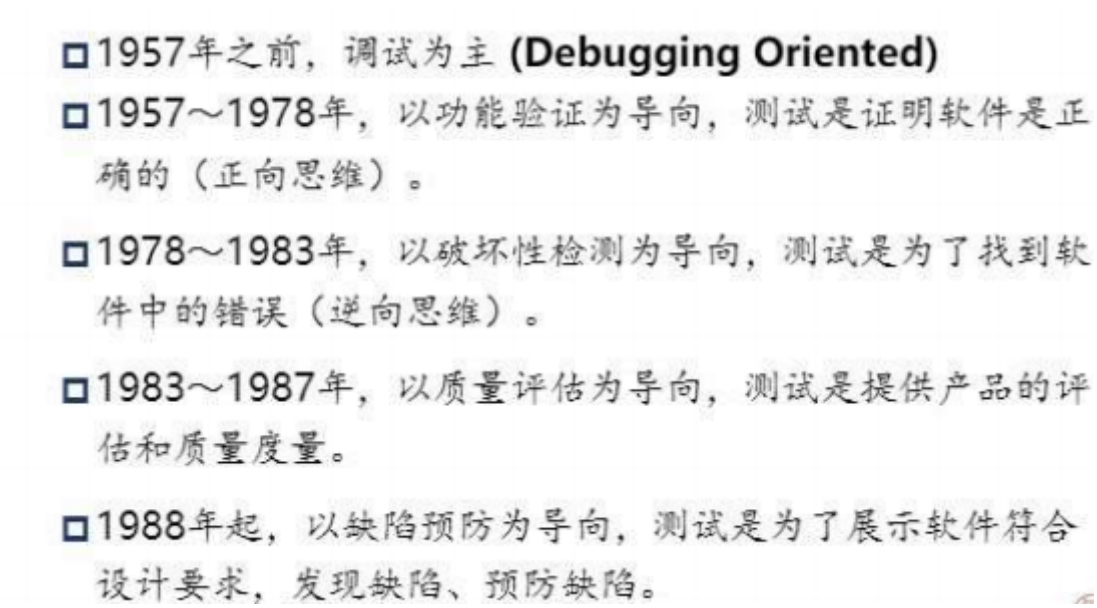
《软件测试技术》复习大纲

第一章 引论

软件测试的定义

在**特定的条件下**运行系统或组件，观察或记录结果，对系统或组件的某个方面做出**评价**

1 软件测试学科的发展



2 正向测试与反向测试的定义，关系

3 从经济视角认知软件测试

测试的经济观点：就是以最**小的代价获得最高的软件产品质量。**

- 经济观点也要求软件测试**尽早**开展工作，发现缺陷越早，返工的工作量就越小，所造成的损**失就越小。**
- **测试的成本<缺陷造成的损失**，测试才有意义。

4 SQA，与软件测试关系

SQA：软件质量保证(Software QualityAssurance，SQA)活动是通过对**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信息，形成分析结果以指导软件过程。

关系：

- 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是一项**管理工作**，侧重于对流程的评审和监控
- 测试是一项**技术性**的工作，侧重对产品进行评估和验证

第二章 软件测试基本概念

1 缺陷定义，现象，判定准则

- 定义：
  - 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，存在缺陷
  - **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
- 判断准则
  - 1.**需求规格说明书**和其它需求、设计规范文档
  - 2.**竞争对手**的产品
  - 3.**启发**式测试预言(Heuristic oracle)
  - 4.**统计**测试预言(Statistical oracle)
  - 5.**一致性**测试预言(Consistency oracle)
  - 6.基于**模型**的测试预言(Model-based oracle)
  - 7.**人类**预言(Human oracle)

2 软件缺陷产生的原因有哪?

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，误解
3. 软件本身
   - **文档错误**、用户使用场合
   - 时间上不协调、或不一致性所带来的问题
   - 系统的自我恢复或数据的异地备份、灾难性恢复等问题

3 产品质量的内容，内部，外部，使用质量

4 如何理解软件规格说明书缺陷(待完善)

5 verification,validation

- verification

  是否正确地构造了软件?即是否**正确地做事**，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的一致性

- validation
  是否构造了正是用户所需要的软件?即是否正在**做正确的事**。验证产品所实现的功能是否满足用户的需求

6 软件测试不同层次测试的对象和任务

- 四个层次
  1. 单元测试
     1. 对象：组件/模块/类/函数
     2. 任务：检查程序模块或组件的已实现的功能与定义的功能是否一致、以及编码中是否存在错误。
  2. 集成测试
     1. 对象：单元之间的**接口**
     2. 任务：发现**与接口有关**的模块之间的问题
  3. 系统测试
     1. 对象：由单元构成的**系统**
     2. 任务：系统功能、安全性、健壮性、效率。
        - 功能性测试：基于**产品功能说明书**，针对产品所实现的功能，从**用户**角度来进行功能验证。
        - 非功能性测试：在实际运行环境下验证系统的非功能性：
          - 性能测试
          - 安全性
          - 稳定性
          - 兼容性
  4. 验收测试
     1. 对象：系统承载的用户**业务**
     2. 任务：目的是向未来的用户**表明系统能够像预定要求那样工作**，验证软件的功能和性能如同用户所合理期待的那样

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?

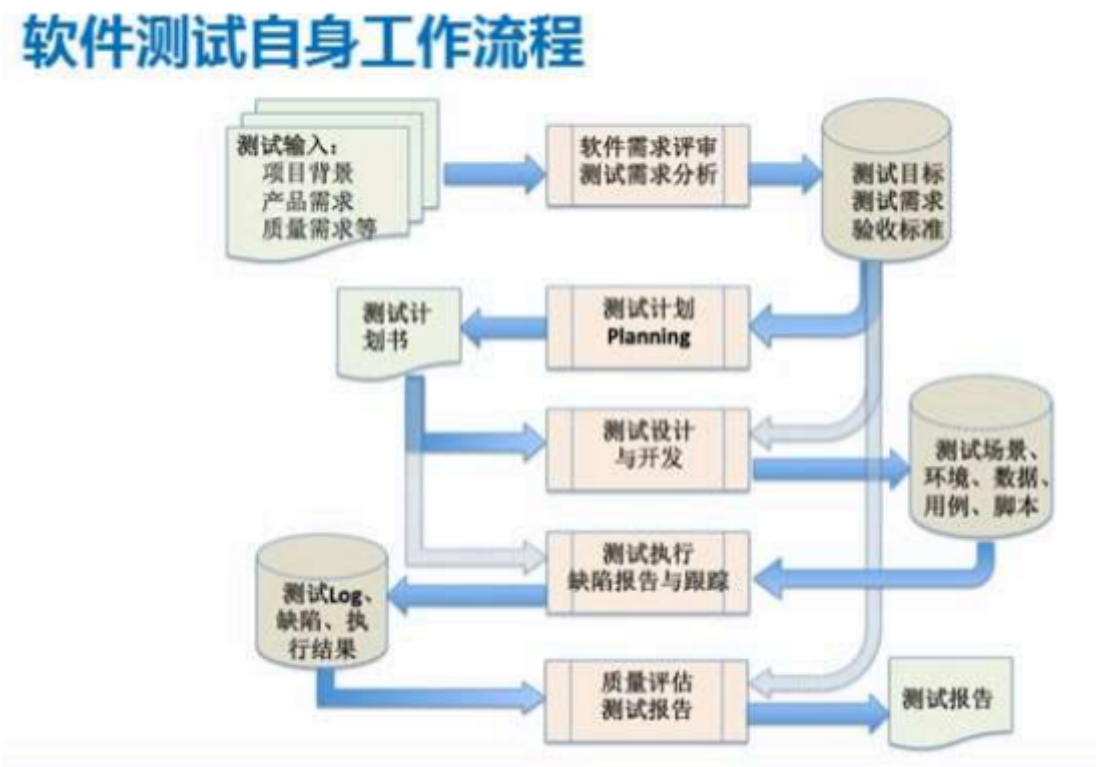
- 静态测试包括对软件产品的需求和设计文档、代码的评审(技术评审、文档评审等)，以及对代码的静态分析等
- 形式
  - 产品评审
    - 需求评审
    - 设计评审
  - 静态分析

- 人工检测（代码风格、质量的检验）
- 计算机辅助静态分析（利用静态分析工具对代码进行特性分析）

代码静态分析：通过**词法分析**(Lexer)、**语法分析**(parser)、**控制流**分析、**数据流**分析等技术对程序代码进行扫描，验证代码是否满足规范性、质量要求等

静态分析技术可以采用**模拟程序执行的技术**，如**符号执行、抽象解释、值依赖分析**等，并采用**数学约束求解工具**进行路径约减或者可达性分析以减少误报、增加效率

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
    - 测试范围
    - 优先级
    - 明确完成哪些相应的测试任务才能确保目标的实现
2. 测试策略指定
    - 测试方式：手工&自动化、静态&动态、探索式&脚本、团队&众测
3. 测试计划
    - 目标和范围
    - 测试项及其优先级
    - 测试风险识别与分析
    - 指定测试策略
    - 进度安排
    - 资源配置
    - 跟踪和控制机制
4. 测试设计
    - “如何测的问题” 分为 `测试总体设计` 和 `测试详细设计`
    - 总体：测试方案、测试结构的设计
    - 详细：测试用例的设计

5. 测试执行
    - 手动执行
    - 自动化执行
6. 测试结果和过程评估
    - 测试结果评估（测试覆盖率、缺陷的趋势和分布分析）
    - 测试过程评估（评审。把计划的测试活动和实际执行的活动比较）

第三章 软件测试方法

1 等价类

- 等价类是某个输入域的子集，在该子集中每个输入数据的作用是等效的
- 将输入数据分成若干个等价类，从每个等价类选取一个代表性的数据作为测试用例
- 等价类分为**有效**等价类(合理的数据)和**无效**等价类(异常的数据)

2 边界值法

- 很多错误发生在输入或输出范围的边界上，因此针对各种边界情况设置测试用例，可以更有效地发现缺陷。
- 设计方法:
    - 确定边界情况(输入或输出等价类的边界)
    - 选取**正好等于、刚刚大于或小于边界**值作为测试数据

3 决策表，因果图法

决策表法

- 在实际应用中，许多输入是由多个因素构成，而不是单一因素，这时就需要多因素组合分析。
- 对于多因素，有时可以直接对输入条件(成立或不成立)进行组合设计，不需要进行因果分析，这时就采用判定表方法
- 判定表由“条件和活动”两部分组成，即列出一个测试活动执行所需的条件组合，所有可能的条件(输入)组合定义了一系列的选择，而测试活动(结果输出)需要考虑每一个选择。

	ID	项目名称	R1	R2	R3	R4	R5
条件项	C1	此商品在经营范围	N	Y	Y	Y	Y
	C2	此商品可以发货	-	Y	Y	N	N
	C3	此客户没有拖欠过付款	-	Y	N	Y	N
动作项	A1	货到后允许客户转账		1			
	A2	货到客户必须立即付款			1		
	A3	重新组织货源				1	1
	A4	电话通知				1	
	A5	书面通知	1				1

因果图方法

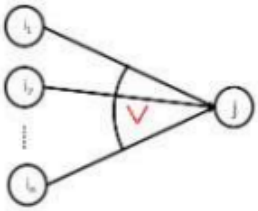
- 分析软件Spec描述的哪些是原因(输入条件)、哪些是结果(输出)，给每个原因和结果赋予一个标示符。
- 找出原因与结果、原因与原因之间的对应关系，画出因果图

- 在因果国上标上哪些不可能发生的因果关系，表明约束或限制条件
- 根据因果图，创建(转化为)判定表，将复杂的逻辑关系转化为简单的组合矩阵
- 把判定表的每一列转化为测试用例。

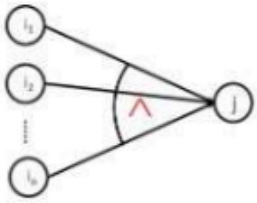
- 有因必有果关系

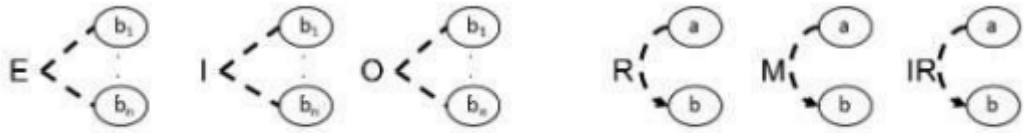
- 非-关系：只有当因 i 不存在时，果 j 才出现

- **或-关系**：如果因 i_1 **或**因 i_2 **或**……因 i_n 存在时，结果 j 才出现。



- **与-关系**：只有当因 i_1 **与**因 i_2 **与**……因 i_n 同时存在时，结果 j 才出现。





E（互斥）：最多只能有一个条件被满足

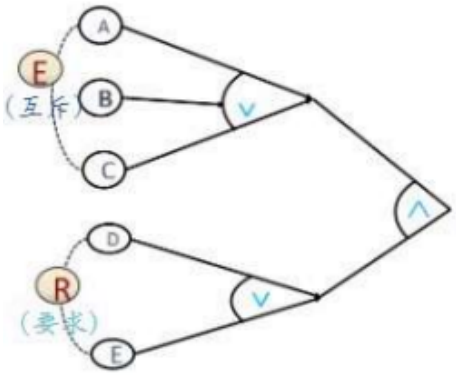
I（包含）：至少有一个条件被满足

O（唯一）：正好（只能一个）条件满足

R（要求）：满足了条件 a 要求满足条件 b

M（屏蔽）：满足条件 a 隐含的要求不满足条件 b

IR（无关）：如果满足了条件 a，条件b 就无关重要了



（A or B or C） And (D or E)

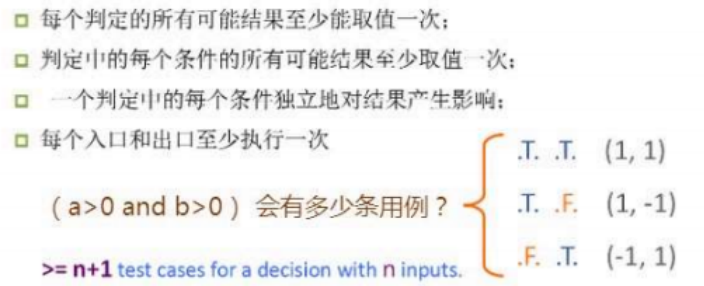
A	1	1	1	1	1	1	0	0	0	0
B	0	0	0	0	1	1	1	1	0	0
C	0	1	0	0	0	0	0	0	1	1
D	0	0	0	1	0	1	0	1	0	1
E	0	0	1	1	0	1	0	1	0	1
结果	0	0	0	0	0	1	0	1	0	1

4 各种逻辑覆盖法

参考连接：https://blog.csdn.net/qq_44239058/article/details/109829236

1. 语句覆盖：使程序中的每个可执行语句至少被执行一次（开发写错了，则检测不出来）
2. 判定/分支覆盖
    1. 判定覆盖:设计若干用例，运行被测程序，使得程序中**每个判断的取真分支和取假分支**至少经历一次，即判断真假值均曾被满足
    2. 一个判定代表着程序的一个分支，所以判定覆盖也被称为分支覆盖。
3. 条件覆盖
    1. 设计若干测试用例，执行被测程序以后，要使每个判断中**每个条件的可能取值至少满足**一次
4. 判定-条件覆盖

    设计足够的测试用例，使得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. 条件组合测试
    设计足够的测试用例，使得判断中每个条件的所有可能至少出现一次，并且每个判断本身的判定结果也至少出现一次
6. 修正条件/判定覆盖（MC/DC）

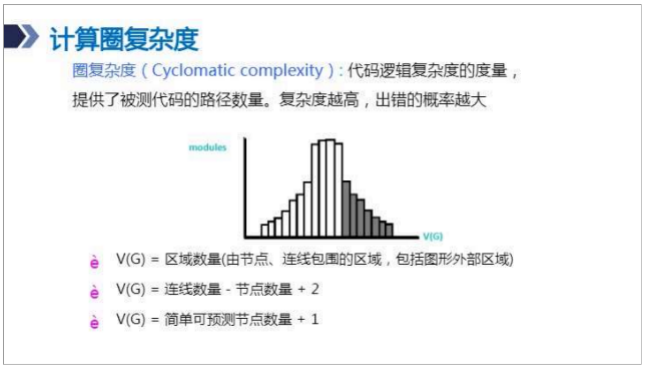


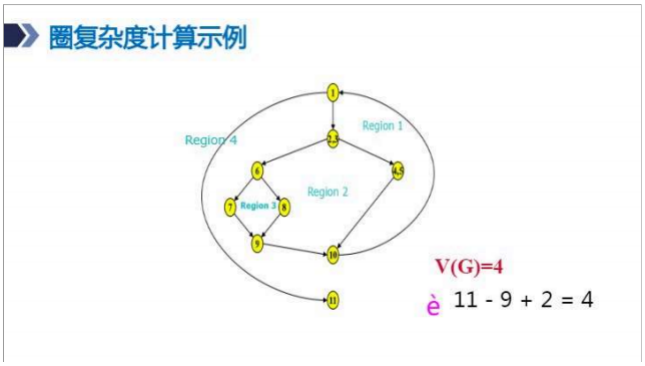
5 路径覆盖法

设计所有的测试用例，来覆盖程序中基本的执行路径

- 基本路径覆盖的设计过程
    1. 依据代码绘制流程图
    2. 确定流程图的圈复杂度(cyclomaccomplexity)
    3. 确定线性独立路径的基本集合(basisset)
    4. 设计测试用例覆盖每条基本路径

- 计算圈复杂度

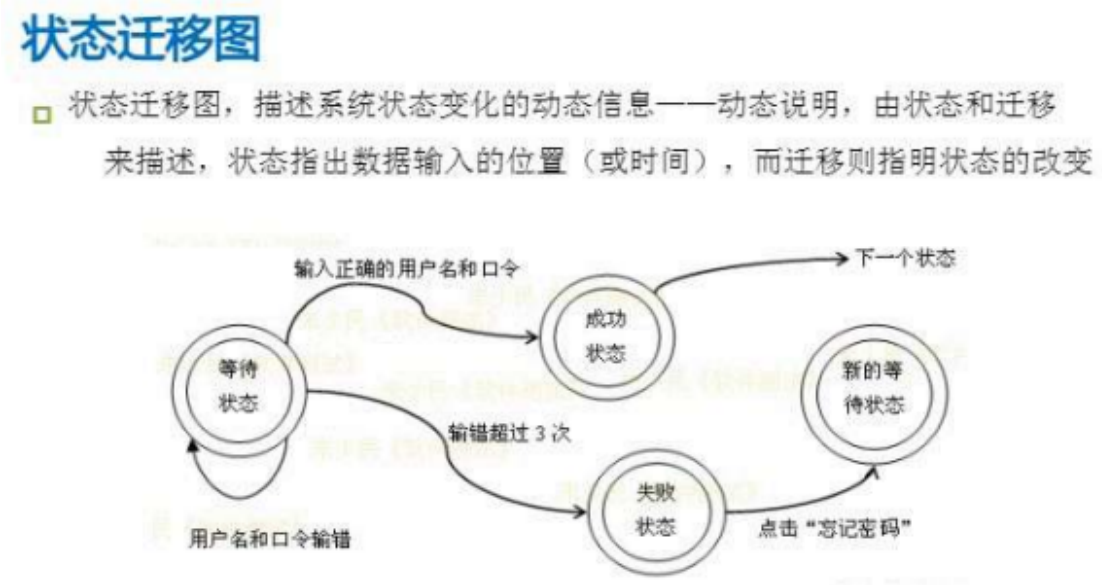




6 功能图法，EFMS***

功能图法

- 每个程序的功能通常由静态说明和动态说明组成
  - 静态说明描述了输入条件和输出条件之间的对应关系
  - 动态说明描述了输入数据的次序或者转移的次序
- 功能图法就是为了解决动态说明问题的一种测试用例的设计方法
- 功能图由**状态迁移图**(statetransition diagram，STD)和**逻辑功能模型**(logic function model, LFM)构成
- 状态迁移图



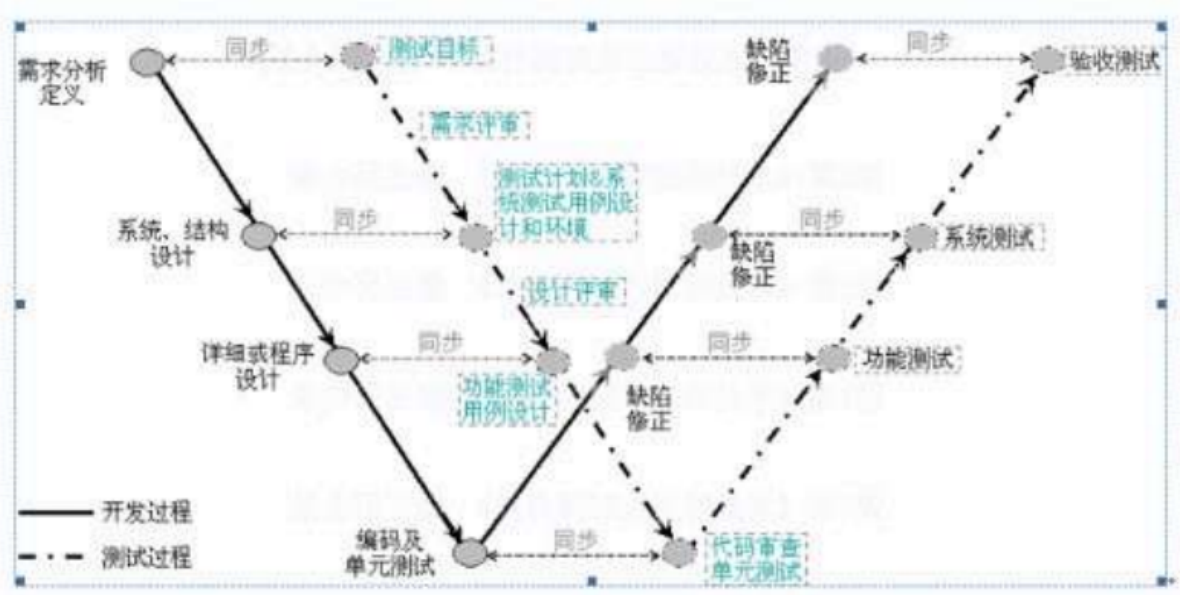
-

第四章 软件测试流程和规范

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，以及第1章介绍的验收测试驱动开发(ATDD);
- 测试右移:是在线测试(Testin Production，TiP)，包括在线性能监控与分析、A/B测试和日志分析等，可以和现在流行的DevOp联系起来

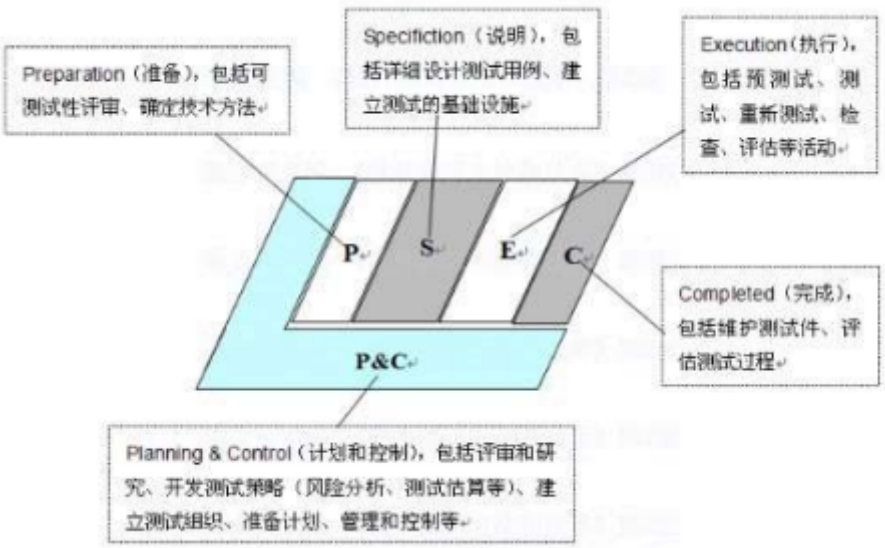
2 w模型



3 TMAP定义，几个阶段，模型基石及关系

定义：TMap (Test Management Approach,测试管理方法)是一种**结构化的、基于风险策略的测试方法体系**,**目的**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

阶段：TMap所定义的测试生命周期由**计划和控制、准备、说明、执行和完成**等阶段组成



基石： **(LOIT)**

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

4 SBTM基本要素，结果，原理

SBTM：基于会话的测试管理

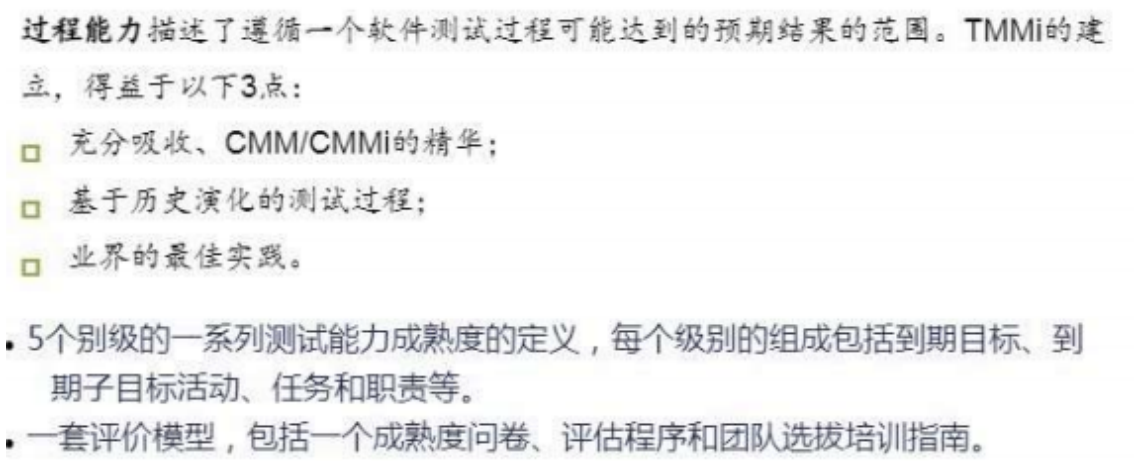
- Session：Session(会话)是**一段不受打扰的测试时间**(通常是90分钟)，是测试管理的最小单元
  每个session关联一个特定的、目标明确的测试任务
- Charter(章程，即测试指导):对**每个session如何执行进行简要的描述**，相当于每个session需要一个简要的计划(提纲)
  一系列Session相互支持，有机地组合在一起，周密地测试了整个产品
- A session sheet(测试报告):相当于测试报告，供第三方(如测试经理、ScrumMaster等)进行检查的材料。它最好能被工具扫描、解析和合成。
  - Session charter(包含任务陈述、测试范围等)
  - Tester name(s)/测试执行者
  - Task breakdown:TBS(Test/Bug/Setup)度量(耗时)，这些数据配合简报有助于估算测试速度、评估测试效率
  - 测试数据、数据文件:为测试数据复用提供了基础
  - Note/测试笔记:测试过程中随时记录的有价值的信息，叙述了测试故事为什么测试，如何测，为什么这样的测试已足够好
  - Issues/问题/风险:测试过程中的问题和疑惑(未来测试的参考资料)
  - 缺陷:测试的直接产出
- Debriefing(听取口头报告):口头汇报，更准确地说，是测试人和其lead/Manager之间的对话
  - PROOF
    - Past
    - Results
    - Obstacles
    - Outlook 未来要做那些测试
    - Feelings
- 原理***:

5 测试几个学派的特点

1. **分析**流派:认为测试是严格的和技术性的，在学术界有许多支持者
2. **标准**流派: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
3. **质量**流派:强调过程规范性，监督开发人员并充当质量的看门人
4. **上下文**驱动流派:强调人的价值，寻找涉众关心的bug
5. **敏捷**流派:强调自动化测试，使用测试来快速验证开发是完整的

6 TMM，TPI，CTP, STEP定义，特点

1. TMMi



2. **TPI**

   TPI(Test Process Improvement)是基于连续性表示法的测试过程改进的参考模型，是在软件控制、测试知识以及过往经验的基础上开发出来的

   特点*：

3. **CTP**：关键测试过程(CriticalTestProcess，CTP):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

4. STEP（系统化测试和评估过程）：

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

第五章 单元测试与集成测试

1 代码评审的形式及各自特点

1. **走查**
   - 定义:采用讲解、讨论和模拟运行的方式进行的查找错误的活动。
   - 注意:
     - 口引导小组成员在走查前通读设计和编码。
     - 限时，避免跑题
     - 发现问题适当记录，避免现场修改检查要点是代码是否符合标准和规范，是否有逻辑错误
     - 怀疑过程中发现的缺陷比通过测试实例本身发现的缺陷更多
2. **互查**
3. **会议评审**（审查）：
   - 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

2 单元测试定义，作用，目标

- **定义**：单元测试是对软件基本的**组成单元**进行独立的测试
- **作用**：
  - 单元测试可以比系统测试测的更彻底
  - 支持扩展
  - 引导更好的设计
  - 支持变化
  - 保证工作的平稳步调
  - 节省测试时间
- **目标**：
  - 单元模块被正确实现(主要指编码)，包括功能、性能、安全性等，但一般主要介绍单元功能测试。
  - (参数)输入是否正确传递和得到保护(容错)，输出是否正常
  - 内部数据能否保持其完整性，包括变量的正确定义与引用、内存及时释放、全局变量的正确处理和影响最低
  - 代码行、分支覆盖或MC/DC达到要求，如高于80%或95%

3桩程序，驱动程序

- 驱动程序（Driver），也称驱动模块
  用于模拟被测模块的上级模块，能够调用被测模块，驱模块接受测试数据，调用被测模块并把相关数据传送给被测模块。
- 桩程序（Stub），也称桩模块
  用于模拟被测模块工作过程中所调用的下层模块，一般很少进行数据处理，一般只检测被测模块传输数据的正确性。
-
```python
def test(a,b)   # 被测模块
    c = a+b
    stub(c)

def stub(c) # 桩程序
    print(c)

if __name == "__main__":
    test(1,2)   # 驱动程序

```

4 集成测试的概念

集成测试，也叫组装测试或联合测试。在单元测试的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。

5 单一系统的集成测试的集成模式及优缺点

- **非渐增式**测试模式: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。
- **渐增式**测试模式: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，
- **优缺点**
  - 编写的测试软件工作量:渐进式多·

- 发现模块间接口错误:渐进式早
- 错误定位:渐进式容易
- 测试彻底:渐进式
- 机器时间:渐进式多
- 并行测试:非

6 微服务特点，测试目标？测试基本步骤？

7 集成测试的目标

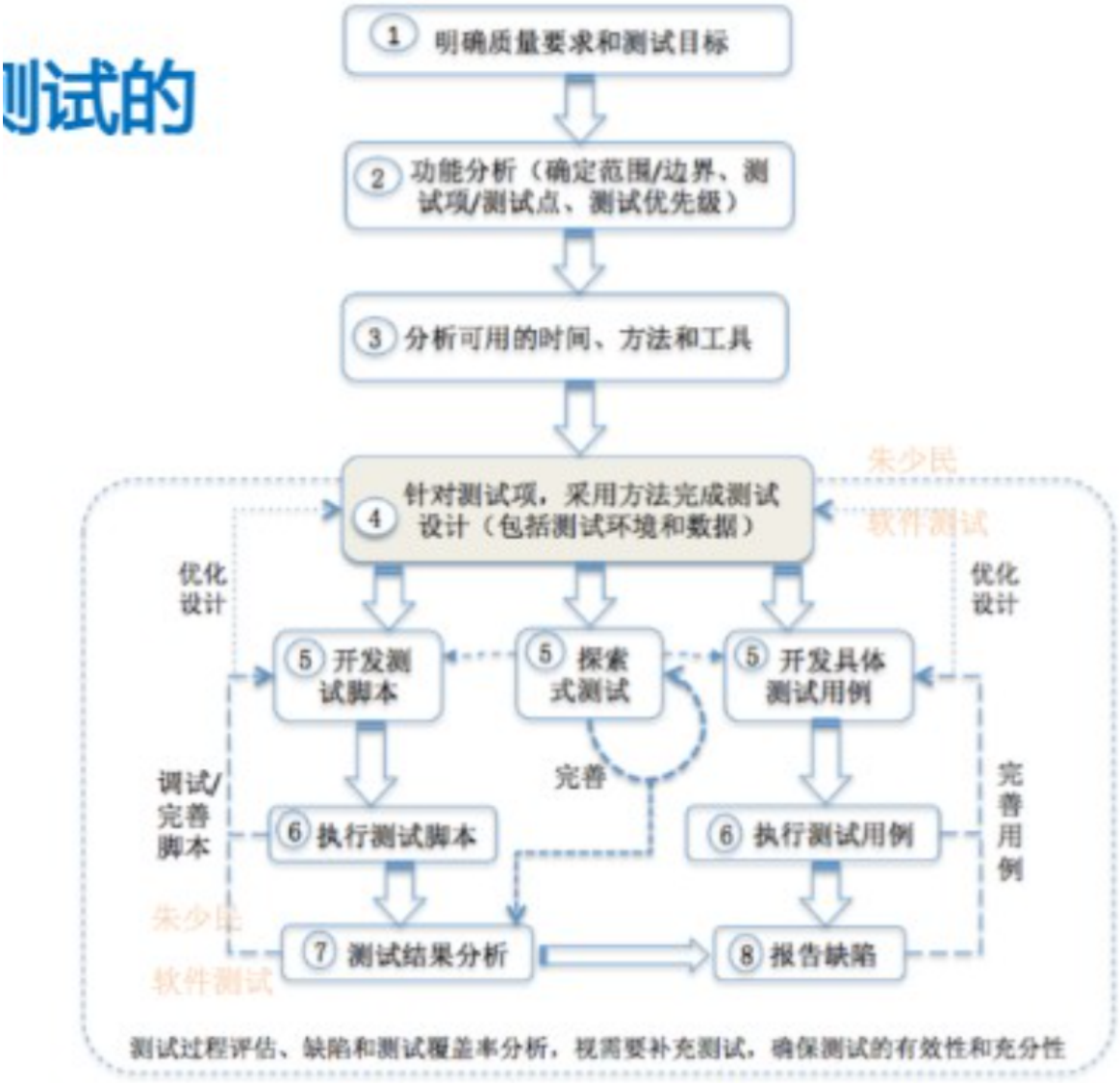
确保在多个独立的软件模块之间集成和交互时系统的正确性、一致性和可靠性。其主要目标包括：

- **确保**各个模块之间的**接口和交互正常工作**，**避免**模块之间出现**不一致和冲突。**
- **检测系统整体性能**，包括系统的响应时间、吞吐量、并发性等指标，以确保系统能够满足用户的需求。
- **发现和修复集成问题**，例如数据传递错误、功能冲突、逻辑错误等。
- 确保**系统的稳定性和可靠性**，避免因为集成问题导致系统崩溃或数据丢失。
- 最终确认系统是否达到了预期的功能和质量标准。

第六章 系统功能测试

1 功能测试的基本思路


明确测试目标和质量要求
测试需求分析，输出测试项
选择合适的测试方法、工具
设计测试用例、开发测试脚本
执行测试、观察结果、报告缺陷
评估测试过程与结果
未达到要求
达到要求
提交测试报告



2 回归测试需要解决的问题。回归测试的策略和方法

回归测试就是为了发现回归缺陷而进行的测试。

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

策略：

- 再测试全部用例
- 基于风险选择测试
- 基于操作剖面选择测试
- 再测试修改的部分

第七章

1 性能测试目标

- 获取系统性能某些指标数据（获取指标数据）
- 为了验证系统是否达到了用户提出的性能指标（验证是否满足用户需求）
- 发现系统中存在的性能瓶颈，优化系统的性能（发现瓶颈，优化性能）

2 什么是性能测试?

性能测试(performance test)就是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，最后对测试结果进厅分析以确定系统的性能状况。

3 系统性能表现

4 按照测试目的分，性能测试类型

- **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(Baseline)
- **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
- **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
- **容量测试**可以看作性能的测试一种，因为系统的容量可以看作是系统性能指标之一
- **压力测试**(Stress test):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。
- **渗入测试**(soaktest)，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
- **峰谷测试**(peak-resttest)，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. 选择测试工具和开发相应的测试脚本
3. 建立性能测测试负载模型
    - 确定并发虚拟用户数量，每次请求的数据量，思考时间，加载方式和持续加载的时间等
    - 是个迭代完善的过程
4. 执行测试
5. 提交测试报告，进行分析

6 什么是安全性测试

软件安全性测试就是全面检验软件在需求规格说明中规定的**防止危险状况措施的有效性**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果，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

- **全程监控**:采用类似wireshark的嗅探软件进行全程抓包嗅探
- **择要监控**:对扫描过程不进行录制，仅仅在数据分析后，准备发起渗透前才开启软件进行嗅探
- **主机监控**:仅监控受测主机的存活状态
- **指定攻击源**:多方监控指定源(某主机)进程、网络连接、数据传输等
- **对关键系统**，可以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？

- 安全功能测试(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
- 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、管理上存在的可被利用的缺陷或弱点

9 安全性测试的任务

- 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、几余、中断处理等方案，进行针对性测试
- 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
- 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**

  从软件外部考家其安全性，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(Cross-Site Scripting,跨站脚本攻击)是一种代码注入攻击。攻击者在目标网站上注入恶意代码，当用户(被攻击者)登录网站时就会执行这些恶意代码，通过这些脚本可以读取cookie,session tokens，或者网站其他敏感的网站信息，对用户进行钓鱼欺诈

> 从Web程序开发角度，需要遵循安全开发原则，采取措施防止跨站脚本攻击：
>
> - 对用户输入进行验证和过滤，验证输入是否符合预期格式，过滤掉一些特殊字符和标签（例如）避免注入恶意脚本。
> - 对用户输入进行转义，也就是将用户输入呈现给用户之前确保对其进行转义，例如将“>”转义为“&gt”，这样可以防止接收到的输入中的一些字符被解释为可执行代码，使恶意脚本失效。
> - 对Cookie采取安全措施，例如设置HttpOnly Cookie属性防止JavaScript读取Cookie，避免用户身份验证令牌和敏感信息被窃取。
> - 使用HTTP的响应头CSP（Concent Security Policy，内容安全策略）限制哪些资源可以被加载和执行，例如限制JavaScript的来源，从而防止恶意脚本注入。
> - Web程序开发需要遵循OWASP的Cross Site Scripting Prevention Cheat Sheet（跨站脚本攻击预防备忘录），继承已有经验。
> - 对Web程序进行渗透测试，加固跨站脚本漏洞。
>
> 从访问网站的用户角度，需要有风险意识，避免遭受跨站脚本攻击：
>
> - 在浏览器中设置禁用脚本。
> - 避免点击电子邮件、论坛中的不明链接。
> - 及时更新软件及操作系统补丁。
> - 安装杀毒软件。

sql注入：根据SQL语句的编写规则，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

> - 1. 使用参数化查询
> - 2. 输入验证和过滤
> - 3. 使用存储过程

> - 4. 最小权限原则
> - 5. 使用ORM框架
> - 6. 使用准备语句
> - 7. 使用安全的数据库连接
> - 8. 避免动态拼接SQL语句
> - 9. 使用防火墙和入侵检测系统
> - 10. 定期更新和维护数据库软件

12 web安全性测试可从哪些方法开展（待完善）

13 什么是软件可靠性？可从哪几个指标度量？各自的定义

- **可靠性**:在规定的一段时间和条件下，软件能维持其性能水平的能力有关的一组属性，可用成熟性、容错性、易恢复性三个基本子特性来度量。
- **成熟性度量**:通过错误发现率DDP(DefectDetectionPercentage)来表现。
  DDP越小，软件越成熟。
  $DDP = 测试发现的错误数量/已知的全部错误数量$
- **容错性测试**在下面一节介绍
- **恢复性的测试先设法**(模拟)使系统崩溃、失效等，然后计算其系统和数据恢复的时间来做出易恢复性评估。

14 容错测试的要点?

容错测试是一种**对抗性**的测试过程。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(IO)故障或无效的数据库指针和关键字等

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，通过用户反馈的真是数据来确定哪一个更好
- 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
- 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
- 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（待完善）

- **软件本地化**:

2 unicode与utf-x关系，特点（待完善）

3 软件本地化基本步骤

1. 建立配置管理体系，跟踪目标语言各个版本的源代码
2. 创造和维护术语表
3. 源语言代码和资源文件分离、或提取需要本地化的文本
4. 把分离或提取的文本、图片等翻译成目标语言
5. 把翻译好的文本、图片重新检入目标语言的源代码版本

6. 如果需要，编译目标语言的源代码
7. 测试翻译后的软件，调整UI以适应翻译后的文本
8. 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

- 功能性测试，所有基本功能、安装、升级等测试
- 翻译测试，包括语言完整性、术语准确性等的检查
- 可用性测试，包括用户界面、度量衡和时区等
- 兼容性调试，包括硬件兼容性、版本兼容性等测试
- 文化、宗教、喜好等适用性测试
- 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（待完善）

第九章

1 自动化测试与测试自动化

- 通过平台、系统或工具**自动地完成测试的某类工作**都可以归为**测试自动化**
- **自动化测试**更侧重的测试用例或测试数据生成、测试执行和测试结果呈现等自动化
- **自动化测试**为评审提供辅助工具和代码静态分析
- **自动化测试**可以覆盖系统的接口测试、U功能测试和专项测试

2 如何理解测试自动化

3 测试自动化实现的原理，几种技术

- **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。
- **对象识别** （Windows 对象 、Mac 对象、Web DOM对象）
- **脚本技术**： 线性脚本 结构化脚本 数据驱动脚本、关键字驱动脚本
- **自动比较技术**：静态比较和动态比较， 简单比较和复杂比较， 敏感性测试比较和健壮性测试比较，比较过滤器
- **测试自动化系统的构成**：测试工具的分类、测试工具的选择、测试自动化普遍存在的问题、自动化测试的引入和应用

4 自动化测试的流程

5 几种脚本技术

- 线性脚本
- 结构化脚本
- 数据驱动脚本
- 关键字驱动脚本

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

- Harness/IDE脚本语言(Script Language)
- 代理(Agents)
- 工具(Tools)
- 任务安排(scheduler)

- 报告(Report)