软件测试技术提纲整理

第一章：引论（了解）

1.为什么要进行软件测试（作用）

软件测试——产品质量的保证

软件测试——控制成本的关键

软件测试——软件可靠性确认

软件测试——让企业具备国际竞争的实力

2.什么是软件测试（重点）

Bill Hetzel 博士（正向思维的代表）：

软件测试就是为程序能够按预期设想那样运行而建立足够的信心。

“软件测试是一系列活动以评价一个程序或系统的特性或能力并确定是否达到预期的结果”

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。

Glenford J. Myers （反向思维的代表）：

测试是为了证明程序有错，而不是证明程序无错误

一个好的测试用例是在于它能发现至今未发现的错误

一个成功的测试是发现了至今未发现的错误的测试

软件测试是由“验证（Verification）”和“有效性确认（Validation）”活动构成的整体

验证”是检验软件是否已正确地实现了产品规格书所定义的系统功能和特性。

“有效性确认”是确认所开发的软件是否满足用户真正需求的活动。

3.测试和质量保证的关系（SQA&测试）（重点）

SQA(软件质量保证)活动是通过对软件产品有计划地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。

a）SQA 和软件测试之间相辅相成，既有包含又有交叉的关系。

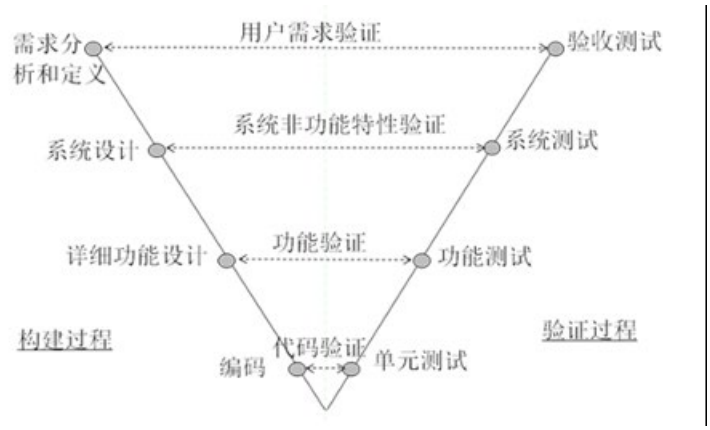
b）SQA 指导、监督软件测试的计划和执行，督促测试工作的结果客观、准确和有效，并协助测试流程的改进。

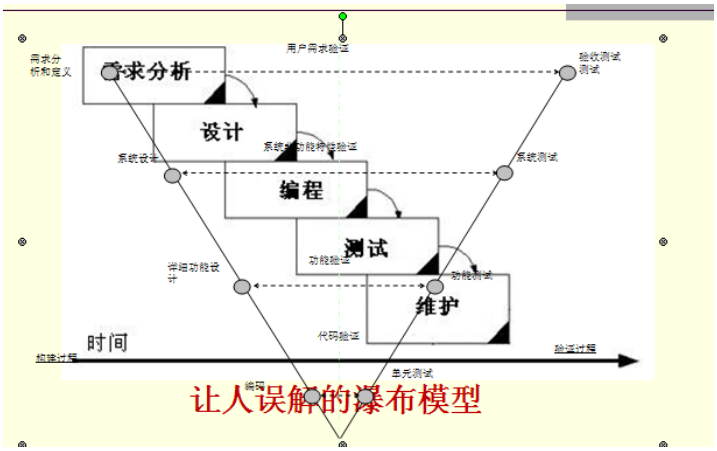
c）软件测试是 SQA 重要手段之一，为 SQA 提供所需的数据，作为质量评价的客观依据。

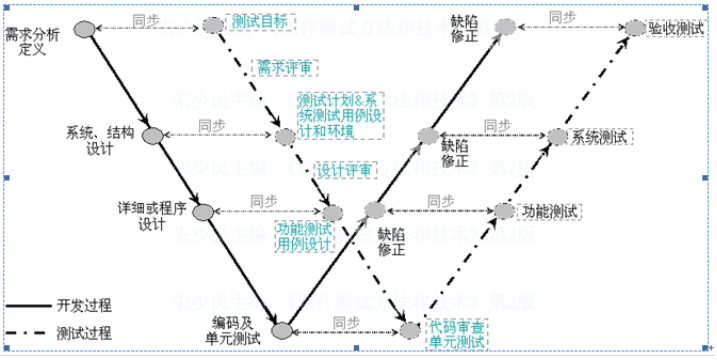
d）相同点：都是贯穿整个软件开发生命周期的流程。

e）不同点：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。

4.软件测试和软件开发的关系（V 模型，W 模型—重点）







5. 软件测试的 4 种导向和 5 大学派

4 种导向：

1)以**功能验证**为导向，测试是证明软件是正确的（正向思维）。

2)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）。

3)以**质量评估**为导向，测试是提供产品的评估和质量度量。

4)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。

5 大流派：

1)分析学派：认为软件是逻辑性的，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。例子：代码覆盖率、结构化测试。

2)标准学派：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。

3)质量学派：软件质量需要规范，测试是过程的质量控制、揭示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。

4)上下文驱动学派：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式测试思维。例子：探索性测试。

5)敏捷学派：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。例子：TDD。

第二章：软件测试的基本概念（了解）

1.软件缺陷定义：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 软件缺陷一个标准的定义：

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；

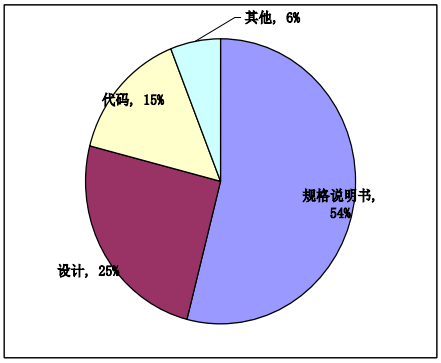
从外部看，软件缺陷是系统所需要实现的某种功能的失效或违背。

2. 软件缺陷的原因

技术问题

算法错误，语法错误，计算和精度问题，接口参数传递不匹配

系统的自我恢复或数据的异地备份、灾难性恢复等问题



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(user scenario)，

时间上不协调、或不一致性所带来的问题

软件规格说明书缺陷最多

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

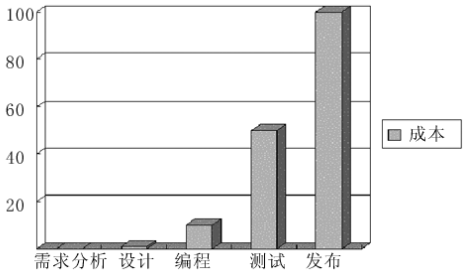
③ 需求变化的不一致性，用户的需求不断变化，不能及时在需求中得到正确描述

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

3.修复软件缺陷的代价

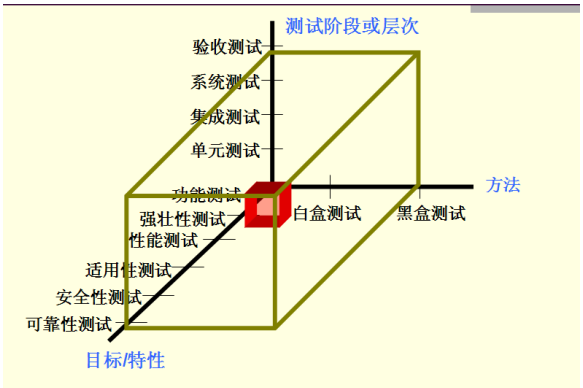
如果需求阶段修正错误的代价是 1，在设计阶段就是它的 3～6 倍，在编程阶段是它的 10 倍，在内部测试阶段是它的 20～40 倍，在外部测试阶段是它的 30～70 倍，而到了产品发布出去时，这个数字就是 40～1000 倍，修正错误的代价不是随时间线性增长，而几乎是呈指数增长的。



用户需求验证 代码验证 功能验证证 系统非功能特性验证

4.结束标准：用例全部测试；覆盖率达到标准；缺陷率达到标准；其他指标达到标准。

5.软件测试的分类（重点）



按测试的对象或范围分类，如单元测试、文档测试、系统测试等）

按测试目的分类，如功能测试、回归测试、性能测试、可靠性测试、安全性测试和兼容性测试等

根据测试过程中被测软件是否被执行，分为静态测试和动态测试

根据是否针对系统的内部结构和具体实现算法来完成测试，可分为白盒测试和黑盒测试

6.静态测试

静态测试就是静态分析，对模块的源代码进行研读，查找错误或收集一些度量数据，并不需要对代码进行编译和仿真运行。静态测试包括对软件产品的需求和设计规格说明书的评审、对程序代码的复审等。静态分析的查错和分析功能是其他方法所不能替代的，可以采用人工检测和计算机辅助静态分析手段进行检测，但越来越多地采用工具进行自动化分析。

7.动态测试

动态测试是通过真正运行程序发现错误，通过观察代码运行过程，来获取系统行为、变量实时结果、内存、堆栈、线程以及测试覆盖度等各方面的信息，来判断系统是否存在问题，或者通过有效的测试用例，对应的输入输出关系来分析被测程序的运行情况，来发现缺陷。

8.回归测试

为保证软件中新的变化(新增加的代码、代码修改等)不会对原有功能的正常使用有影响而进行的测试。也就是说，满足用户需求的原有功能不应该因为代码变化而出现任何新的问题。

9.压力测试

也称负载测试，用来检查系统在不同负载(如数据量、并发用户、连接数等)条件下的系统运行情况，特别是高负载、极限负载下的系统运行情况，以发现系统不稳定、系统性能瓶颈、内存泄漏、CPU 使用率过高等问题。

10.ST / ET

脚本测试（ST）

1. 使用手工测试的测试用例（Test case）和自动化的测试脚本（Test script）
2. 先设计后执行（过程：分析->设计->执行->报告)
3. 阶段性明显，属于较传统的测试方式

探索性测试（ET）

1. 强调测试学习，设计和执行同时展开
2. 没有测试用例，一边想（在头脑中设计）一边测试
3. 缺乏良好的系统性、复用性

11.ST 与 ET 的对比（重点）

ST	ET
系统性强，	高效率
容易管理（可视性强）	适应性强
验证自己的思路	不断问系统
可预见性	学习的过程
先设计、后执行	学习、设计和执行并行
强调逻辑分析	上下文驱动
关注需求和测试文档	强调个人能力
有明确的测试标准	Test Oracle
强调评审、可控	关注与产品的交互
严谨、规范	拥抱变化、乐趣

12.软件测试的工作范畴

软件测试工作的组织与管理：制定测试策略、测试计划，确认所采用的测试方法与规范，控制测试进度，管理测试资源。

测试工作的实施：编制符合标准的测试文档，搭建测试环境，开发测试脚本、与开发组织协作实现各阶段的测试活动。

第三章：软件测试方法（理解）

1.白盒测试的概念

白盒测试也称结构测试或逻辑驱动测试，它是按照程序内部的结构测试程序，通过测试来检测产品内部动作是否按照设计规格说明书的规定正常进行，检验程序中的每条通路是否都能按预定要求正确工作。

2.黑盒测试的概念

黑盒测试也称功能测试，它是通过测试来检测每个功能是否都能正常使用。在测试中，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。

3.什么是测试用例

为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略。内容包括测试目标、测试环境、输入数据、测试步骤、预期结果、测试脚本等，并形成文档。

4.为什么要设计测试用例

测试用例构成了设计和制定测试过程的基础。

测试的“深度”与测试用例的数量成比例。由于每个测试用例反映不同的场景、条件或经由产品的事件流，因而，随着测试用例数量的增加，对产品质量和测试流程也就越有信心。判断测试是否完全的一个主要评测方法是基于需求的覆盖，而这又是以确定、实施和/或执行的测试用例的数量为依据的。

测试工作量与测试用例的数量成比例。根据全面且细化的测试用例，可以更准确地估计测试周期各连续阶段的时间安排。

测试设计和开发的类型以及所需的资源主要都受控于测试用例。

测试用例通常根据它们所关联关系的测试类型或测试需求来分类，而且将随类型和需求进行相应地改变。最佳方案是为每个测试需求至少编制两个测试用例：

一个测试用例用于证明该需求已经满足，通常称作正面测试用例；

另一个测试用例反映某个无法接受、反常或意外的条件或数据，用于论证只有在所需条件下才能够满足该需求，这个测试用例称作负面测试用例。

测试用例是软件测试的核心。

5.白盒测试方法的分类

语句覆盖，语句覆盖法的基本思想是设计若干测试用例，运行被测程序，使程序中的每个可执行语句至少被执行一次。

判定覆盖，判定覆盖法的基本思想是设计若干用例，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。

条件覆盖，条件覆盖的基本思想是设计若干测试用例，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。

路径覆盖，路径覆盖就是设计所有的测试用例，来覆盖程序中的所有可能的执行路径。

判定-条件覆盖，是判定和条件覆盖设计方法的交集，即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

条件组合覆盖，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（即所有组合至少出现一次）

基本路径测试法，在程序或业务控制流程的基础上，分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例。

6.求环路复杂度的 3 种方法

V(G) = 区域数量(由节点、连线包围的区域，包括图形外部区域)

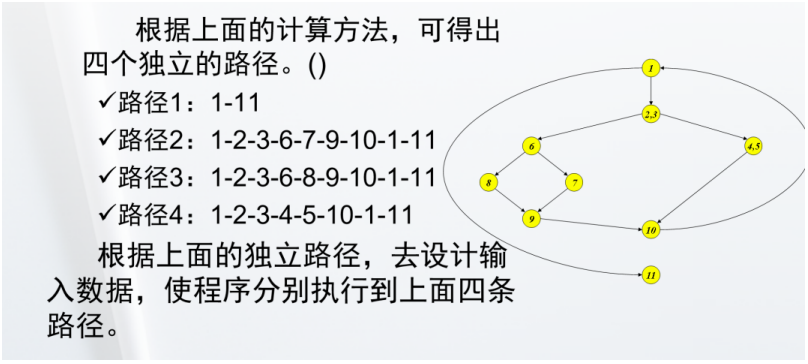
V(G) = 连线数量 - 节点数量 + 2

V(G) = 判定节点数量 + 1

7.为什么需要求环路复杂度

代码逻辑复杂度的度量，提供了被测代码的路径数量。复杂度越高，出错的概率越大。

8.构造基本路径集合的方法



9.黑盒测试方法的分类

等价类划分法（等价分类法），分为有效等价类和无效等价类。有效等价类是有意义的、合理的输入数据，可检查程序是否实现了规格说明中所规定的功能和性能。无效等价类与有效等价类的意义相反。

边界值分析法，确定边界情况（输入或输出等价类的边界），选取正好等于、刚刚大于或刚刚小于边界值作为测试数据。

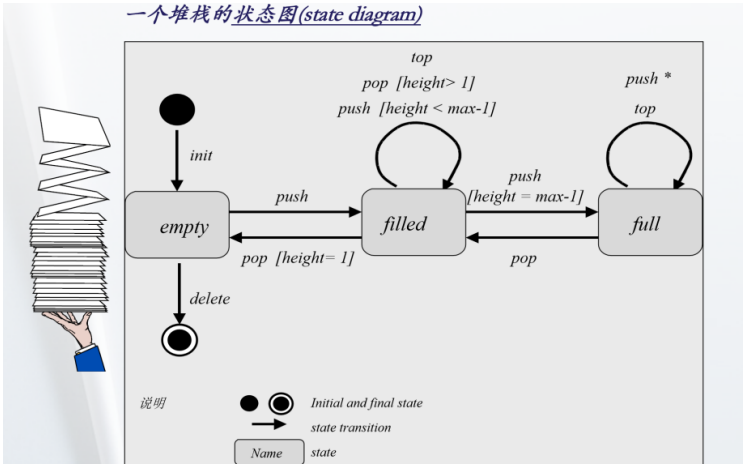
因果图法，借助图形，着重分析输入条件的各种组合，每种组合条件就是因，其必然的输出就是果，利用图解法分析输入的各种组合情况，有时还要依赖所生成的判定表，多种输入条件的组合，产生多种结果设计测试用例。

判定表法，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进行组合设计，不需要进行因果分析，即直接采用判定表方法。

用条件覆盖法设计白盒测试用例

用等价类和边界值设计黑盒测试用例

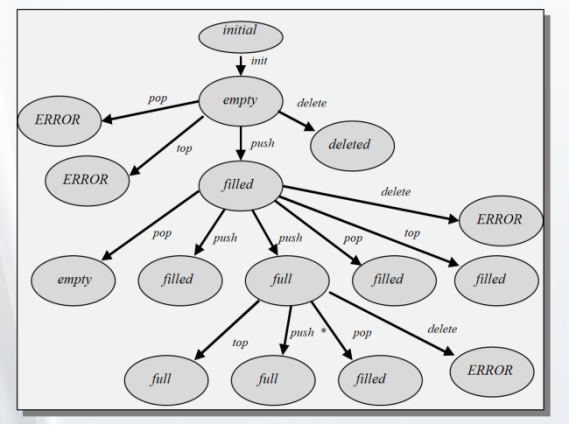
10.堆栈的状态图、状态表、状态转化树（重点）



输入 / 状态	init	push	pop	delete	top
initial	empty				
empty		filled	error	deleted	error
filled		filled(1) full(2)	empty(3) filled(4)	error	filled
full		full	filled	error	full
deleted					

(1)--push [height < max-1]　　(3)--pop [height = 1]

(2)--push [height = max-1]　　(4)--pop [height > 1]

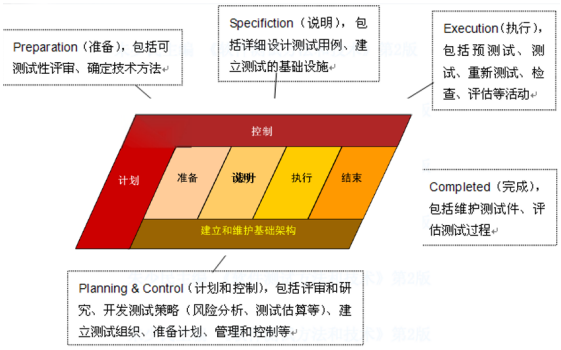


第四章：软件测试流程和规范（了解）

0.TMAP

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。

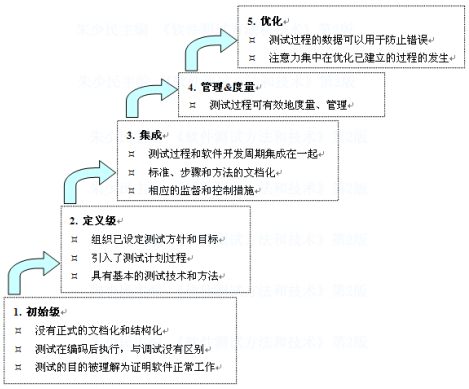
TMap所定义的测试生命周期由计划和控制、准备、说明、执行和完成等阶段组成。



1.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。TMM 的建立，得益于以下 3 点：

充分吸收、CMM 的精华；
基于历史演化的测试过程；
业界的最佳实践。



2.TPI

TPI 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。

3.CTP

关键测试过程评估模型主要是一个内容参考模型，一个上下文相关的方法，并能对模型进行裁剪。

4.STEP

STEP（Systematic Test and Evaluation Process，系统化测试和评估过程）是一个内容参考模型，认定测试是一个生命周期活动，在明确需求后开始直到系统退役。

第五章：单元测试（掌握）

1.单元测试

单元测试是对软件基本组成单元（如函数、类的方法等）进行的测试。

定义：单元测试是对软件基本组成单元进行的测试。

时机：一般在代码完成后由开发人员完成,QA 人员辅助.

概念：模块, 组件, 单元

单元测试的测试人员：程序人员和开发人员

单元测试的测试方法：

检查每一条独立执行路径的测试。保证每条语句被至少执行一次。

检查局部数据结构完整性

检查模块接口是否正确

检查临界数据处理的正确性

预见、预设的各种出错处理是否正确有效

单元测试的依据：详细设计和概要设计

单元测试是对软件基本组成单元进行测试。依据是：软件详细说明书。

单元测试测试的不仅仅是代码，有：接口测试、局部数据结构测试、独立路径测试、独立路径测试、边界条件测试、错误处理测试、功能测试、性能测试、内存使用测试等。

2.单元测试的主要目标

单元模块被正确编码

信息能否正确地流入和流出单元

在单元工作过程中，其内部数据能否保持其完整性，包括内部数据的形式、内容及相互关系不发生错误，也包括全局变量在单元中的处理和影响。

在为限制数据加工而设置的边界处，能否正确工作。

单元的运行能否做到满足特定的逻辑覆盖。

单元中发生了错误，其中的出错处理措施是否有效。

第六章：集成测试和系统测试（掌握）

1.集成测试

集成测试是将软件集成起来，对模块之间的接口进行测试。

顾名思义，集成测试是将软件集成起来后进行测试。集成测试又叫子系统测试、组装测试、部件测试等。

模块内的集成，主要是测试模块内各个接口间的交互集成关系；

子系统内的集成，测试子系统内各个模块间的交互关系；

系统内的集成，测试系统内各个子系统和模块间的集成关系。

集成测试的测试人员：有经验的测试人员和开发者共同

2.集成测试的集成模式和经典代表（重点）

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进来进行测试，测试完后再把下一个应该测试的模块结合起来测试。渐增式测试又可以根据每次添加模块的路线分为自顶向下测试、自底向上测试和混合测试等方式。

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

3.集成测试的主要目标

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。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。

4.驱动程序和桩程序（重点）

驱动模块: 代替上级模块传递测试用例的程序

桩模块：代替下级模块的仿真程序

5.系统测试的概念

系统测试（特征测试）：检验系统所有元素之间协作是否合适，整个系统的性能和功能是否达到要求。其测试内容包括：功能测试，非功能测试与回归测试等。

系统测试的测试人员：软件测试工程师

系统测试的内容：功能测试，回归测试，非功能性试；

非功能性测试（特征测试）包含：性能测试（性能验证测试、性能基准测试、性能规划测试、容量测试）、压力测试、容量测试、安全性测试、可靠性测试、容错性测试

6.系统测试的测试依据

需求说明书，概要设计说明书，详细设计说明书，最重要的是需求说明书。

7.确认测试

确认测试又称有效性测试。有效性测试是在模拟的环境下，运用黑盒测试的方法，验证被测软件是否满足需求规格说明书列出的需求。任务是验证软件的功能和性能及其他特性是否与用户的要求一致。对软件的功能和性能要求在软件需求规格说明书中已经明确规定，它包含的信息就是软件确认测试的基础。

第七章：验收测试（掌握）

1.验收测试

检查软件是否符合合同要求，包括需求规格说明、设计规格说明和用户手册等。其测试内容包括：

易用性测试（用户界面和可用性测试）、

兼容性测试（软件兼容性测试、数据共享兼容性测试、硬件兼容性测试）、

安装测试和可恢复性测试、文档测试等（安装与卸载测试、可恢复性测试）

验收测试的内容（正确性、完备性、易理解性、一致性）

验收测试的测试人员（用户和测试部门共同完成）

验收测试的测试依据：国家规范、行业标准、合同条款、用户确认的需求规格说明书

2.α，β 测试

α 测试是指软件开发公司组织内部人员模拟各类用户行对即将面市软件产品（称为 α 版本）进行测试，试图发现错误并修正。

经过 α 测试调整的软件产品称为 β 版本。紧随其后的 β 测试是指软件开发公司组织各方面的典型用户在日常工作中实际使用 β 版本，并要求用户报告异常情况、提出批评意见。然后软件开发公司再对 β 版本进行改错和完善。

第八章 软件本地化测试（了解）

1.软件国际化（I18N）（重点）

是通过功能设计和代码实现中软件系统有能力处理多种语言和不同文化，使创建不同语言版本时不需要重新编写代码的软件工程方法。

2.软件本地化（L10N）

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.软件全球化（G11N）

是同时实现软件本地化和软件国际化的结果。是一个概念化产品的过程，它基于全球市场考虑，以便一个产品只做较小的改动就可以在世界各地出售。

4.I18N VS L10N

I18N 是 L10N 的基础和前提，为 L10N 做准备

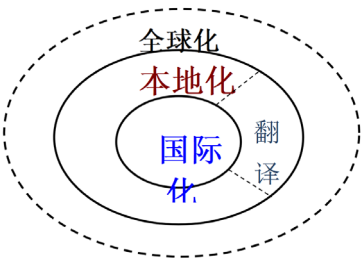
L10N 是 I18N 向特定本地语言环境的转换

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

5. 软件本地化测试

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。



第九章 软件测试自动化（掌握）

1.自动化测试

相对手工测试而存在的一个概念，由手工逐个地运行测试用例的操作过程被测试工具自动执行的过程所代替。

测试工具的使用是自动化测试的主要特征

测试自动化指“一切可以由计算机系统自动完成的测试任务都已经由计算机系统或软件工具、程序来承担并自动执行”

2.测试自动化实现的原理

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。

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

3.自动化测试的引入和应用

找准测试自动化的切入点

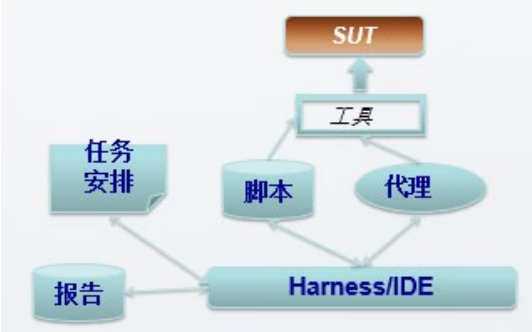
把测试开发纳入整个软件开发体系

测试自动化依赖测试流程和测试用例

软件测试自动化的投入较大

进行资源的合理调度

4.自动化测试框架



Harness/IDE：“夹具”，核心，其他组成部分作为插件与之集成

Agents：负责夹具与工具的通信，控制测试工具的运行

Scheduler：安排和提交定时任务，事件触发任务，以便实现无人值守的自动化测试执行

了解：

5.功能测试工具：QTP

6.性能测试工具：Loadrunner