北京交通大学
BEIJING JIAOTONG UNIVERSITY

第2章 软件测试策略与过程

2020年9月

hhli@bjtu.edu.cn

北京交通大学
BEIJING JIAOTONG UNIVERSITY

第2章 软件测试策略与过程

2.1 软件测试的复杂性分析

2.2 软件测试策略定义

2.3 软件测试分类与方法

2.4 软件测试过程及模型

2.5 单元测试与集成测试

2.6 确认、系统与验收测试

2.1 软件测试的复杂性分析

1、无法对程序进行完全测试

（1）测试所需要的输入量太大

（2）测试的输出结果太多

（3）软件实现的途径太多

（4）软件规格说明没有一个客观标准

2、测试无法显示潜在的软件缺陷和故障

——通过软件测试只能报告软件已被发现的缺陷和故障，无法报告隐藏的软件故障。

3、存在的故障现象与发现的故障数量成正比

——结论：应当对故障集中的程序段进行重点测试

软件测试的复杂性分析（续）

4、不能修复所有的软件故障

——原因：没有足够的时间进行修复；修复的风险较大；不值得修复；可不算做故障的一些缺陷。

——结论：关键是要进行正确的判断、合理的取舍，根据风险分析决定哪些故障必须修复，哪些故障可以不修复。

5、软件测试的代价

——工作原则：如何将故障的可能性减小到一个可以控制的范围，以及如何针对软件风险做出恰当选择，找到最佳的测试量，使得测试工作量不多也不少，既能达到测试的目的，又能较为经济。

软件测试的复杂性分析（续）

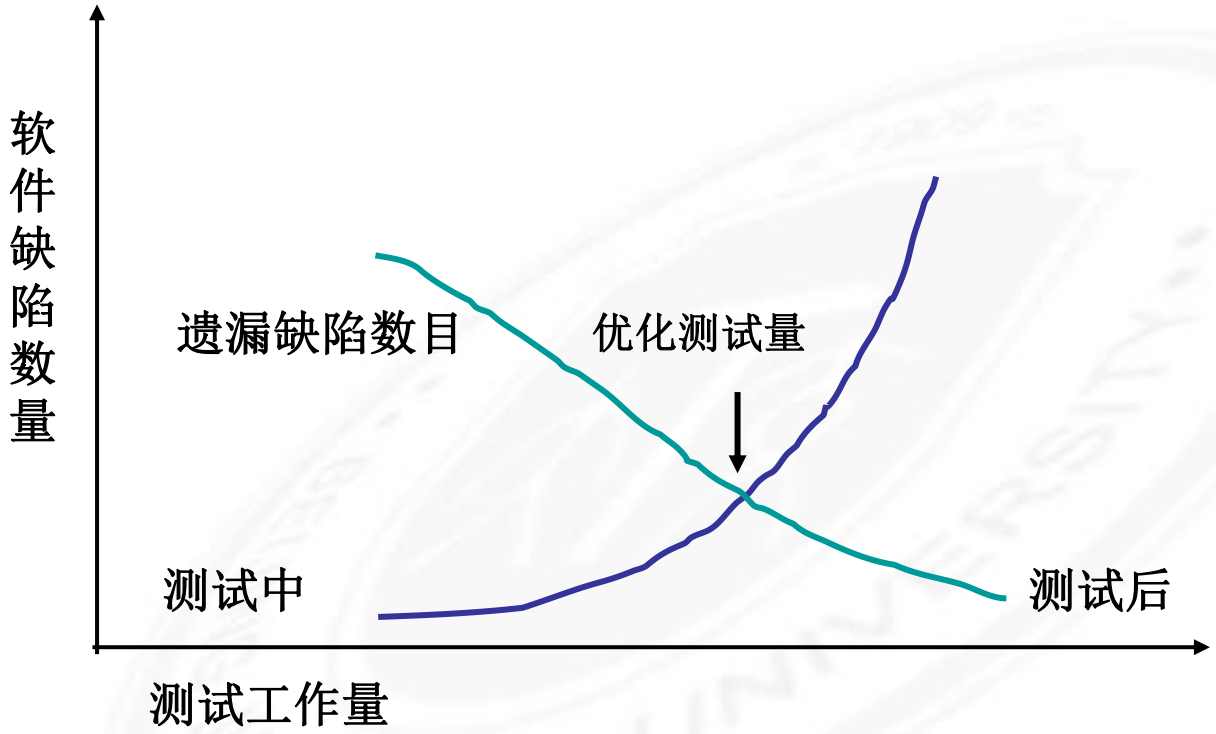


图2-1 测试工作量和软件遗留缺陷数量之间的关系

2.2 软件测试方法与策略

2.2.1 软件测试策略定义

——为软件工程过程定义的一个软件测试的模板，即把特定的测试用例方法放置进去的一系列步骤。

软件测试策略包含的特征：

（1）测试从模块层开始，然后扩大延伸到整个基于计算机的系统集合中。

（2）不同的测试技术适用于不同的时间点。

（3）测试是由软件的开发人员和（对于大型系统而言）独立的测试组来管理的。

（4）测试和调试是不同的活动。

2.2 软件测试方法与策略

软件测试与软件调试的区别

二者在目的、技术和方法等方面存在很大的区别，主要有：

（1）目的不同：测试是为了发现软件中存在的错误；调试是为了证明软件开发的正确性。

（2）任务不同：软件测试属于质量保证活动，它贯穿于整个开发过程；调试是编码活动的一部分，它的任务主要就是排错。 测试的对像可以是文档和代码 而调试的对像只能是代码。

（3）操作者不同： 测试经常是由独立的测试组在不了解软件设计的条件下完成的；调试必须由了解详细设计的开发人员完成。

软件测试与软件调试的区别（2）

（4）指导原则和方法不同:

a. 测试以已知条件开始，且有预知的结果，不可预见的仅是程序是否通过测试；调试一般是以不可知的内部条件开始，结果是不可预见的。

b. 测试是有计划的，需要进行测试设计；调试是不受时间约束的。

c. 测试历经发现错误、改正错误、重新测试的过程；调试是一个推理的过程。

2.2 软件测试方法与策略

- 软件测试策略与测试方法的区别：
- 测试方法是指解决测试问题的技术手段或工具的集合。
- 测试策略指如何选择运用技术方法或工具解决具体问题。
- 软件测试方法有许多，如何运用这些方法测试实际项目呢？需要制定测试策略，确定在测试项目中什么时间、什么任务需要运用哪些技术或哪些工具、如何组织起来完成测试任务。
- 测试策略就是根据项目需要从测试方法集中选择合适的方法，把他们合理地组织起来以完成测试任务。测试策略应能够指导测试工作顺利进行。

软件测试充分性准则（1）

- 对任何软件都存在有限的充分测试集合。

（1）如果一个软件系统在一个测试数据集合上的测试是充分的，那么再多测试一些数据也应该是充分的。这一特性称为单调性。

（2）即使对软件所有成分都进行了充分的测试，也并不表明整个软件的测试已经充分了。这一特性称为非复合性。

软件测试充分性准则（2）

（3）即使对软件系统整体的测试是充分的，也并不意味软件系统中各个成分都已经充分地得到了测试。这个特性称为非分解性。

软件测试的充分性应该与软件的需求和软件的实现都相关。

（4）软件越复杂，需要的测试数据就越多。这一特性称为复杂性。

（5）测试得越多，进一步测试所能得到的充分性增长就越少。这一特性称为回报递减率。

2.3 软件测试分类与方法

1、软件测试分类

软件测试可以从不同角度(维度)加以分类，可以根据测试对象、测试方法、测试目标和测试阶段进行分类。

① 按照测试目标，可分为功能性测试和非功能性测试。

② 按照测试的方式（是否运行被测试软件），软件测试可分为：静态测试和动态测试。

③ 按照软件测试用例的设计方法而论，软件测试可以分为白盒测试法、黑盒测试法和灰盒测试。

④ 按照软件测试阶段来分类，软件测试可分为单元测试、集成测试、系统测试、验证测试和确认测试。

1、软件测试分类

⑤ 按照测试的实施组织方式可分为开发方测试、用户测试和第三方测试。

⑥ 软件测试技术还处于不断发展阶段，类型划分不统一。根据不同的分类方法能够分出几十种测试类型。

1、软件测试分类

- 依据国家软件产品测试相关标准(GB/T25000-2016,GB/T16260)，可从功能性、性能效率、兼容性、易用性、可靠性、信息安全性、维护性、可移植性、使用质量、用户文档等10个质量特性对软件产品进行测试和评价。

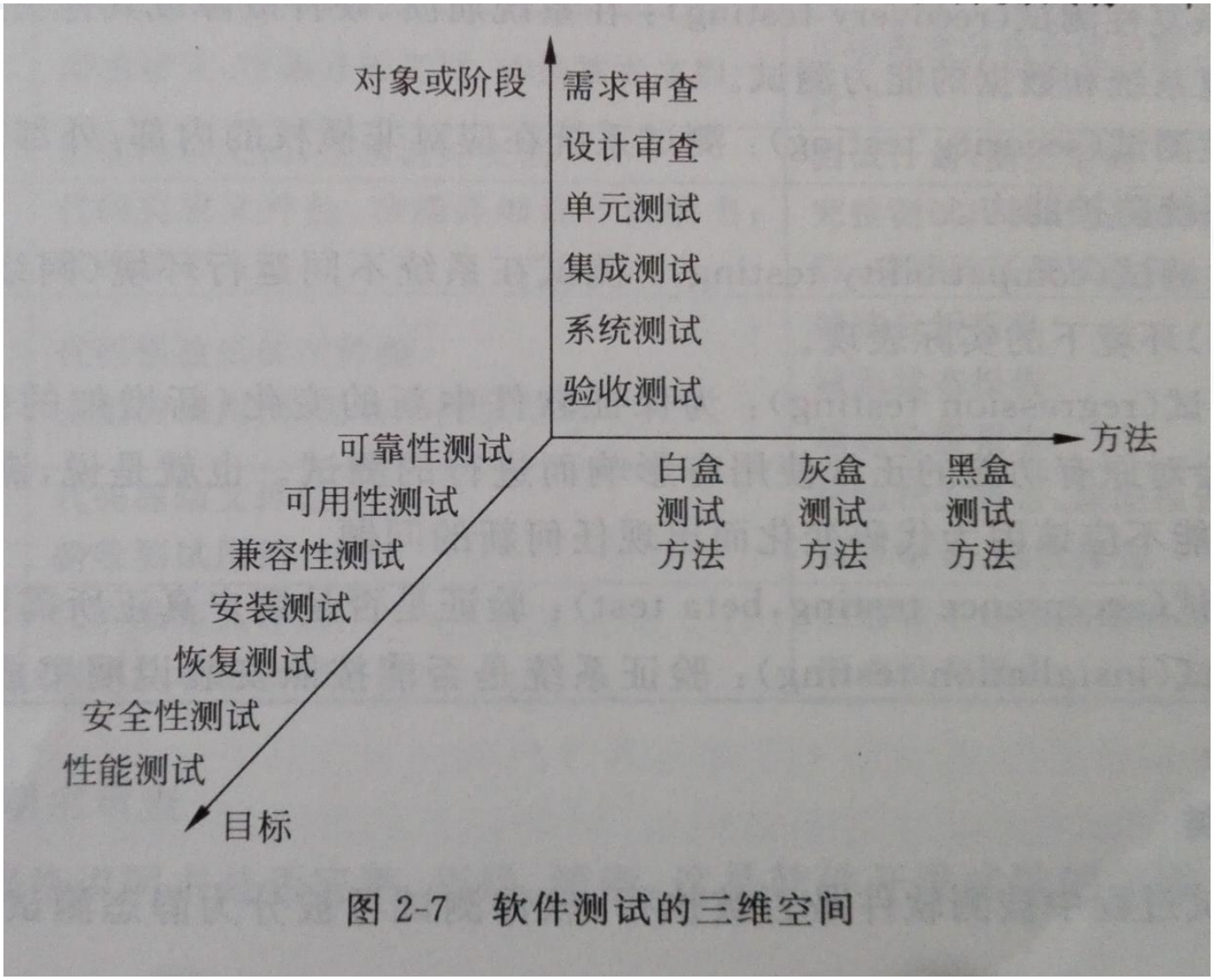
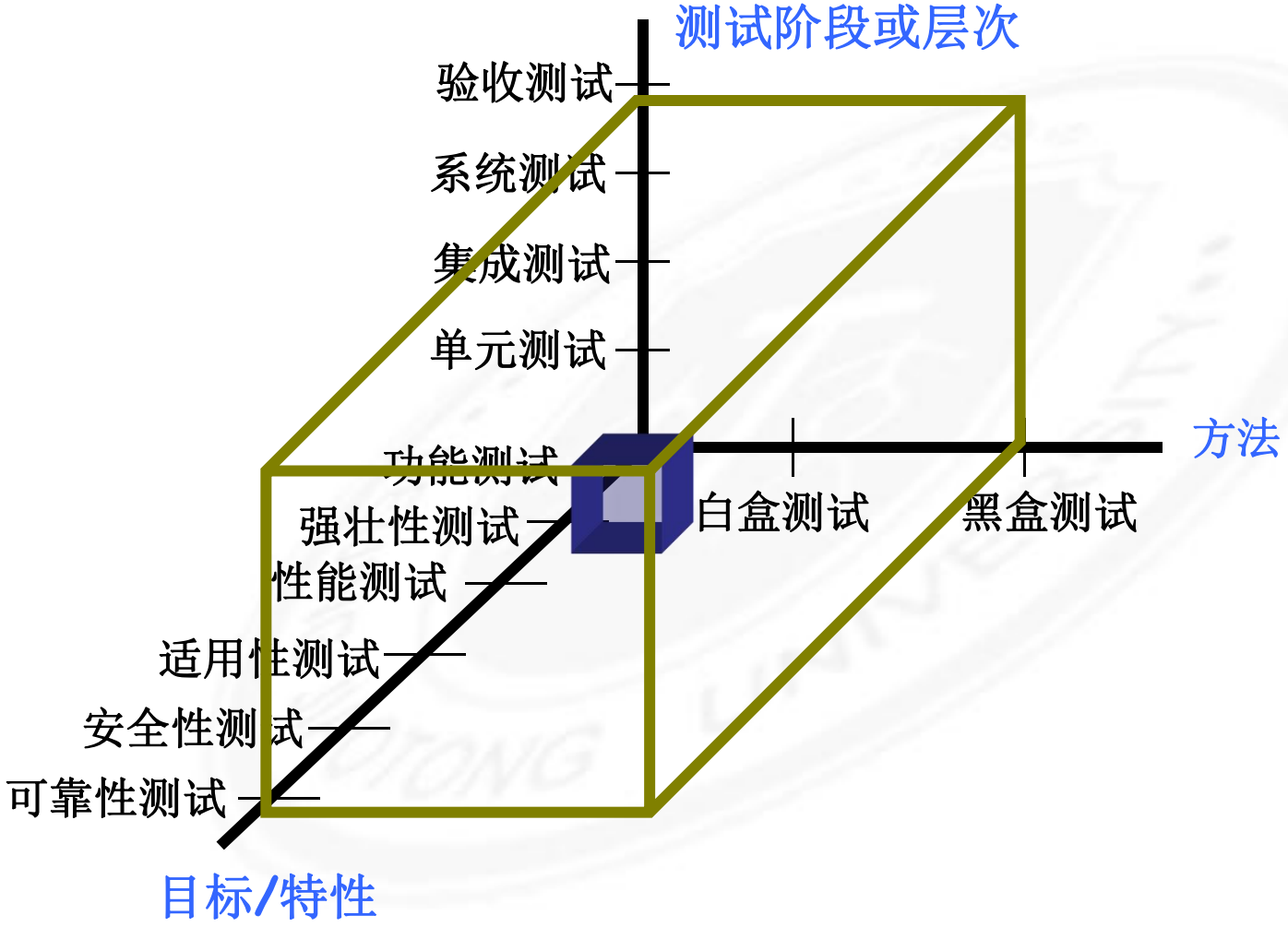


图 2-7　软件测试的三维空间


测试阶段或层次
验收测试
系统测试
集成测试
单元测试
功能测试
强壮性测试
性能测试
适用性测试
安全性测试
可靠性测试
白盒测试
黑盒测试
方法
目标/特性

2、软件测试相关国家标准

- GB/T25000.X《系统与软件工程 系统与软件质量要求和评价（SQuaRE）系列标准
- [1]GBT 25000.51-2016《系统与软件工程 系统与软件质量要求和评价（SQuaRE）第51部分：就绪可用软件产品（RUSP）的质量要求和测试细则》
- [2]GB/T 11457 信息技术 软件工程术语
- [3]GB/T 25000.10-2016系统与软件工程系统与软件质量要求和评价(SQuaRE) 第10部分：系统与软件质量模型（替代GB /T 16260.1-2006 ）
- [4]GB/T 25000.22-2019 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第22部分：使用质量测量（替代GB/T 16260.4-2006 ）
- [5]GB/T 25000.23-2019 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第23部分：系统与软件产品质量测量（替代GB/T 16260.2-2006 ，GB/T 16260.3-2006 ）
- [6]GB/T 25000.40-2018系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第40部分：评价过程（替代GB/T 18905.1-2002 ）
- [7]GB/T 25000.41-2018系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第41部分：开发方、需方和独立评价方评价指南（替代GB / T 18905.3- -2002 ，GB /T 18905.4- 2002 ，GB / T 18905.5- 2002 ）

GB/T 25000—2010标准
软件产品测试指标体系

图2-3 测试指标体系

[4]GB /T 16260.1-2006 软件工程 产品质量 第1部分：质量模型

图2-4 测试质量模型

GB/T 25000.10-2016：系统与软件质量模型

3、功能性测试

- 软件的功能需求指明软件必须执行的功能，定义系统的行为，即软件在某种输入条件下要给出确定的输出必须做的处理或转换。软件功能需求是“硬指标”，软件非功能需求是“软指标”，如软件产品的效率、可伸缩性、稳定性等。
- 功能性测试是基于软件产品功能方面的测试，可用于单元级别、集成级别和系统级别的测试。
- 功能性测试不关注程序的执行路径，仅关注功能是否实现。

3、功能性测试

- 功能测试，也称为黑盒测试或数据驱动测试。黑盒测试试图发现以下类型的错误：
  - 功能错误或遗漏
  - 界面错误
  - 数据结构或外部数据库访问错误
  - 初始化或终止错误
- 黑盒测试的优点：与软件实现无关；用例设计可与软件实现并行。
- 黑盒测试的缺点：用例冗余，可能有未测试的功能漏洞。

4、非功能性测试

- 非功能性测试主要是基于软件产品的性能、负载、可维护性、安全性、可靠性、可移植性、易用性等方面进行的测试。也包括产品是否遵循制定的标准、规范和约束。
- 典型非功能性测试有：
  - 压力测试
  - 性能测试
  - 文档测试
  - 可靠性测试
  - 配置、安装测试
  - 兼容性测试等

5、静态测试与动态测试

1）静态测试

- 静态测试不实际运行软件，主要是对软件的编程格式、结构等方面进行评估。
- 静态测试包括代码检查、静态结构分析、代码质量度量等。它可以由人工进行，也可以借助软件工具自动进行。
- 静态测试方法也可利用计算机作为对被测程序进行特性分析的工具，但与人工测试方式有着根本区别。另一方面，因它并不真正运行被测程序，只进行特性分析，这又与动态方法不同。
- 所以，静态测试方法常常称为“分析”，静态测试是对被测程序进行特性分析方法的总称。

5、静态测试与动态测试

- 静态测试阶段的任务：

（1）检查算法的逻辑、模块接口正确性。
（3）检查输入参数是否有合法性检查。
（4）检查调用其它模块的接口是否正确。
（5）检查是否设置了适当的出错处理。
（6）检查表达式、语句是否正确，是否含有二义性。
（7）检查常量或全局变量使用是否正确。
（8）检查标识符的使用是否规范、一致。
（9）检查程序风格的一致性、规范性。
（10）检查代码是否可以优化，算法效率是否最高。
（11）检查代码注释是否完整，是否正确反映了代码的功能。

5、静态测试与动态测试

2）动态测试

- 动态方法的主要特征：

  ——必须真正运行被测试的程序，通过输入测试用例，对其运行情况即输入与输出的对应关系进行分析，以达到检测的目的。

- 动态测试任务包括：

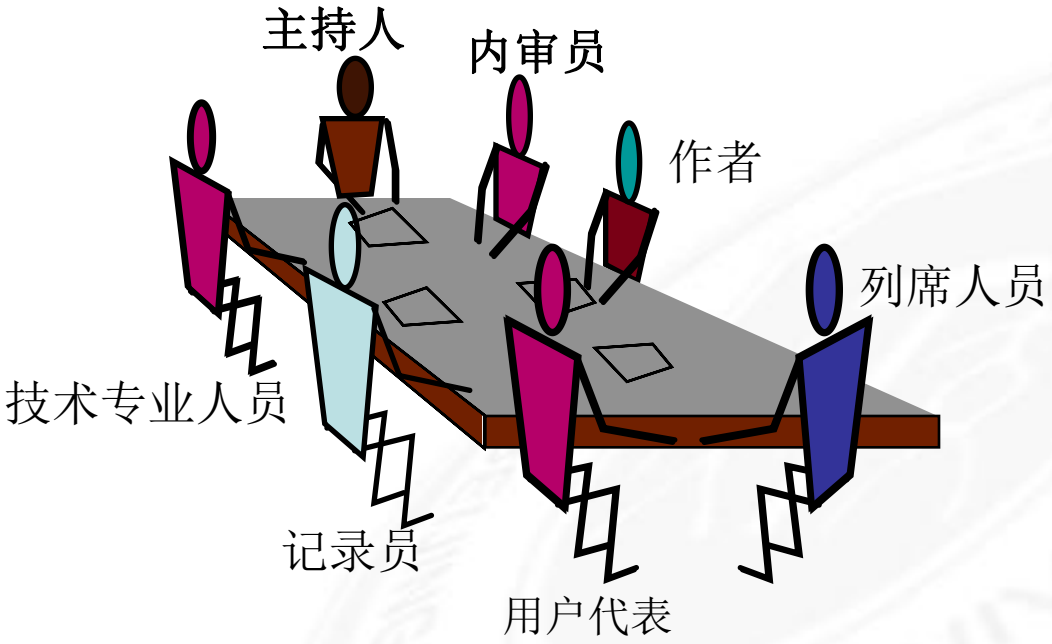
  （1）功能确认与接口测试

  （2）覆盖率分析

  （3）性能分析

  （4）内存分析

静态的和动态的



运行程序

不正式 正式

互审 走读 审查会议

6、黑盒测试和白盒测试

- 若测试规划是基于产品的功能，目的是检查程序各个功能是否能够实现，并检查其中的功能错误，则这种测试方法称为黑盒测试(Black-box Testing)方法。

——黑盒测试又称为功能测试、数据驱动测试或基于规格说明的测试。它是一种从用户观点出发的测试，一般被用来确认软件功能的正确性和可操作性。

6、黑盒测试和白盒测试

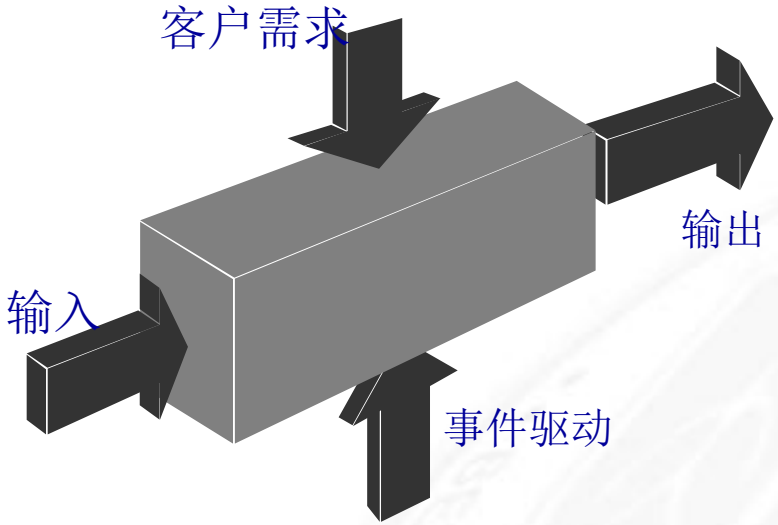
- 若测试规划基于产品的内部结构进行测试，检查内部操作是否按规定执行，软件各个部分功能是否得到充分使用，则这种测试方法称为白盒测试(White-box Testing)方法。

  ——白盒测试又称为结构测试、逻辑驱动测试或基于程序的测试，一般用来分析程序的内部结构。

6、黑盒测试和白盒测试

黑盒测试和白盒测试两种测试方法从完全不同的角度出发，反映了测试思路的两方面情况，适用于不同的测试阶段。

黑盒子和白盒子

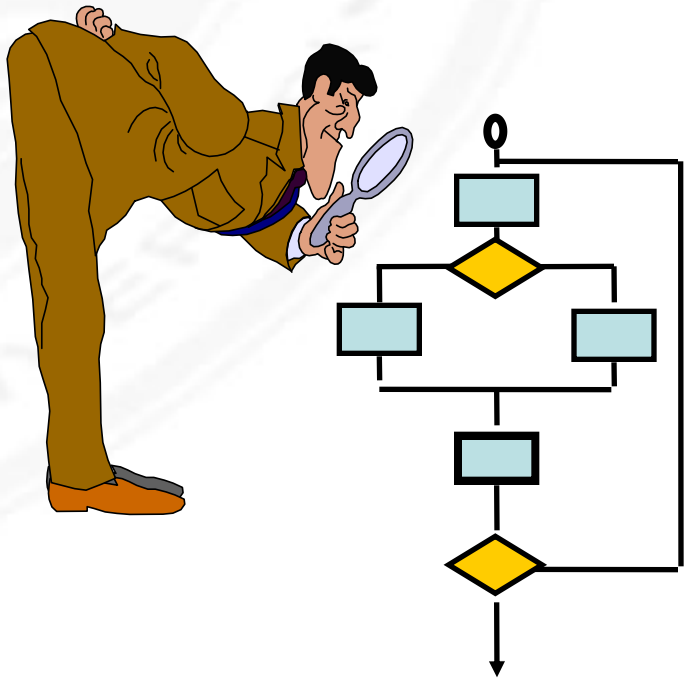


功能测试

数据驱动测试

结构测试

逻辑驱动测试



黑盒测试和白盒测试区别

项目	黑盒测试法	白盒测试法
规划方面	功能的测试	结构的测试
优点	能确保从用户的角度出发进行测试	能对程序内部的特定部位进行覆盖测试
缺点	无法测试程序内部特定部位；当规格说明有误，则不能发现问题	无法检查程序的外部特性； 无法对未实现规格说明的程序内部欠缺部分进行测试
应用范围	边界分析法 等价类划分法 决策表测试	语句覆盖，判定覆盖， 条件覆盖，判定/条件覆盖， 路径覆盖，循环覆盖， 模块接口测试

7、 形式化测试方法

- 为了解决基于自然语言的设计和描述所带来的问题，人们提出了形式化方法。形式化方法实际上就是用基于数学的方法来描述目标软件系统属性的一种技术。
- 凡是采用严格的数学语言、具有精确的数学语义的方法，都称为形式化方法。
- 形式化规范说明语言，一般由3个主要的成分构成：语法、语义和一组关系。

形式化的方法

- 形式化的方法可分为以下几类：
- 基于模型的方法，如Z语言、B语言等
- 代数方法，如OBJ、CLEAR、ASL、ACT等
- 过程代数方法，如CSP、CCS、ACP、LOTOS、TPCCS等
- 基于逻辑的方法，如区间时序逻辑、模态逻辑、时序逻辑、时序代理模型等。
- 基于网络的方法

形式化验证：

- 形式化验证，就是根据某些形式规范或属性，使用形式逻辑方法证明其正确性或非正确性。
- 一般通过形式化规范进行分析和推理，研究它的各种静态和动态性质，验证是否一致、完整，从而找出所存在的错误和缺陷。
- 无法证明某个系统没有缺陷，因为不能定义“没有缺陷”。只能证明一个系统不存在我们可以想得到的缺陷，以及验证满足系统质量要求的属性。

形式化验证的具体方法:

- 有限状态机（FSM）
- UML 语义转换
- 标准RBAC模型
- 符号模型检验
- BAN逻辑模型
- ..

基于模型的软件测试

- 基于模型的测试（Model-based testing，MBT）是利用模型来生成相应的测试用例，然后根据实际结果和原先预想的结果的差异来测试系统.
- 先从概念上形成模型，然后试图用数学的方法来描述这个模型，形成仿真模型，完成所需的测试.

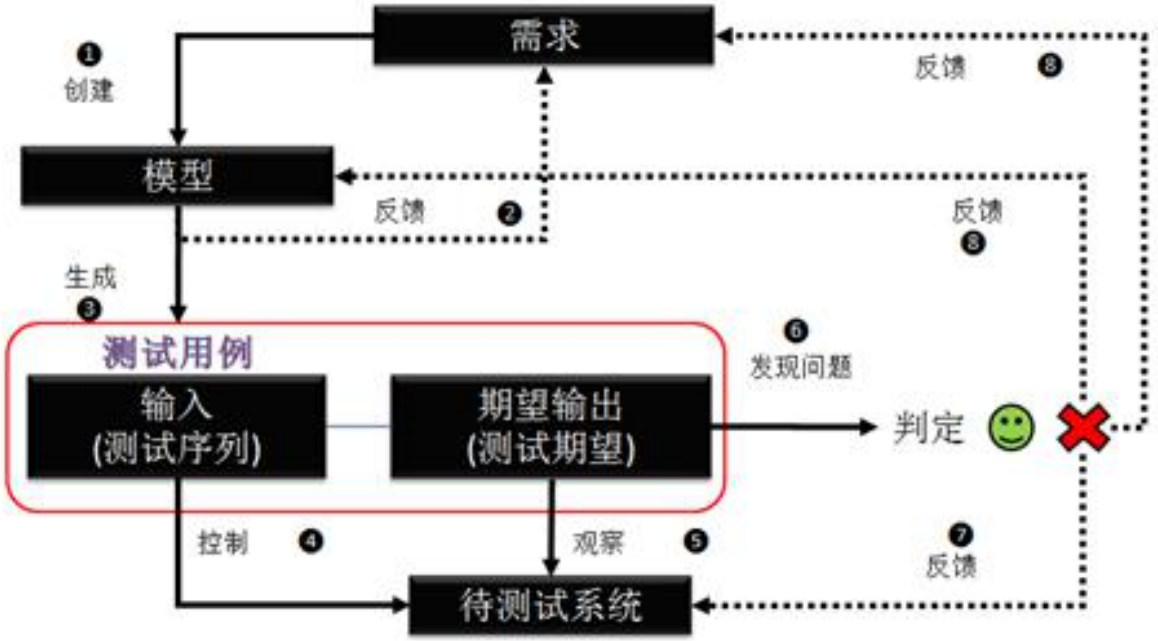


图2-5 基于模型的软件测试流程

8、模糊测试方法

- 模糊测试（Fuzz testing）方法，简单的说，就是构造大量的随机数据作为系统的输入，从而检验系统在各种数据情况下是否会出现问题。
- 模糊测试方法可模拟黑客对系统发动攻击测试，完成安全性测试，并能应用于服务器的容错性测试。
- **模糊测试**是一种通过向目标系统提供非预期输入并监视异常结果来发现软件漏洞的方法。

8、模糊测试方法

- 模糊测试是一个自动或半自动的过程，这个过程包括反复操纵目标软件并为其提供处理数据。模糊测试中的关键是模糊测试用例的生成方法，用于生成模糊数据的工具可称为模糊器。
- 模糊器可分为两大类：基于变异的模糊器和基于生成的模糊器。前者对已有数据样本应用变异技术创建新的测试用例；后者通过对目标协议或文件格式建模的方法从头开始产生测试用例。

8、模糊测试方法

- 模糊测试与传统测试技术对比：
- 白盒测试主要采用源代码审计的方法查找漏洞，白盒测试具有覆盖能力强的特点，但可能工作量大，复杂性高。
- 黑盒测试不需待测软件的源代码，操作简单，可用性好，但是测试用例多出自测试人员的主观猜想，覆盖能力较差。
- 灰盒测试是一种介于白盒测试和黑盒测试之间的测试方法，这种方法既包括黑盒测试审核,也包括通过逆向分析获取汇编形式的源代码.灰盒测试有很好的**可用性**，但却存在很大的复杂性。
- 模糊测试技术属于灰盒测试，经常用于软件漏洞分析。

- 与传统漏洞挖掘方法相比：
- 模糊测试的测试目标是二进制可执行代码，比基于源代码的白盒测试适用范围更广；
- 模糊测试是动态实际执行的，不存在静态分析技术中存在的大量误报问题；
- 模糊测试的原理简单，自动化程度高，不需要逆向工程中大量的人工参与。
- 模糊测试技术的这些优点都促使它成为一种应用范围广泛的高效软件漏洞挖掘技术。

2.4 软件测试过程

1、软件测试过程

2、测试过程实用模型

3、案例

1、软件测试过程

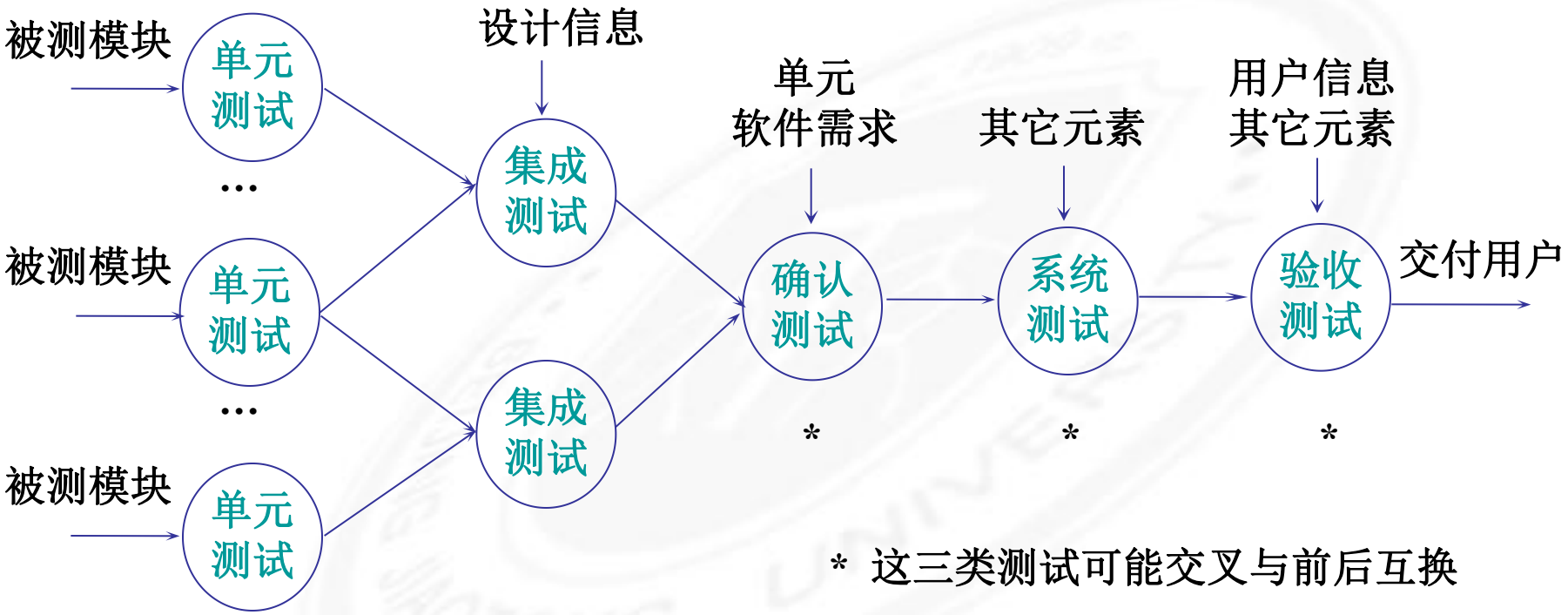


图2-6 软件测试过程的各阶段

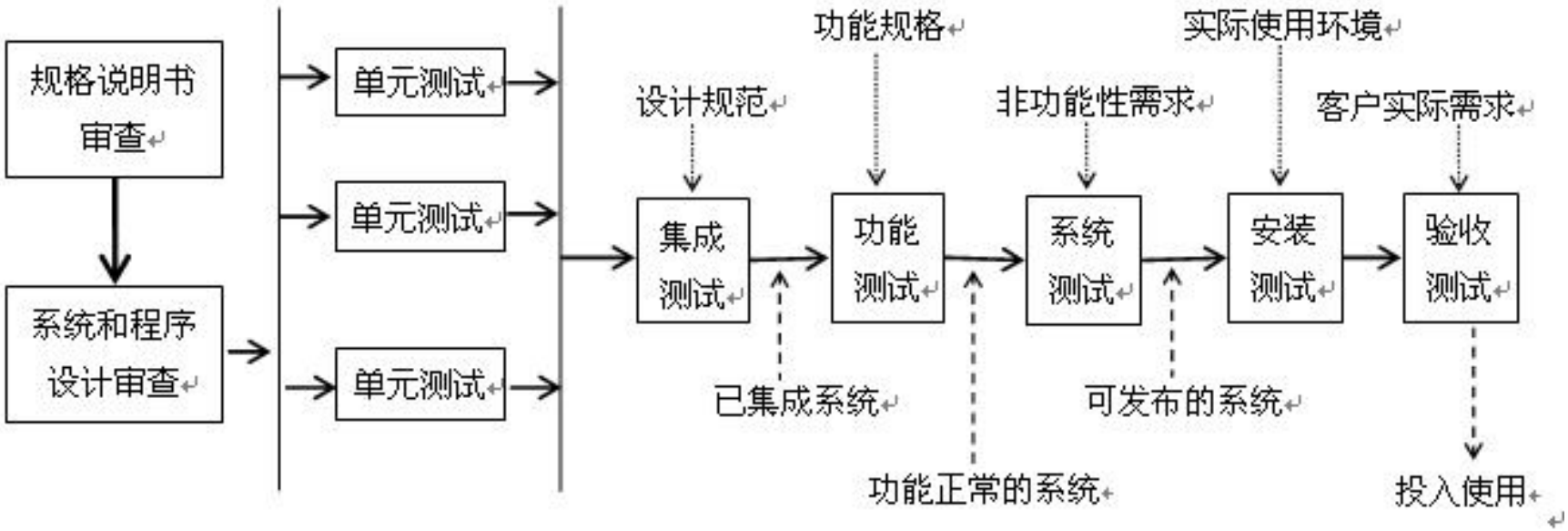
1、 软件测试过程

- 软件产品的测试通常要经过以下测试阶段：

1. 单元测试：针对每个单元的测试， 以确保每个模块能正常工作为目标。

- 集成测试：对已测试过的模块进行组装，进行集成测试。目的在于检验与软件设计相关的程序结构问题。
- 确认（有效性）测试：是检验所开发的软件能否满足所有功能和性能需求的最后手段。
- 系统测试：检验软件产品能否与系统的其他部分（比如，硬件、数据库及操作人员）协调工作。
- 验收（用户）测试：检验软件产品质量的最后一道工序。主要突出用户的作用，同时软件开发人员也应有一定程度的参与。

测试阶段（SDLC)



对于不同的软件系统或产品，测试的阶段可以适当裁减或合并，如，验收测试与安装测试可合并。

图2-7 软件测试过程的各阶段（细化）

2、 实用的软件测试过程模型

一种简单实用的软件测试过程模型 POCERM。

测试过程中必需的基本测试活动及其产生的结果：

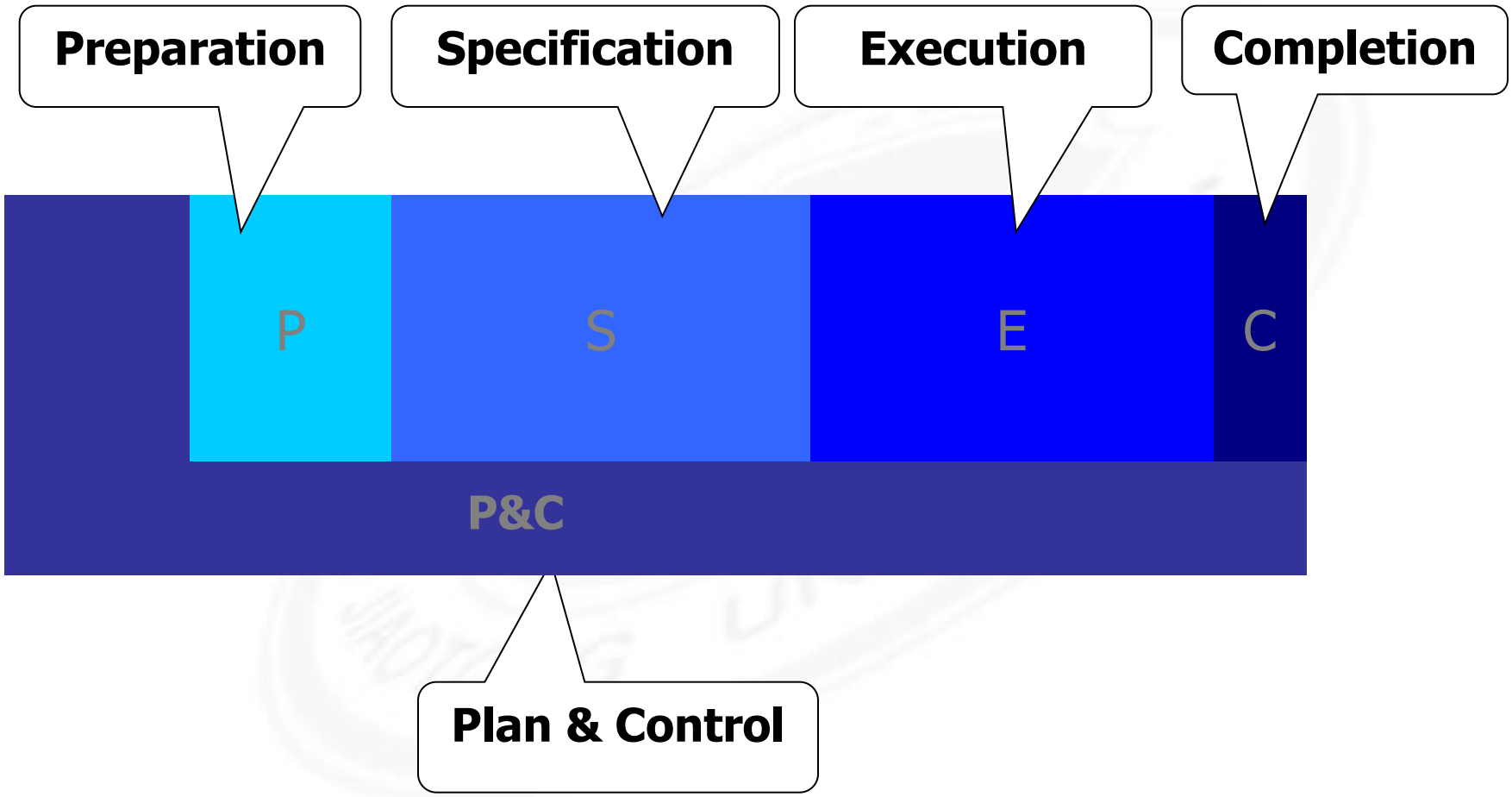
1. 拟定软件测试计划 (Plans)
2. 编制软件测试大纲 (Outlines)
3. 设计和生成测试用例 (test Case generation)
4. 实施测试 (Execution)
5. 生成软件测试报告 (software testing Reports)
   - 软件问题报告SPR (Software Problem Report)
   - 测试结果报告 (test result Reports)

测试阶段

测试过程的三个主要的测试活动（计划、准备和实施） 可被分成五个阶段：

- 1.计划和控制阶段The planning and control phase
- 2.准备阶段The preparation phase
- 3.规范阶段The specification phaseThe
- 4.实施执行阶段execution phaseThe
- 5.完成阶段completion phase

测试的五个阶段


Preparation
Specification
Execution
Completion
P
S
E
C
P&C
Plan & Control

1）计划与控制阶段

- 它是整个测试过程中最重要的阶段，为实现可管理且高质量的测试过程提供基础 。
- 本阶段的主要工作内容：

（1）拟定测试计划

（2）论证那些使开发过程难于管理和控制的因素

（3）明确软件产品的最重要部分（风险评估）

2）准备阶段

- 开始本阶段的前提条件：
  - —完成测试计划的拟定。
  - —需求规格说明书（版本）的确定。
- 本阶段的主要工作内容：
  - —对需求规格说明书的仔细研究。
  - —将要测试的产品分解成可独立测试的单元。
  - —为每个测试单元确定采用的测试技术。
  - —为测试的下一个阶段及其活动制定计划。

3）规范(设计)阶段

- 本阶段的主要工作内容：
  - —编写测试大纲/测试用例，测试脚本
  - —搭建测试环境
    （测试数据库，软件环境，硬件环境）
- 测试用例描述的内容：
  - —输入
  - —执行过程
  - —预期输出

4）实施执行阶段

- 根据测试大纲/测试用例/测试脚本进行测试

（1）根据测试大纲/测试用例进行测试，找出预期的测试结果和实际测试结果之间的差异；

（2）填写软件问题报告；

（3）确定造成这些差异的原因：

- 产品有缺陷？规格说明书有缺陷？
- 测试环境和测试下属部件有缺陷？测试用例设计不合理？
- 测试问题报告→与管理层进行沟通的方式
- 已测试部分占产品多大的百分比？还有什么工作要做？
- 发现了多少个问题或不足？测试的发展趋势如何？
- 测试可以结束了吗？

5）完成阶段

- 本阶段的主要工作内容：

—选择和保留测试大纲、测试用例、测试结果、测试工具。

—编制、审核、提交测试报告。

- 工作的意义：

—产品如果升级或功能变更，或维护，只要对保留下来的相关测试数据作相应调整，就能够进行新的测试。

测试的工作流程

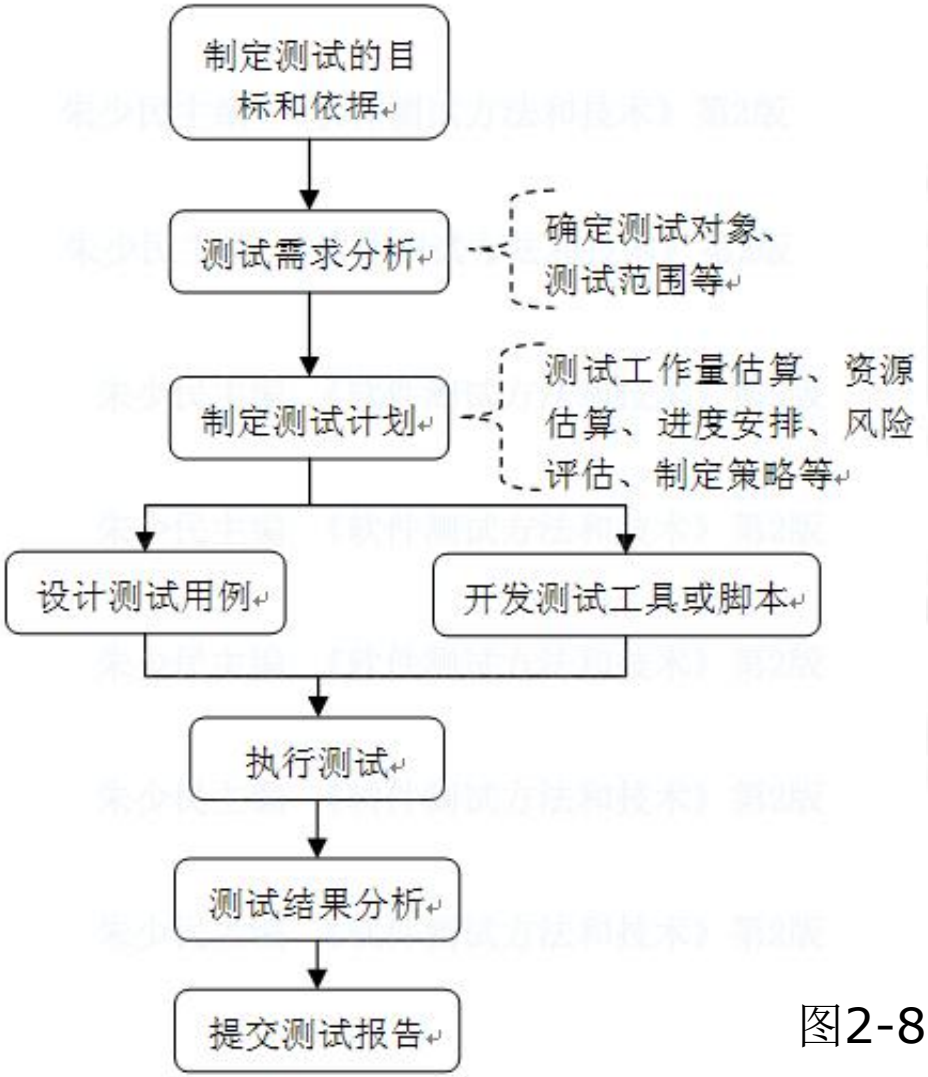

制定测试的目标和依据
测试需求分析
确定测试对象、测试范围等
制定测试计划
测试工作量估算、资源估算、进度安排、风险评估、制定策略等
设计测试用例
开发测试工具或脚本
执行测试
测试结果分析
提交测试报告

图2-8

3、 案例--箱管系统验收测试所需资料

1、建议用户提供的文档

- 系统需求说明书（最新版）
- 系统技术总体方案（或开发方的投标文件）
- 用户操作手册（最新版）
- 系统安装手册（最好有）

案例--箱管系统验收测试所需资料(续)

2、性能测试强度估算所需数据

- 目前用户最大车站使用本系统的用户数量、各岗位中用户数最大的岗位？用户数及对应主要业务是什么？
- 用户可接受的系统响应时间？
- 系统每天总业务笔数或期望系统每秒交易处理能力？
- 系统有哪些主要交易任务 ？
- 高峰期主要有哪些操作？
- 中间件操作有多少？数据库操作有多少？
- 如果任务失败，那么商业风险有多少？
- 系统未来发展状况如何？如每年业务增长情况。。
- 系统网络容量规划
- ….

进度计划实例

测试阶段	工作时间	参与方
提交文档	1工作日	业主方提交，开发方协助
测试方案	2 工作日	测试方制订，开发方协助
测试方案评审	1 工作日	业主方、测试方、开发方、监理方均参与
测试规范	4 工作日	测试方制订、开发方协助
测试规范评审	1 工作日	测试方质量控制组评审
测试执行	6 工作日	测试方执行、开发方协助、业主方监督
测试问题提交和确认	2 工作日	测试方提交、开发方确认、业主方监督
问题修改	6 工作日	开发方修改、业主方监督
回归测试	3 工作日	测试方执行、开发方协助、业主方监督
最终测试报告	7 工作日	测试方提交
合计	33工作日	

测试计划(实例)

1. 项目计划阶段：依据用户针对本项目制定的项目目标、项目原则等要求，参考项目技术文档，制定该项目的测试计划，并根据用户意见修订测试计划，在通过评审后进入项目设计阶段；
2. 项目设计阶段：对本项目进行详细的测试需求分析后编写测试方案和测试规范（设计），组织相应的评审工作，并且准备测试数据以及确认现场测试环境；
3. 项目实施阶段：按照测试计划和测试设计，负责完成项目的执行，包括对被测软件的功能、安全性、性能等质量特性的测试，并记录测试结果；
4. 报告结果分析阶段：根据系统的测试情况进行项目的评估与总结，测试报告编写后将与用户进行沟通和进一步的完善，并通过测试报告的评审。

测试大纲(实例)(1)

1. 测试概述
   - 项目概述
   - 测试目的和意义
   - 测试原则
   - 测试依据
   - 测试流程
2. 测试计划
   - 测试进度安排
   - 项目测试阶段划分
   - 项目测试分工
3. 组织结构
   - 测试项目组的构成
   - 测试人员岗位职责
   - 测试投入人员

测试大纲(实例)(2)

- 4 测试内容及方法
  - 4.1 功能度测试
  - 4.2 性能测试
  - 4.3 安全可靠性测试
  - 4.4 易用性测试
  - 4.5 兼容性测试
  - 4.6 可扩充性测试
  - 4.7 用户文档检查
- 5 评测结果
  - 5.1 问题等级
  - 5.2 测试结果
- 6 测试环境

问题报告模版(例)

产品名称		版本号	
开发单位		联系人	
测试过程	根据测试规范及用户手册，采用相应的测试用例及工具逐项进行测试。		
问题描述	问题详细描述见附页		

问题分析	问题总数量	？、	严重问题数量	？	一般问题数量	？	建议数量	？
	严重问题？个，一般问题？个，建议问题？个；							

修改意见	▪严重性问题必须修改 ▪一般性问题尽量修改		
		测试工程师	
		审核签字	
		签署日期	

修改确认			
	修改结果： 已完成修改数量：严重：1 一般：8 建议：5 未完成修改数量：严重：0 一般：0 建议：2 达到要求的程度：严重：100% 一般：100% 建议：71.43%	厂商	
		测试工程师	
		审　核	
		报告日期	

2.5 单元测试与集成测试

2.5.1 单元测试

2.5.2 集成测试

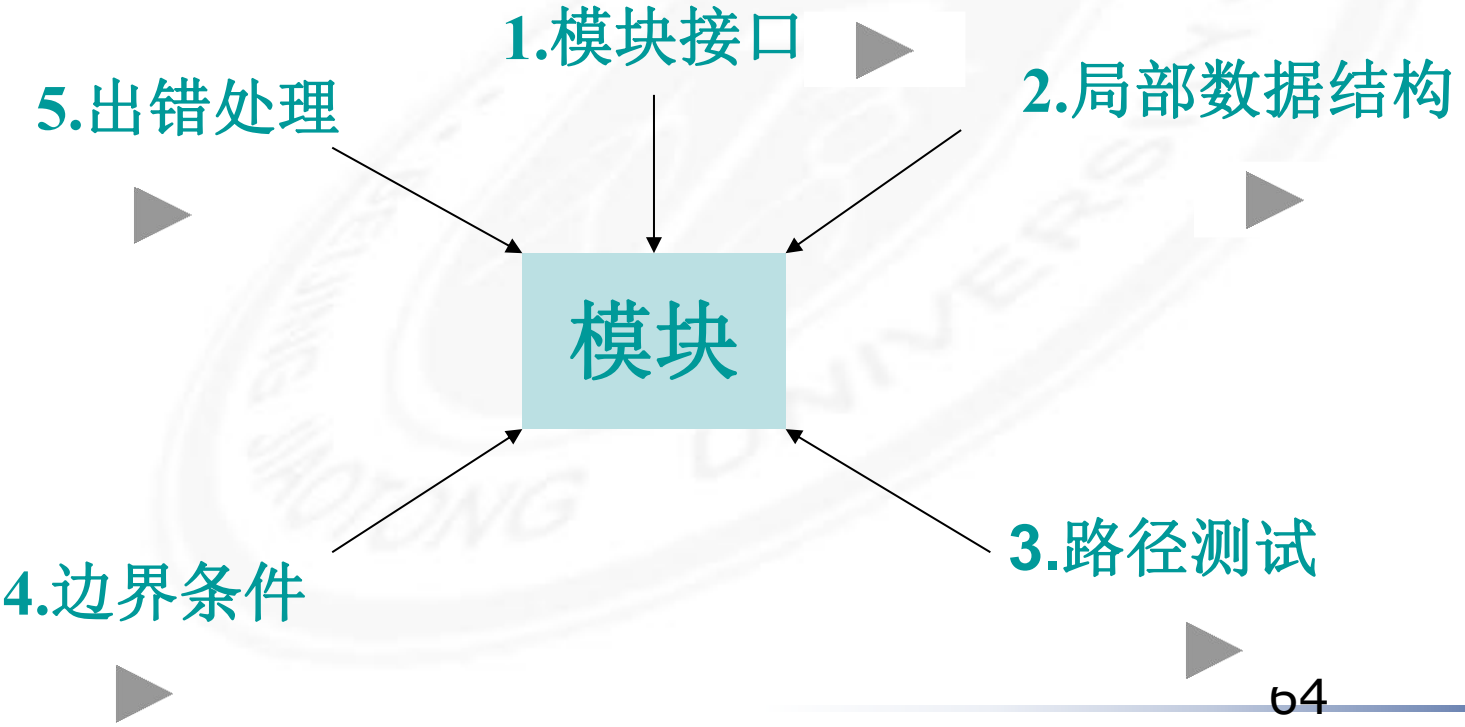
2.5.1 单元测试

- 1、单元测试任务
- 单元测试是针对各个代码单元进行的测试。测试仅围绕具体的程序模块或类进行。
- 单元测试是软件开发过程中要进行的最低级别的测试活动，或者说是针对软件设计的最小单位即程序模块、函数、类或方法所进行的正确性检验的测试工作。
- 其目的是发现每个单元内部可能存在的缺陷。

2.5.1 单元测试

- 1、单元测试任务
- 单元测试主要测试5个方面的问题：

——模块接口、局部数据结构、边界条件、独立的路径和错误处理。



1、单元测试任务（续）

模块接口：

- 是对模块接口进行的测试，检查进出程序单元的数据流是否正确。模块接口测试必须在任何其它测试之前进行。
- 模块接口测试至少需要如下的测试项目：

（1）调用所测模块时的输入参数与模块的形式参数在个数、属性、顺序上是否匹配；

（2）所测模块调用子模块时，它输入给子模块的参数与子模块中的形式参数在个数、属性、顺序上是否匹配；

（3）是否修改了只做输入用的形式参数；

（4）调用标准函数的参数在个数、属性、顺序上是否正确；

（5）全局变量的定义在各模块中是否一致。

1、单元测试的任务（续）

局部数据结构：

- 测试过程中，必须测试模块内部的数据能否保持完整性，包括内部数据的内容、形式及相互关系不发生错误。
- 对于局部数据结构，应该注意发现以下几类错误：

（1）不正确的或不一致的类型说明。

（2）错误的初始化或默认值。

（3）错误的变量名，如拼写错误或书写错误。

（4）下溢、上溢或者地址错误。

1、单元测试的任务（续）

路径测试：

- 单元测试中的重点是针对路径的测试。测试用例必须能够发现由于计算错误、不正确的判定或不正常的控制流而产生的错误。
- 常见的错误有：

  误解的或不正确的算术优先级；混合模式的运算；错误的初始化；精确度不够精确；表达式的不正确符号表示。
- 针对判定和条件覆盖，测试用例还要能够发现如下错误：

  不同数据类型的比较；不正确的逻辑操作或优先级；应当相等的地方由于精确度的错误而不能相等；不正确的判定或不正确的变量；不正确的或不存在的循环终止；当遇到分支循环时不能退出；不适当地修改循环变量。

1、单元测试的任务（续）

边界条件：

- 边界测试必须采用边界值分析方法来设计测试用例，认真仔细地测试为限制数据处理而设置的边界处，看模块是否能够正常工作。
- 一些可能与边界有关的数据类型如数值、字符、位置、数量、尺寸等，还要注意这些边界的首个、最后一个、最大值、最小值、最长、最短、最高、最低等特征。
- 在边界条件测试中，应设计测试用例检查以下情况：

（1）在n次循环的第0次、1次、n次是否有错误。

（2）运算或判断中取最大值、最小值时是否有错误。

（3）数据流、控制流中刚好等于、大于、小于确定的比较值是否出现错误。

1、单元测试的任务（续）

出错处理

- 测试出错处理的重点是:当模块在工作中发生了错误，其出错处理设施是否有效?
- 检验程序中的出错处理是否有以下有：

（1）对运行发生的错误描述难以理解；

（2）所报告的错误与实际遇到的错误不一致；

（3）出错时，在错误处理之前就引起系统的干预；

（4）例外条件的处理不正确；

（5）提供的错误信息不足，以至于无法找到错误的原因。

2、单元测试的执行过程

- 何时进行单元测试？
- 单元测试常常是和代码编写工作同时进行，在完成了程序编写、复查和语法正确性验证后，就应进行单元测试用例设计。
- 在单元测试时，如果模块不是独立的程序，需要设置一些辅助测试模块。辅助测试模块有两种：

（1）驱动模块(Drive)　用来模拟被测试模块的上一级模块，相当于被测模块的主程序。它接收数据，将相关数据传送给被测模块，启动被测模块，并输出出相应的结果。

（2）桩模块(Stub)　用来模拟被测模块工作过程中所调用的模块。它们一般只进行很少的数据处理。

- 驱动模块和桩模块都是额外的开销，虽然在单元测试中必须编写，但并不需要作为最终的产品提供给用户。

2、单元测试的执行过程（续）

被测模块、驱动模块和桩模块共同构成了的单元测试环境

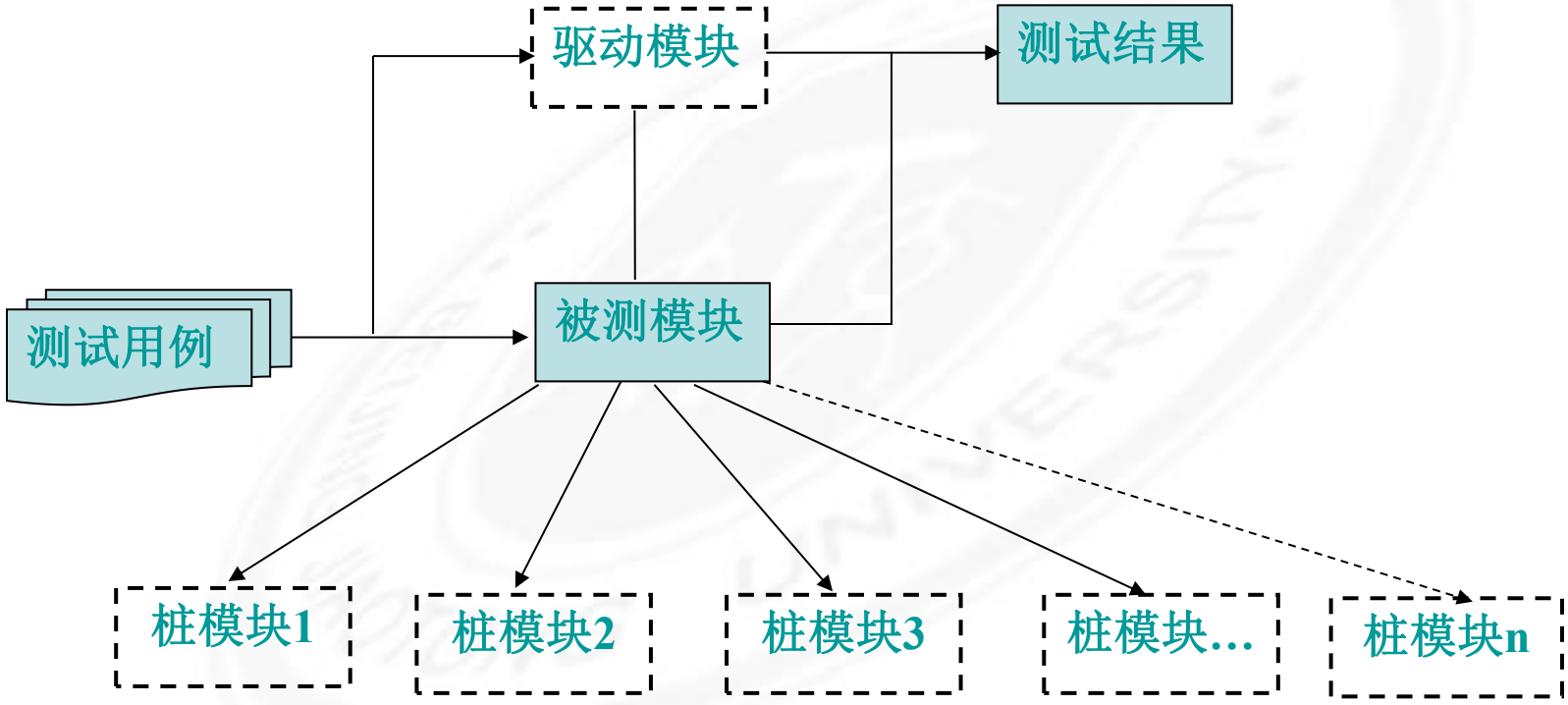


图2-9 单元测试场景

2.5.2 集成测试

1. 集成测试概念
2. 非增量式集成测试
3. 增量式集成测试
4. 集成测试方法比较
5. 回归测试

2.5.2 集成测试

1、集成测试定义

- 集成测试是在单元测试的基础上将几个模块按照概要设计的要求组装成子系统或系统的测试，是对模块间接口或系统接口以及集成后的子系统或系统的功能进行正确性检验的一项测试工作。
- 集成测试的主要工作是把通过单元测试的模块或类逐步集成在一起，来测试数据是否能够在各模块或类之间正确流动，以及各模块或类能否正确同步？

2.5.2 集成测试

- 集成测试的层次
- 对于结构化技术开发的软件系统而言，按照集成粒度不同，可分为4个层次：

① 模块内集成：当模块内部包括不同的函数或过程时可能需要模块内集成。

② 子系统内集成：当子系统由不同的模块组成，所以需要模块之间的集成。集成时以模块结构图为依据。

③ 子系统之间的集成：如果系统包括几个相互独立的子系统，需要子系统之间的集成测试。集成时以软件结构图为依据。

④ 不同系统之间集成。

2.5.2 集成测试

集成测试的层次

- 对于使用面向对象技术开发的软件系统而言，按照集成粒度不同，可分为以下4个层次：

① 类内集成：对一个类内部的不同方法进行集成，可以依据类状态进行集成。

② 类间集成：类实例化后，类之间有消息传递，类间集成可以序列图或协作图为依据。

③ 子系统之间的集成：同上

④ 系统之间的集成：同上

集成测试环境

集成测试环境

- 硬件环境
- 操作系统环境
- 数据库环境
- 网络环境
- 测试工具环境
- 开发驱动器和桩
- 其它环境

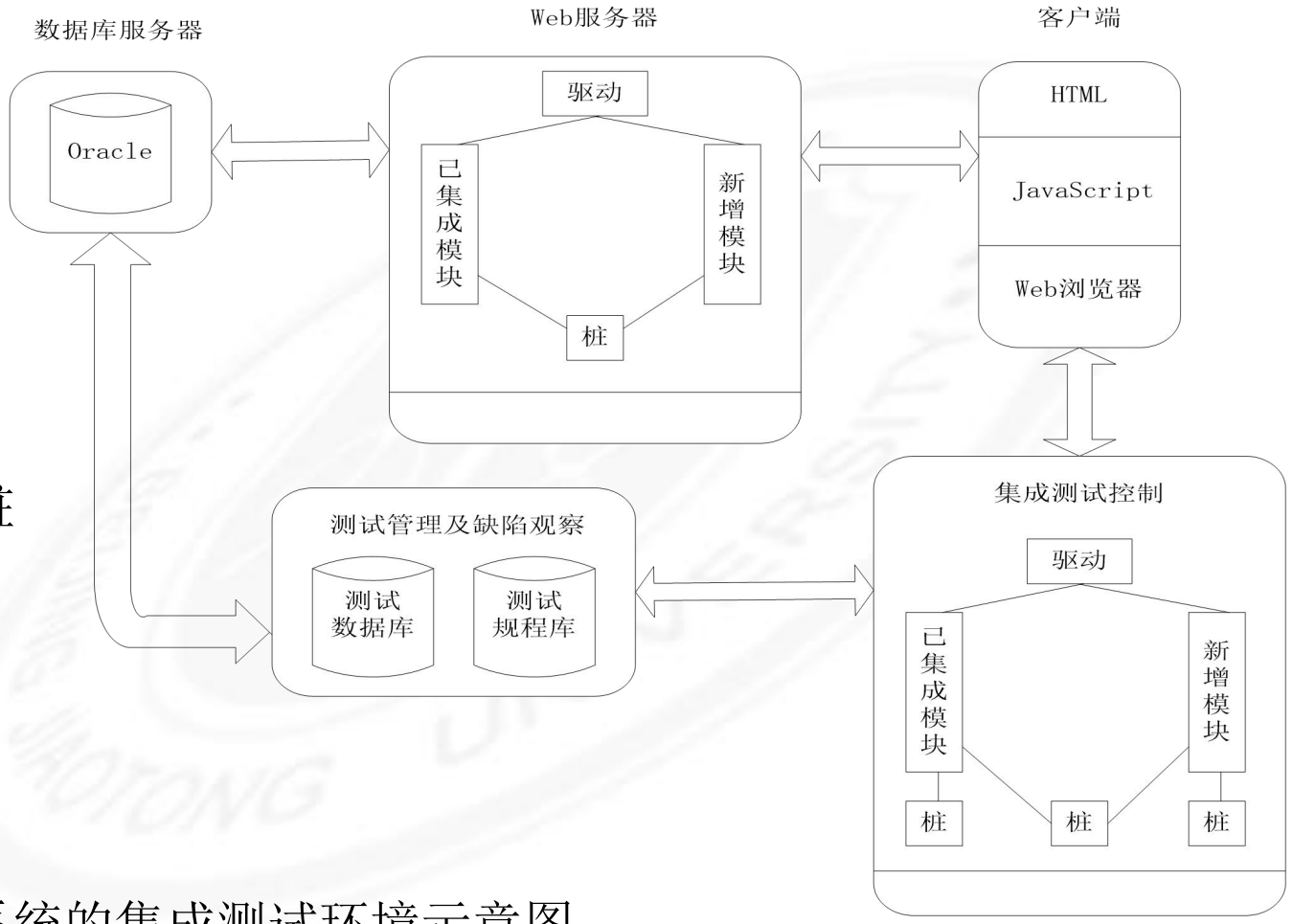


图2-10 一个系统的集成测试环境示意图

2. 非增量式集成测试

- 集成模式是软件集成测试中策略体现，直接关系到测试效率、结果等。一般根据系统特点来决定采用哪种方式。
- 集成测试的模式可分为：非增量式集成测试和增量式集成测试。
- 非增量式测试是采用一步到位的方法来构造测试：

——对所有模块进行个别的单元测试后，按照程序结构图将各模块连接起来，把连接后的程序当作一个整体进行测试。

实例　采用非增量式测试方法进行集成测试

2. 非增量式集成测试

- 非增量式测试的缺点：

  ——当一次集成的模块较多时，非增量式测试容易出现混乱，因为测试时可能发现了许多故障，每一个故障定位和纠正非常困难，并且在修正一个故障的同时，可能又引入了新的故障，新旧故障混杂，很难判定出错的具体原因和位置。

非增量式测试（续）

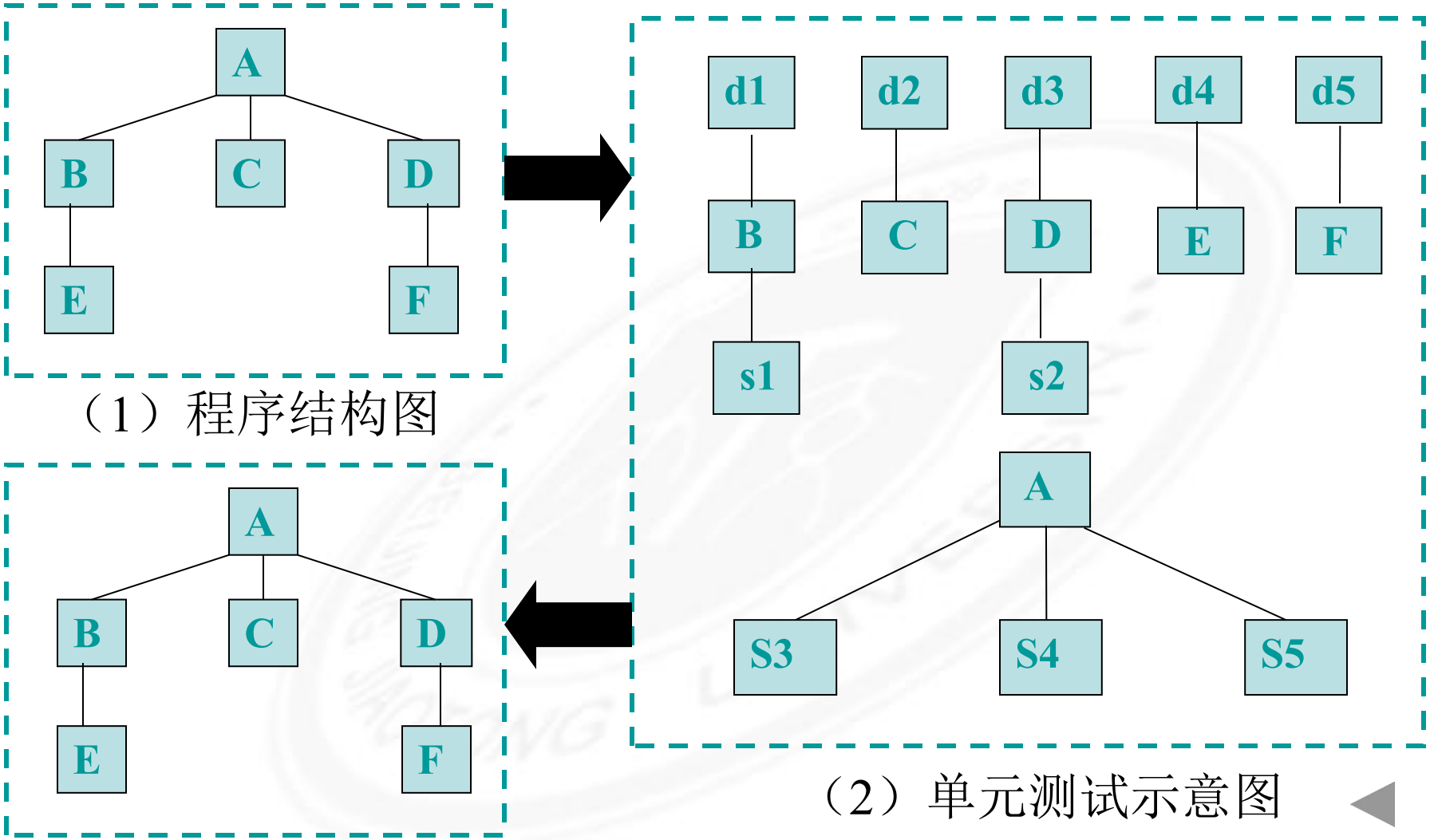


图2-11

3、增量式测试

- 增量式测试：

  ——其集成是逐步实现的，即逐次将未曾集成测试的模块和已经集成测试的模块（或子系统）结合成程序包，再将这些模块集成为较大系统，在集成的过程中边连接边测试，以发现连接过程中产生的问题。

- 按照不同的实施次序，增量式集成测试又可以分为三种不同的方法：

  （1）自顶向下增量式测试

  （2）自底向上增量式测试

  （3）混合增量式测试（三明治集成测试）

（1）自顶向下增量式测试

- 自顶向下增量式测试，表示逐步集成和逐步测试是按照结构图自上而下进行的，即模块集成的顺序是首先集成主控模块（主程序），然后依照控制层次结构向下进行集成。
- 从属于主控模块的可按深度优先方式（纵向）或者广度优先方式（横向）集成到结构中去。
- 深度优先方式的集成：

  ——首先集成在结构中的一个主控路径下的所有模块，主控路径的选择是任意的。
- 广度优先方式的集成：

  ——首先沿着水平方向，把每一层中所有直接隶属于上一层的模块集成起来，直到底层。

（1）自顶向下增量式测试（续）

- 集成测试的整个过程由3个步骤完成：

（1）主控模块作为测试驱动器。

（2）根据集成的方式（深度或广度），下层的桩模块一次一次地被替换为真正的模块。

（3）在每个模块被集成时，都必须先进行单元测试。

重复第2步，直到整个系统被测试完成。

实例　按照广度优先方式进行集成测试 ▶

实例　按照深度优先方式进行集成测试 ▶

自顶向下增量式测试（续）

广度优先方式

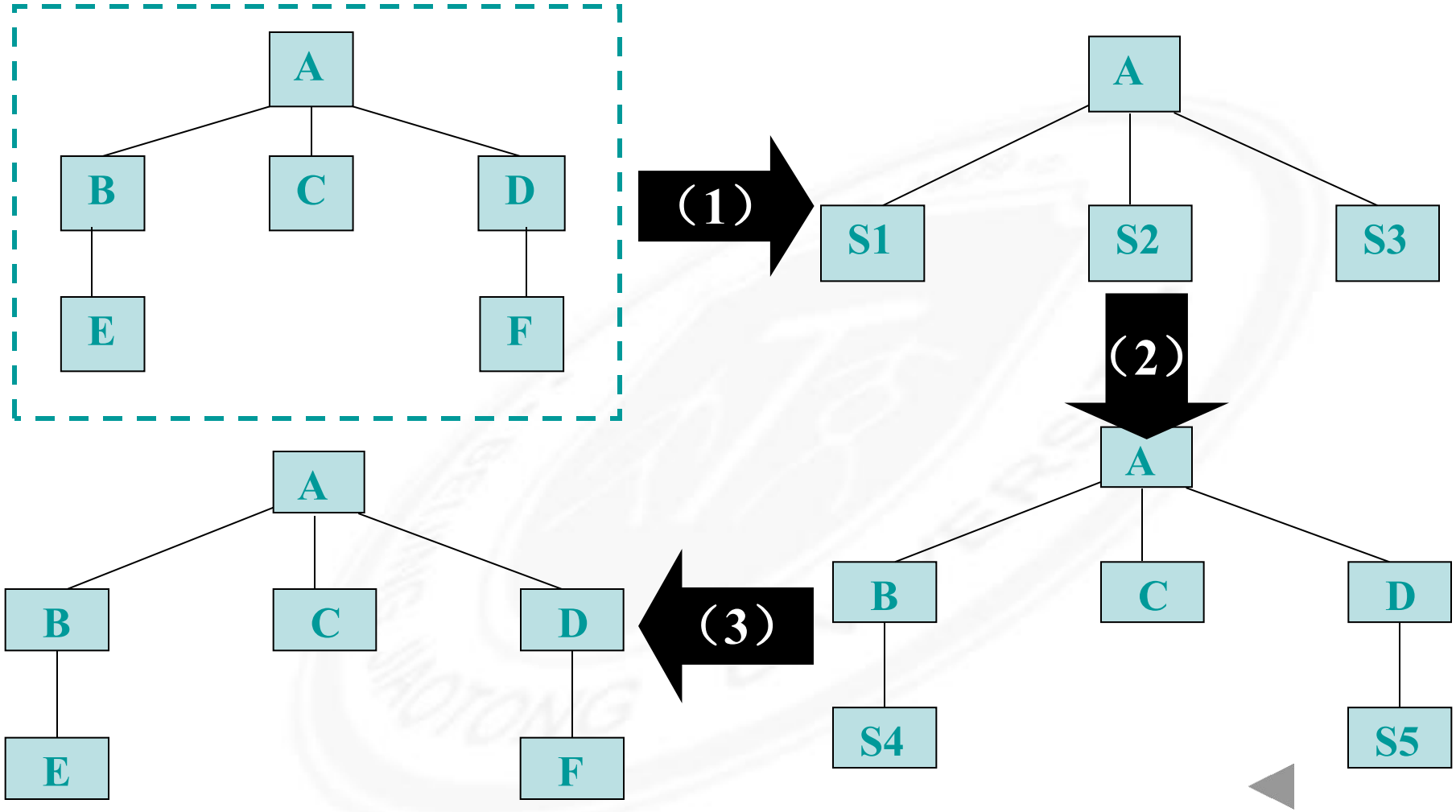


图2-12

自顶向下增量式测试（续）

深度优先方式

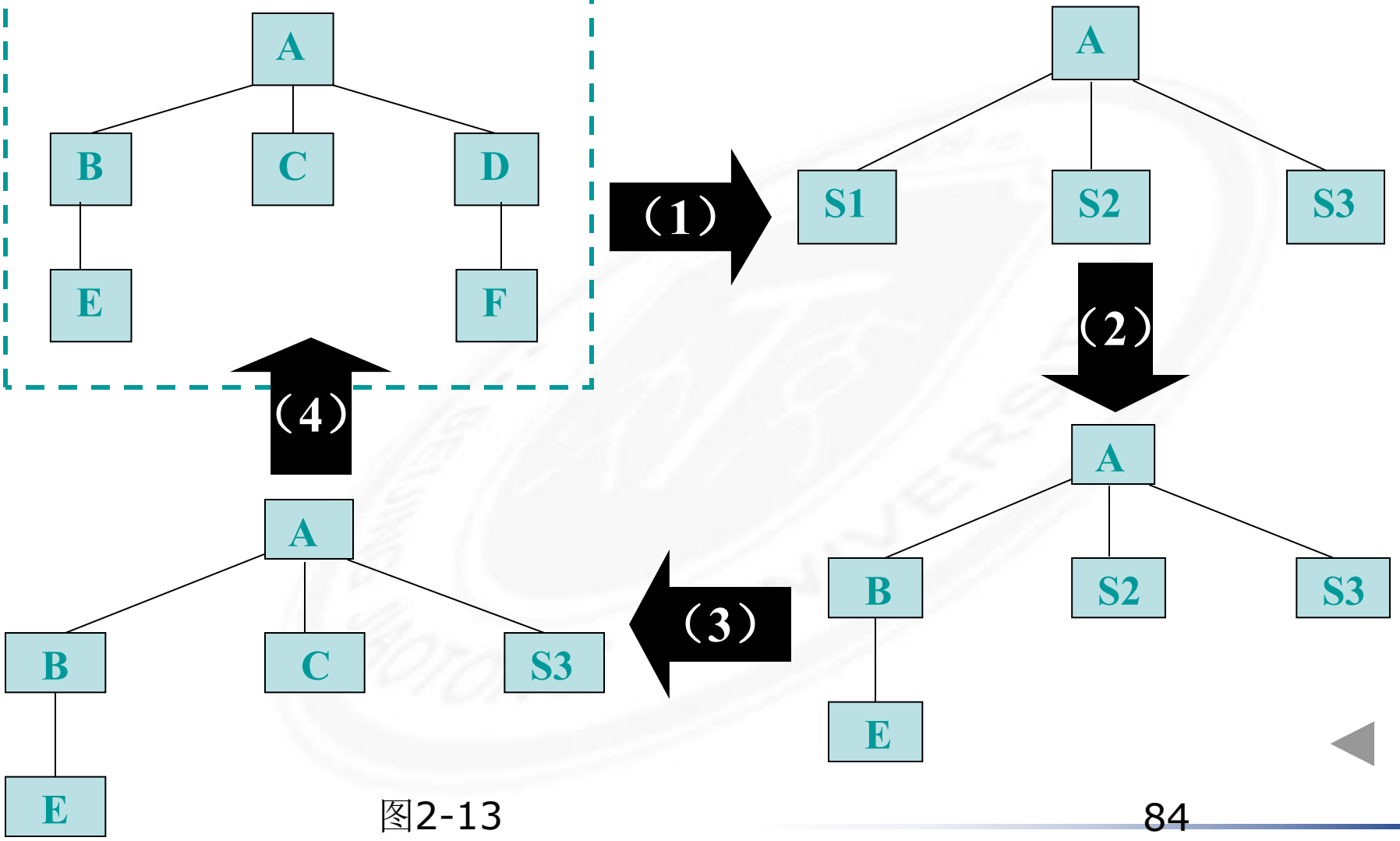


图2-13

（2）自底向上增量式测试

- 自底向上增量式测试，表示测试工作是按结构图自下而上进行的，即从程序模块结构的最底层模块开始集成和测试。
- 由于是从最底层开始集成，对于一个给定层次的模块，它的子模块（包括子模块的所有下属模块）已经集成并测试完成，所以不再需要使用桩模块进行辅助测试。在模块的测试过程中需要从子模块得到的信息可以直接运行子模块得到。

实例　采用自底向上增量式测试方法

自底向上增量式测试（续）

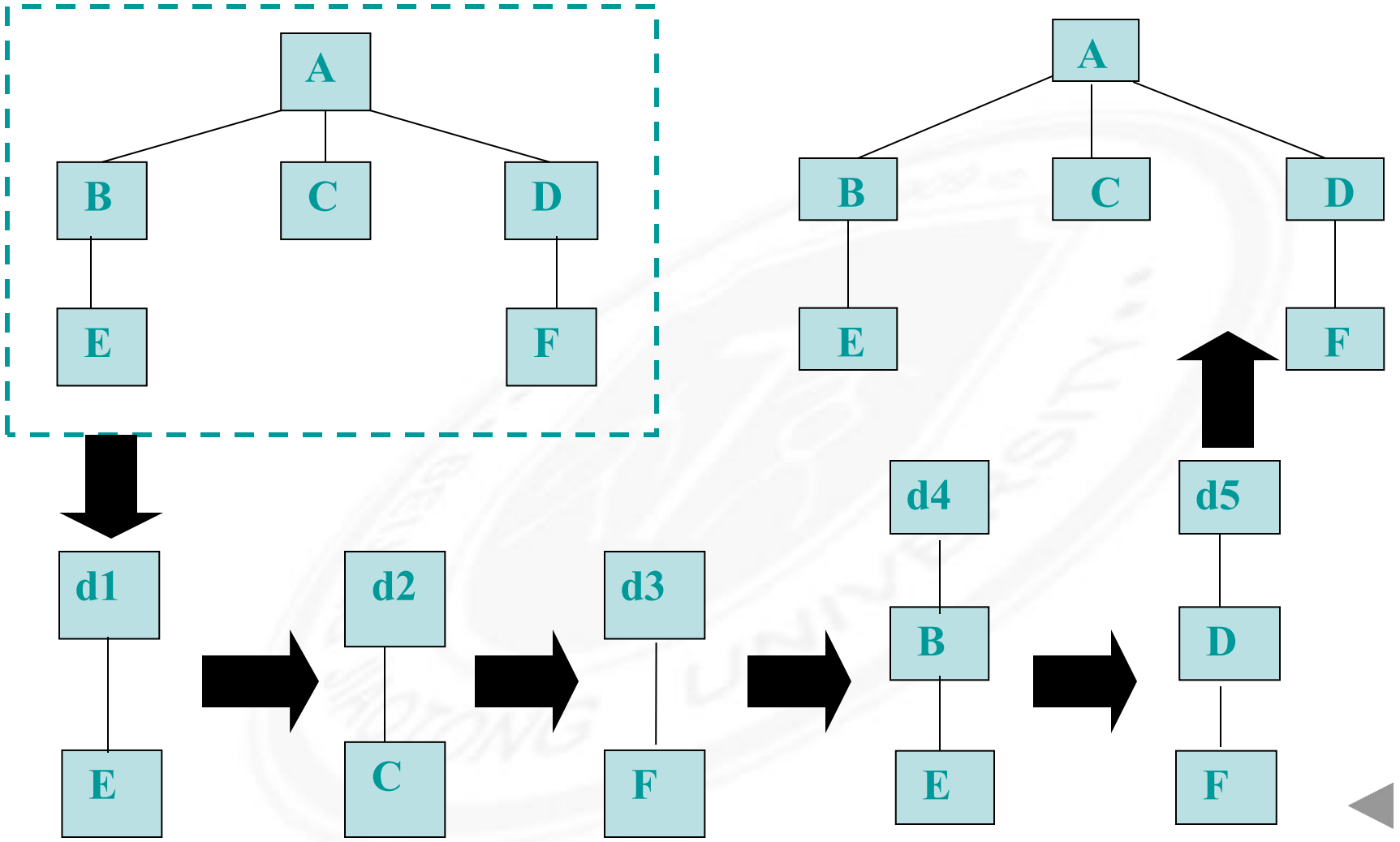


图2-14

（3）混合增量式测试

- 混合增量式测试是把自顶向下测试和自底向上测试这两种方式结合起来进行集成和测试。
- 常见的两种混合增量式测试方式：

（1）衍变的自顶向下增量式测试：基本思想是强化对输入/输出模块和引入新算法模块的测试，并自底向上集成为功能相对完整且相对独立的子系统，然后由主模块开始自顶向下进行增量式测试。

（2）自底向上-自顶向下的增量式测试：首先对含读操作的子系统自底向上直至根节点模块，进行集成和测试；然后对含写操作的子系统做自顶向下的集成与测试。

4、各种集成测试方法的比较

1）非增量式测试与增量式测试的比较

- 非增量式测试的方法是先分散测试，然后集中起来再一次完成集成测试。假如在模块的接口处存在错误，只会在最后的集成测试时一下子暴露出来。
- 增量式测试是逐步集成和逐步测试的方法，把可能出现的差错分散暴露出来，便于找出问题和修改。而且一些模块在逐步集成的测试中，得到了较多次的考验，因此，可能会取得较好的测试效果。

结论：增量式测试要比非增量式测试具有一定的优势。

集成测试方法的比较（续）

2）自顶向下与自底向上增量式测试的比较

- 自顶向下增量式测试：

  ——优点：在于它可以自然的做到逐步求精，一开始就能让测试者看到系统的框架。

  ——缺点：需要提供桩模块，并且在输入/输出模块接入系统以前，在桩模块中表示测试数据有一定困难。

- 自底向上增量式测试：

  ——优点：由于驱动模块模拟了所有调用参数，即使数据流并未构成有向的非环状图，生成测试数据也无困难。

  ——缺点：直到最后一个模块被加进去之后才能看到整个程序（系统）的框架。

5、回归测试

- 回归测试：

  ——在集成测试策略的环境中，回归测试是对某些已经进行过的测试的某些子集再重新进行一遍，以保证上述改变不会传播产生无法预料的副作用或引发新的问题。

  ——在更广的环境里，回归测试就是用来保证（由于测试或其它原因的）改动不会带来不可预料的行为或另外的错误。

- 回归测试可以通过人工重新执行全部测试用例的一个子集来进行，也可以使用自动化的捕获回放工具来进行。
- 回归测试集一般应包括三种不同类型的测试用例：

（1）能够测试软件的所有功能的代表性测试用例

（2）专门针对可能会被修改而影响软件功能的附加测试

（3）针对修改过的软件成分的测试

2.6 确认、系统与验收测试

- 2.6.1 确认测试
- 2.6.2 系统测试
- 2.6.3 验收测试

2.6.1 确认测试Validation Test

1、确认测试

- 按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是确认测试的任务。
- 确认测试也称为合格性测试，是验证软件的功能和性能及其它特性是否与用户需求一致。
- 确认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。

2.6.1 确认测试Validation Test

- 经过确认测试，可以为已开发的软件给出结论性评价：

（1）经过检验的软件的功能、性能及其它要求均已满足需求规格说明书的规定，则可被认为是合格的软件。

（2）经过检验发现与需求说明书有相当的偏离，得到一个缺陷清单。

2.6.1确认测试（续）

2、配置审查的内容

- 确认测试过程的重要环节就是配置审查工作。其目的在于确保已开发软件的所有文件资料均已编写齐全，并得到分类编目，足以支持运行以后的软件维护工作。
- 配置审查的文件资料包括用户所需的以下资料：

（1）用户手册

（2）操作手册

（3）设计文档

——如：设计说明书、源程序以及测试资料（测试说明书、测试报告）等。

2.6.2 系统测试System Testing

- 为什么要进行系统测试？

  ——由于软件只是计算机系统中的一个组成部分，软件开发完成之后，最终还要和系统中的硬件系统、某些支持软件、数据信息等配套运行。

- 系统测试是将已经确认的软件、计算机硬件、外设、网络等其它元素结合在一起，进行信息系统的各种组装测试和确认测试，其目的是通过与系统的需求相比较，发现所开发的系统与用户需求不符或矛盾的地方，从而提出更加完善的方案。

2.6.2 系统测试System Testing

- 系统测试应该由若干个不同测试组成，目的是充分运行系统，验证系统各部件是否都能正常工作并完成所赋予任务。这里所谓的系统不仅仅包括软件本身，而且还包括计算机硬件及其相关的外围设备、实际运行时大批量数据、非正常操作（如黑客攻击）等。
- 系统测试的对象：产品系统的软件，软件运行所依赖的硬件、外设，以及相关数据、支持软件及接口等。因此，必须将系统中的软件与各种依赖的资源结合起来，在系统实际运行环境下来进行测试。

2.6.2 系统测试System Testing

- 系统测试通常包括：

1. 功能测试
2. 压力测试
3. 性能测试
4. 安全测试
5. 容错测试等。

1. 功能（正确性）测试

- 功能（正确性）测试检查软件的功能是否符合规格说明。
- 功能（正确性）测试的方法：

  ——枚举法，即构造一些合理输入，检查是否得到期望的输出。测试时应尽量设法减少枚举的次数，关键在于寻找等价区间，因为在等价区间中，只需用任意值测试一次即可。

  ——边界值测试，即采用定义域或者等价区间的边界值进行测试。因为程序设计容易疏忽边界情况，程序也容易在边界值处出错。

2、压力测试stress test

- 压力测试：也称强度测试、负载测试。压力测试是模拟实际应用的软硬件环境及用户使用过程的系统负荷，**长时间或超大负荷**地运行测试软件，来测试被测系统的性能、可靠性、稳定性等。
- 压力测试的目的就是在软件投入使用以前或软件负载达到极限以前，通过执行可重复的负载测试，了解系统可靠性、性能瓶颈等，以提高软件系统的可靠性、稳定性，减少系统的宕机时间和因此带来的损失。
- 压力测试需要在反常规数据量、频率或资源的方式下运行系统，以检验系统能力的最高实际限度。

2、压力测试stress test

- 举例：

  — 如果正常的中断频率为每秒5次，压力测试设计为每秒50次中断。

  — 把输入数据的量提高一个数量级来测试输入功能会如何响应。

  — 若某系统正常运行可支持200个终端并行工作，压力测试则检验1000个终端并行工作的情况。

  — 运行大量的消耗内存或其它系统资源的测试实例。

3、性能测试performance test

- 性能测试：通过测试确定系统运行时的性能表现，如得到运行速度、响应时间、占有系统资源等方面的系统数据。
- 性能测试用来测试软件在系统集成中的运行性能，特别是针对实时系统和嵌入式系统，仅提供符合功能需求但不符合性能需求的软件是不能被接受的。
  - 如某个网站可以提供预先设定的功能，但每打开一个页面都需要1～2分钟，用户不可忍受，其结果没有用户愿意使用这个网站所提供的服务。

3、性能测试performance test

- 性能测试可以在测试过程的任意阶段进行，但只有当整个系统的所有成分都集成在一起后，才能检查一个系统的真正性能。
- 性能测试常常和压力测试结合起来进行，而且常常需要硬件和软件测试设备，也就是说，常常有必要在一种苛刻的环境中衡量资源的使用。

4、安全测试security test （1）

- 安全测试：检查系统对非法侵入的防范能力，验证安装在系统内的保护机制能否在实际中保护系统且不受非法入侵，不受各种非法干扰。
- 在安全测试中，测试者扮演着试图攻击系统的个人角色，采用各种办法试图突破防线。系统安全设计的准则是，使非法侵入的代价超过被保护信息的价值。

4、安全测试security test(续)

- 安全测试常用方法有：
  - 尝试去通过外部的手段来获取系统的密码
  - 使用可以瓦解任何防守的客户软件来攻击系统
  - 使系统“瘫痪”，使得其他用户无法访问
  - 有目的地引发系统错误，期望在恢复过程中侵入系统
  - 通过浏览非保密的数据，从中找到进入系统的钥匙
- 系统的安全测试要设置一些测试用例试图突破系统的安全保密措施，检验系统是否有安全保密的漏洞。

5、可靠性测试

- 可靠性测试是从验证的角度出发，检验系统的可靠性是否达到预期的目标，同时给出当前系统可能的可靠性增长情况。
- 对可靠性测试来说，最关键的测试数据包括失效间隔时间，失效修复时间，失效数量，失效级别等。根据获得的测试数据，应用可靠性模型，可以得到系统的失效率及可靠性增长趋势。
- 可靠性指标有时很难测试，通常采用平均无故障时间或系统投入运行后出现的故障不能大于多少数量这些指标来对可靠性进行评估。

6、容错测试（recovery test）

- 容错测试：主要检查系统的容错能力。当系统出错时，能否在指定时间间隔内修正错误并重新启动系统。
- 容错测试首先要通过各种手段，让软件强制性地发生故障，然后验证系统是否能尽快恢复。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的正确性；对于人工干预的恢复系统，还需估测平均修复时间，确定是否在可接受的范围内。

7、兼容性测试

- 随着软件逐步被推向市场，更多的用户安装，兼容性问题也日益凸现出来了。理论上任何两个软件之间都有冲突的可能，因此软件的兼容性就成为了衡量软件好坏的一个重要指标。
- 兼容性测试是指测试软件在特定的硬件平台上、不同的应用软件之间、不同的操作系统平台上、不同的网络等环境中是否能很好地运行的测试。简单的说，兼容性测试是指测试某新开发的软件在某一特定环境下与各种软件的协调性，软件之间能否很好地协同工作。
- 例如，会不会有相互不良的影响？还有软件和硬件之间能否发挥很好的高效率工作？会不会影响或导致系统的崩溃？等等。

7、兼容性测试（续）

- 兼容性测试的核心内容：
- 一是测试软件是否能在不同的操作系统平台上兼容，或测试软件是否能在同一操作系统平台的不同版本上兼容；
- 二是软件本身能否向前或者向后兼容；
- 三是测试软件能否与其它相关的软件兼容；
- 四是数据兼容性测试，主要是指数据能否共享等。

7、兼容性测试（续）

- 归结起来，软件兼容性通常可分为有4种：

——向前兼容与向后兼容、不同版本间的兼容、标准和规范兼容、数据共享兼容.

（1）向前兼容和向后兼容

向前兼容是指可以使用软件的未来版本，向后兼容是指可以使用软件的以前版本。并非所有的软件都要求向前兼容和向后兼容，这是软件设计者需要决定的产品特性。

例：使用文本文件可以对向前兼容和向后兼容作一个简单的演示：在Windows 98上用Notepad创建的文本文件，它可以向后兼容MS-DOS 1.0后的所有版本，它还可以向前兼容Windows 2000甚至以后的版本。

兼容性测试（续）

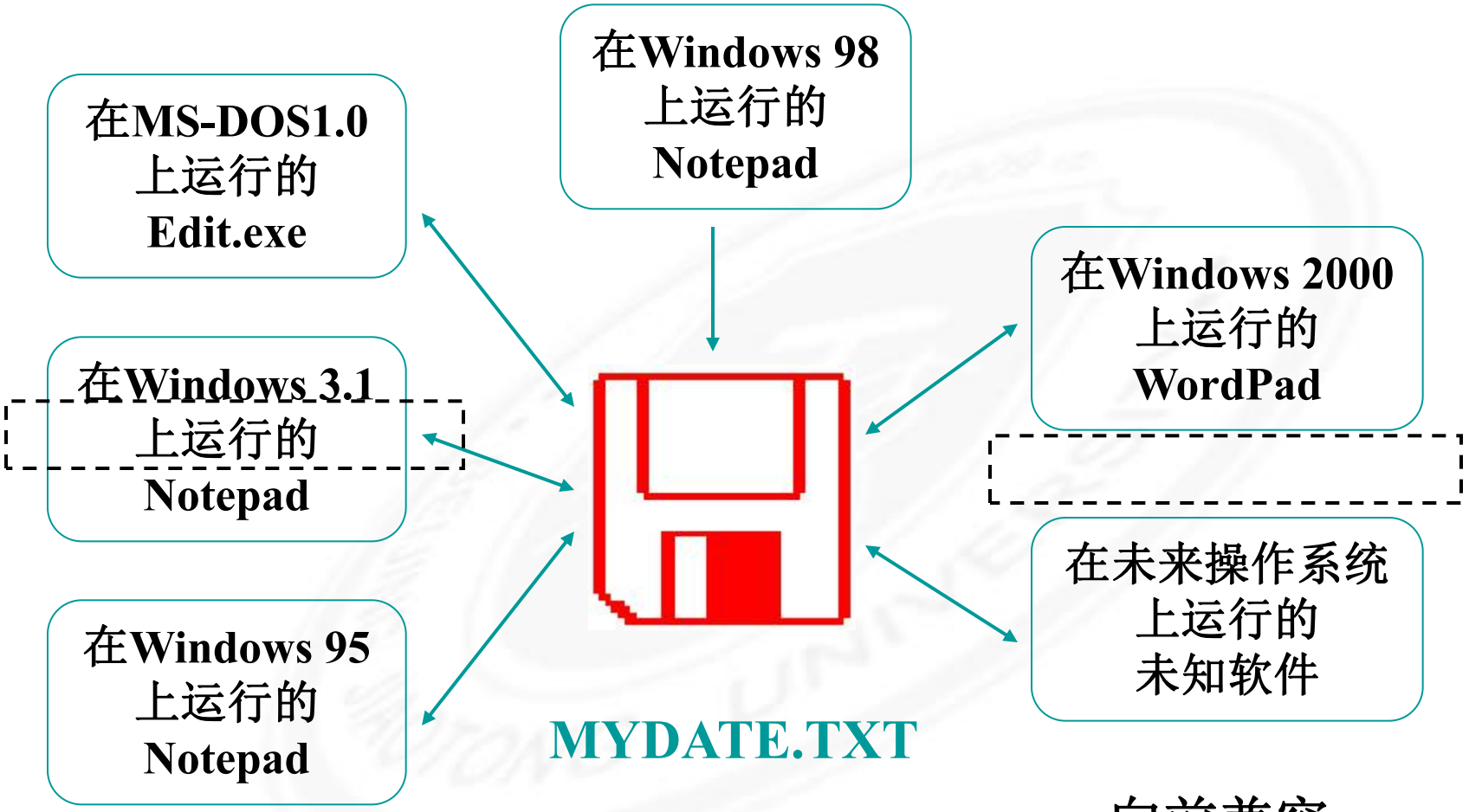

在MS-DOS1.0
上运行的
Edit.exe
在Windows 98
上运行的
Notepad
在Windows 2000
上运行的
WordPad
在Windows 3.1
上运行的
Notepad
在未来操作系统
上运行的
未知软件
在Windows 95
上运行的
Notepad
MYDATE.TXT
向后兼容
向前兼容

图2-15

7、兼容性测试（续）

（2）不同版本间的兼容

不同版本间的兼容是指要实现测试平台和应用软件多个版本之间能够正常工作。

- 举例：

现在要测试一个流行的操作系统的新版本，当前操作系统上可能有数十上百万现有程序，则新操作系统的目标是与它们百分之百兼容。因为不可能在一个操作系统上测试所有的软件程序，因此需要决定哪些是最重要的、必须进行的。

对于测试新应用软件也一样，需要决定在哪个平台版本上测试，以及和什么应用程序一起测试。

兼容性测试（续）

（3）标准和规范兼容

适用于软件平台的标准和规范有两个级别——高级标准和低级标准。

- 高级标准是产品应当普遍遵守的，例如软件能在何种操作系统上运行？是互联网上的程序吗？它运行于何种浏览器？每一项问题都关系到平台，假若应用程序声明与某个平台兼容，就必须遵守关于该平台的标准和规范。
- 低级标准是对产品开发细节的描述，从某种意义上说，低级标准比高级标准更加重要。

兼容性测试（续）

（4）数据共享兼容

数据共享兼容是指要在应用程序之间共享数据，它要求支持并遵守公开的标准，允许用户与其她软件无障碍的传输数据。

- 实例：

— 在Windows环境下，程序间通过剪切、复制和粘贴实现数据共享。在此状况下，传输通过剪贴板的程序来实现。若对某个程序进行兼容性测试就要确认其数据能够利用剪切板与其它程序之间进行相互复制。

— 通过读写移动外存实现数据共享，如软磁盘、U盘、移动硬盘等，但文件的数据格式必须符合标准，才能在多台计算机上保持兼容。

例：一个Web网站的系统测试

- Web网站的网页是由文字、图形、音频、视频和超级链接组成的文档。
- 对网站的测试包含许多方面，如配置测试、兼容测试、可用性测试、文档测试等；黑盒测试、白盒测试、静态测试和动态测试技术都有可能采用。
- 通常Web网站测试包含以下内容：

（1）文字测试　　（2）链接测试

（3）图像、视频测试　　（4）表单测试

（5）动态内容测试　　（6）数据库测试

（7）服务器性能及负载测试　　（8）安全性测试

2.6.3 验收测试acceptance testing

- 验收测试是部署软件前的最后一个测试操作，目的是确保软件准备就绪，并可以让最终用户将其用于执行软件的既定功能和任务。它是向未来的用户表明系统能够像预定要求那样工作。
- **验收测试**一般是由用户方或者用户方认可的评测机构进行，按照软件采购或者研制委托合同进行测试，以确定软件是否符合其验收准则，用户或客户确定是否接受系统。验收测试也可由开发方进行，目的是使客户确认是否接受此系统。验收测试是客户对软件质量评价的一个重要标准.
- 经系统测试后，接着就应该进行验收测试，进一步验证软件的有效性，即软件的功能和性能是否如同用户所合理期待的那样。它的测试数据通常是系统测试的测试数据的子集。

1、验收测试的内容

- 验收测试应完成的工作包括：

（1）明确验收项目，规定验收测试通过的标准。

（2）确定测试方法。

（3）决定验收测试的组织机构和可利用的资源。

（4）选定测试结果分析方法。

（5）指定验收测试计划并进行评审。

（6）设计验收测试所用的测试用例。

（7）审查验收测试准备工作。

（8）执行验收测试。

（9）分析测试结果。

（10）做出验收结论，明确通过验收或不通过验收。

1、验收测试的内容(续)

验收测试可能包括的内容：

- 功能测试。如，完整的工资计算过程。
- 逆向测试。如，检验不符合要求数据而引起出错的恢复能力。
- 特殊情况。如，极限测试、不存在的路径测试。
- 压力测试。如，大批量的数据或者最大用户并发使用。
- 容错测试。如，硬件故障或用户不良数据引起的一些情况。
- 可维护性的评价。
- 用户操作测试。如启动、退出系统等。
- 用户友好性检验。
- 安全测试。
- 文档测试等。

2. 软件配置和文档资料测试

- 对文档的测试包括以下内容：

（1）检查产品说明书属性

（2）检查是否完整

（3）检查是否准确

（4）检查是否精确

（5）检查是否一致

（6）检查是否贴切

（7）检查是否合理

（8）检查代码无关

（9）检查可测试性

本章重点内容

- 软件测试的复杂性
- 软件测试的方法与策略
- 单元测试的主要任务和过程
- 集成测试的方法和确认测试的准则
- 系统测试的测试类型和测试要点
- 验收测试的主要内容和相关配置

本章完