软件测试课件整理

目录

[part0-课程概述 3](#_Toc226299878)

[1、 什么是计算机软件？它由哪几部分组成？ 3](#_Toc226299879)

[定义 3](#_Toc226299880)

[由两部分组成 3](#_Toc226299881)

[2、 软件失效的机理是什么？什么是软件的错误、缺陷、故障和失效？ 3](#_Toc226299882)

[3、 GB/T12504-90中软件质量的定义是什么？ 3](#_Toc226299883)

[4、 IEEE1983中软件测试的定义是什么？ 4](#_Toc226299884)

[5、 软件测试与软件质量保证之间的关系是什么？ 4](#_Toc226299885)

[part1-软件测试概述 4](#_Toc226299886)

[1、 软件测试由哪几个阶段组成？ 4](#_Toc226299887)

[软件测试的四个阶段 4](#_Toc226299888)

[2、 软件测试主要包括哪些测试技术？ 4](#_Toc226299889)

[静态测试技术 4](#_Toc226299890)

[动态测试技术 5](#_Toc226299891)

[测试用例 6](#_Toc226299892)

[3、 软件测试的充分性的含义是什么？ 6](#_Toc226299893)

[软件测试的充分性概念的提出 6](#_Toc226299894)

[含义 6](#_Toc226299895)

[4、 良好的软件测试充分性准则应具有那些基本性质？ 6](#_Toc226299896)

[良好的软件测试充分性准则应具有的基本性质 6](#_Toc226299897)

[5、 测试覆盖准则具有什么作用？ 7](#_Toc226299898)

[测试覆盖准则与测试充分性 7](#_Toc226299899)

[测试覆盖准则的作用 7](#_Toc226299900)

[part3-功能性测试 7](#_Toc226299901)

[边界值分析 7](#_Toc226299902)

[局限性 7](#_Toc226299903)

[优点 7](#_Toc226299904)

[part04-静态测试 7](#_Toc226299905)

[主要由人工进行的静态测试方法有哪几种？ 8](#_Toc226299906)

[代码审查与代码走查有什么区别？ 8](#_Toc226299907)

[代码审查 8](#_Toc226299908)

[阅读的方法 10](#_Toc226299909)

[代码走查 11](#_Toc226299910)

[part05-结构性测试 12](#_Toc226299911)

[各种覆盖准则之间的包含关系 12](#_Toc226299912)

[程序切片 13](#_Toc226299913)

[静态切片的计算方法 13](#_Toc226299914)

[用于方法评估的指标（解决） 14](#_Toc226299915)

[part06-集成测试 14](#_Toc226299916)

[什么是集成测试 14](#_Toc226299917)

[集成测试的必要性 14](#_Toc226299918)

[集成测试的方法 15](#_Toc226299919)

[基于分解的集成 15](#_Toc226299920)

[基于调用图的集成 16](#_Toc226299921)

[基于路径的集成 17](#_Toc226299922)

# part0-课程概述

## 什么是计算机软件？它由哪几部分组成？

### 定义

计算机软件是与计算机系统操作有关的程序、规程、规则及任何与之有关的文档及数据。

### 由两部分组成

#### 可执行部分

程序：按事先设计的功能和性能需求执行的指令序列。

数据：是程序能正常操纵信息的数据结构。

#### 不可执行部分

文档：与软件开发、运行、维护、使用和培训有关的图文材料。

## 软件失效的机理是什么？什么是软件的错误、缺陷、故障和失效？

软件失效机理可以描述为：软件错误→软件缺陷→软件故障→软件失效

* 软件错误（**software error**）：是指在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生。
* 软件缺陷（**software defect**）：是存在于软件（文档、数据、程序）之中的那些不希望或不可接受的偏差，其结果是软件运行于某一特定条件时出现软件故障，此时称软件缺陷被激活。当软件意指程序时，软件缺陷（**defect**）与软件**/**程序污点（**bug**）同义。软件缺陷是存在于软件内部的、静态的一种形式。
* 软件故障**（software fault）**：是指软件运行过程中出现的一种不希望或不可接受的内部状态。出现软件故障时若无适当措施（容错）加以及时处理便产生软件失效软件故障是一种动态行为
* 软件失效（**software failure**）：是指软件运行时产生的一种不希望或不可接受的外部行为结果。

## GB/T12504-90中软件质量的定义是什么？

* 软件产品中能满足给定需求的各种特性的总和
* 这些特性称为质量特性，它包括功能度、可靠性、易使用性、时间经济性、资源经济性、可维护性和可移植性等

## IEEE1983中软件测试的定义是什么？

* 使用人工或自动手段来运行或测定某个系统的过程
* 其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别

## 软件测试与软件质量保证之间的关系是什么？

* Program testing can be used to show the presence of bugs, but never their absence.

- Dijkstra, 1972

* 软件质量不是在软件产品中被测试出来的，而是在软件开发和生产过程中形成的。
* 软件测试不是万能的，它不能取代其它软件质量保证手段。

软件质量需要综合运用

# part1-软件测试概述

## 软件测试由哪几个阶段组成？

### 软件测试的四个阶段

* 单元测试是对软件设计的最小单位：模块的测试，目的是检验每个软件单元能否正确地实现其功能满足其性能和接口要求。
* 单元测试验证程序与详细设计说明的一致性。
* 组装测试将经过单元测试的模块逐步进行组装和测试，组装测试亦称集成测试。
* 组装测试验证程序和概要设计说明的一致性。
* 确认测试是检验所开发的软件是否按软件需求规格说明中确定的软件功能、性能、约束及限制等技术要求进行工作。
* 系统测试指软件与该软件所属的系统对接并测试其接口的过程，目的是在真实工作环境下检验软件是否能与系统正确连接，并满足软件研制的系统目标。

## 软件测试主要包括哪些测试技术？

### 静态测试技术

定义：不执行程序代码而寻找程序代码中可能存在的缺陷或评估程序代码的过程

* 主要由人工进行。
  + 代码审查。
  + 代码走查。
  + 桌面检查。
* 主要由软件工具自动进行的静态分析。

### 动态测试技术

定义：通过在抽样测试数据上运行程序来检验程序的动态行为和运行结果以发现缺陷

* 核心内容
  + 生成测试用例
  + 运行程序
  + 验证程序的运行结果
* 辅助工作
  + 文档编制
  + 数据管理
  + 操作规程
  + 工具应用
  + **…**

#### 动态测试技术分类

* 黑盒测试（功能测试、基于规约的测试）
  + 测试人员无须了解程序的内部结构，直接根据程序输入和输出之间的关系或程序的需求规范确定测试数据，推断测试结果的正确性
  + 边界值分析
  + 等价类划分
  + 基于决策表的测试
  + 因果图
  + 正交实验设计
  + 状态测试
  + …
* 白盒测试（结构测试、基于程序的测试）
  + 测试人员根据程序的内部结构特性和与程序路径相关的数据特性设计测试数据。
  + 控制流测试
    - 语句覆盖
    - 分支覆盖
    - 条件覆盖
    - 判定-条件覆盖
    - 路径覆盖
    - …
  + 数据流测试
    - 定义覆盖
    - 引用覆盖
    - 定义-引用覆盖
    - …
  + 变异测试
  + …
* 灰盒测试（程序与规约相结合的测试）
  + 黑盒测试与白盒测试相结合的方法
  + 先用黑盒测试方法设计测试用例，然后尽可能多地用白盒测试方法完成测试

### 测试用例

* 动态测试最重要的问题是生成测试用例的策略，它是动态测试有效、高效的关键。
* 测试用例包括输入数据和期望结果

## 软件测试的充分性的含义是什么？

### 软件测试的充分性概念的提出

Goodenough 和Gerhart 于1975 年在研究软件测试能否保证软件的正确性时提出

### 含义

软件测试的充分性是根据被测软件在有限多个测试数据上的行为判断在所有测试数据上的行为的逻辑基础

## 良好的软件测试充分性准则应具有那些基本性质？

### 良好的软件测试充分性准则应具有的基本性质

#### 空集不充分性：

空测试数据集对任何软件都是不充分的

#### 有限性：

对任何软件都存在有限的充分测试数据集合

#### 单调性：

如果一个软件在一个测试数据集上的测试是充分的，那么再增加一些测试数据也应该是充分的

#### 非复合性：

即使一个测试数据集对于软件所有的成分是充分的，也不能保证它对于整个软件是充分的

#### 非分解性：

即使一个测试数据集对于整个软件是充分的，也不能保证它对于软件所有的成分是充分的

#### 非外延性：

即使两个程序语义上是等价的，也不能保证对其中一个程序充分的测试数据集对另一个程序也是充分的

#### 一般多重修改性：

即使两个程序语法结构相同，也不能保证对其中一个程序充分的测试数据集对另一个程序也是充分的

#### 复杂性：

软件越复杂，需要的测试数据就越多

#### 回报递减律：

测试得越多，进一步测试所得到的充分性增长就越少

## 测试覆盖准则具有什么作用？

### 测试覆盖准则与测试充分性

* 通常用测试覆盖准则度量测试充分性
* 针对程序内部结构的测试覆盖准则，主要包括控制流测试覆盖准则和数据流测试覆盖准则
* 控制流测试覆盖准则

􀂄 包括语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、路径覆盖

* 数据流测试覆盖准则

􀂄 包括定义覆盖、引用覆盖、定义-引用覆盖等准则

### 测试覆盖准则的作用

* 定量地规定软件测试需求
* 指导测试数据的选择
* 度量测试数据集揭示软件特定特征的能力
* 对测试结果和软件可靠性评估具有重要影响

# part3-功能性测试

## 边界值分析

### 局限性

* 假定N个变量是相互独立的，没有考虑这些变量之间的相互依赖关系
* 不考虑变量的语义

### 优点

边界值分析测试用例通过引用物理量的边界独立变量极值导出，不考虑函数的性质，也不考虑变量的语义，简便易行，生成测试数据的成本很低

# part04-静态测试

## 主要由人工进行的静态测试方法有哪几种？

* **代码审查（Code Inspection）**
* **代码走查（Walkthrough）**
* **桌面检查（Peer-to-Peer）**

## 代码审查与代码走查有什么区别？

### 代码审查

* 代码审查的测试内容
  + 检查代码和设计的一致性
  + 检查代码对标准的遵循、可读性
  + 检查代码的逻辑表达的正确性
  + 检查代码结构的合理性
* 代码审查的组成和方式
  + 由一组程序和错误检查技术组成
  + 以代码审查组方式进行
* 代码审查组
  + 通常由四人组成，其中一人为组长
  + 组长是关键，最好是一个称职的程序员，但不是被测试程序的编写者，也不需要对所检查的程序很熟悉，但需要较强的组织协调和语言能力
  + 组长的职责包括分配资料、安排计划、主持开会、记录并保存被发现的错误
  + 其余成员包括资深程序员、程序编写者与专职测试人员
  + 根据测试的组织方式（如内部测试和独立测试）不同，代码审查小组组成可以调节，但组长角色不能变动

#### 代码审查的步骤

* + 四个步骤
    - 准备
    - 程序阅读
    - 审查会
    - 跟踪及报告

##### 准备

* + 组长提前把程序目录表和设计说明书等材料分配给小组成员
  + 小组成员熟悉这些材料
  + 由被测程序的设计和编码人员向审查组详细说明所准备的材料，特别是代码的主要功能与功能间的关系

##### 程序阅读

* + 审查组人员仔细阅读代码和相关材料
  + 对照代码审查单标出明显缺陷及错误

##### 审查会

* + 审查会由组长主持
  + 首先由程序员逐句阐明程序的逻辑，在此过程中可由程序员或其他小组成员提出问题，追踪错误是否存在
  + 经验证明在上述阐述过程中，有很多错误由讲述程序者而不是其他小组成员发现
  + 大声地朗读程序给听众，这样简单的工作是有效的错误检测技术
  + 然后利用代码审查单来分析讨论
  + 组长负责讨论沿着建设性的方向前进，而其他人则集中注意力发现错误，但不去纠正错误

##### 跟踪及报告

* + 会后把发现的错误登记造表并交给程序开发人员
  + 如果发现错误较多或发现重大错误，那么在改正之后，组长要再次组织审查会议
  + 为了改进以后的审查工作，对错误登记表也要分析，归类和精炼
* 审查会的时间
  + 大约以1.5-2小时为宜
  + 审查会需要高度集中注意力，时间太长反而容易使效率降低
  + 每次会议可能处理一个或几个模块
* 代码审查单
  + 代码审查过程所用的主要技术
  + 通常是把程序设计及编码中可能发生的各种错误进行分类，对每一类列举出尽可能多的典型错误，然后制成表格
  + 其它测试中发现的错误也要及时归入代码审查单，形成某一类型软件针对性的代码审查单，以供审查时使用

### 阅读的方法

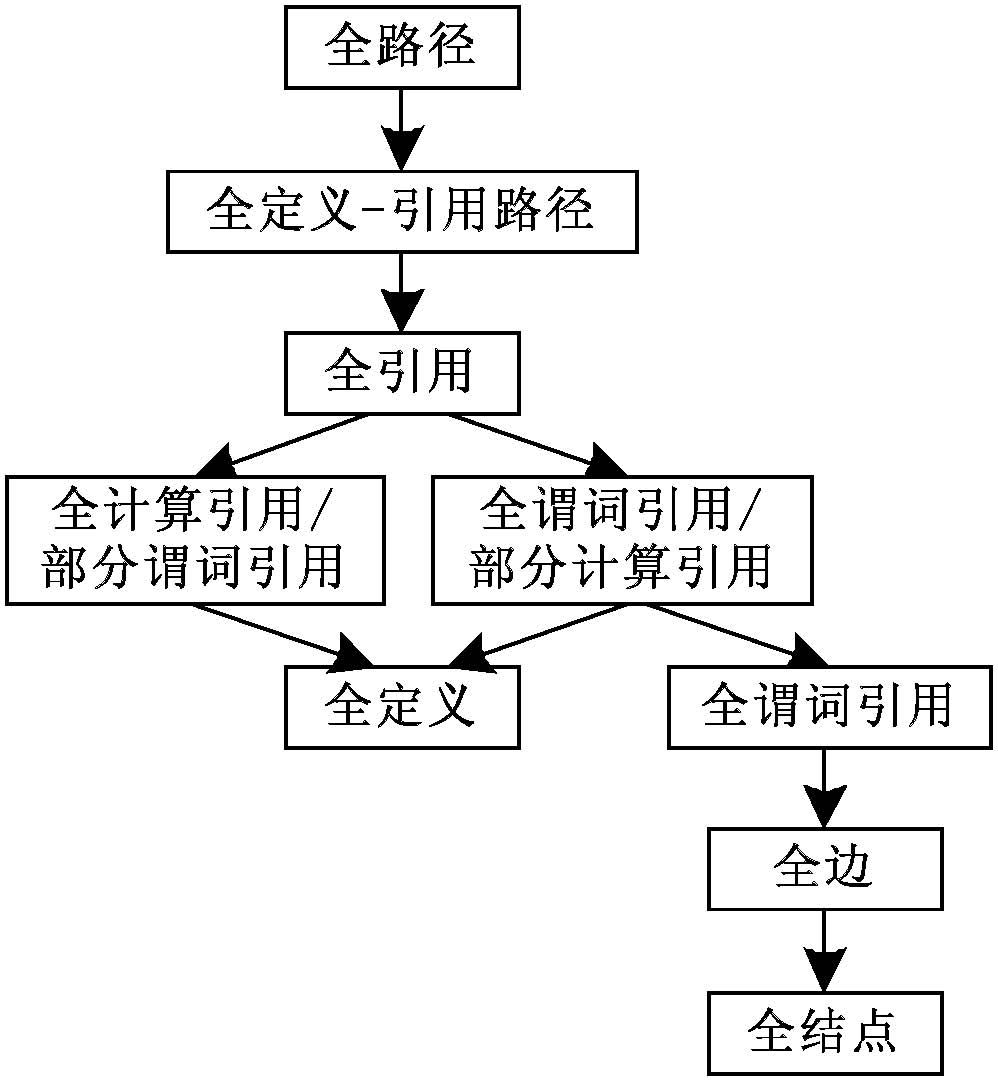
* + 要仔细阅读需求设计等文档，特别是了解软件的整体物理意义、应用背景以及在大系统中的地位
  + 对大型软件而言，这些信息会在阅读程序时有效地帮助读者从一定的高度审视，而不是停留在逐行扫描代码
  + 有些错误要有整体观才能发现
* 阅读结构化代码的两种方法
  + 追踪通过每个子程序的主要逻辑行，主要逻辑行全部跟踪完，然后开始跟踪第二条路径
    - 相当于深度优先遍历。
  + 按排列顺序追踪代码，从主要行开始，然后检查较低层的程序段
    - 相当于广度优先遍历
* 两种阅读方法的综合使用
  + 这两种方式的差别在于何时进入下层模块，选择那种方式要视具体程序特点定
    - 广度优先遍历有助于很快地了解程序的全貌
    - 深度优先适于详细查阅功能处理步骤
  + 应综合使用上述两种方法，在头一两遍阅读时，采用广度优先，然后用深度优先
* 阅读程序的次数
  + Beizer提出至少要读程序4次，分别针对印刷错误、数据结构、控制流和处理
  + 4次阅读要比读一次能更快、更容易、更可靠的完成任务
  + 多遍阅读程序、分步检查问题是代码审查的工作原则
* 代码审查的辅助工具
  + 汇编或编译器生成的交叉引用表（变量、标号、子程序）
  + 逆向工程工具（例如从源代码生成流程图）
  + 带有快速查找的编辑器

### 代码走查

* 代码走查
  + 代码走查与代码审查相似，它也是由一组程序和错误检查技术组成，只是程序和错误检查技术不完全相同
* 代码走查组
  + 代码走查以小组方式进行
  + 代码走查组包括
    - 组长，类似代码审查组长
    - 秘书，负责记录发现的错误，要有一定水平
    - 测试人员，应是具有经验的程序设计人员，或精通程序设计语言的人员，或从未介入被测试程序的设计工作的技术人员（这样的人没有被已有的设计框住），没有约束，比较容易发现问题
* 代码走查的过程
  + 与代码审查过程相似
  + 先把材料交给每个小组人员，让他们认真研究程序，然后再开会
* 代码走查会的内容
  + 与代码审查不同，不是读程序和使用代码审查单
  + 而是由被指定的作为测试员的小组成员提供若干测试用例（程序的输入数据和期望的输出结果），让参加会的成员当计算机，在会议上对每个测试用例用头脑来执行程序，也就是用测试用例沿程序逻辑走一遍，并由测试人员讲述程序执行过程，在纸上或黑板上监视程序状态（变量的值）
  + 每次开会时间以1-2小时为宜，但不允许中断
  + 如果发现问题由秘书记下来，中间不讨论任何纠错问题，主要是发现错误
* 代码走查中测试用例的作用
  + 代码走查中，测试用例并不是关键，也并不是仅想验证这几个测试用例运行是否正确，人脑毕竟比计算机慢太多
  + 这里测试用例是作为怀疑程序逻辑与计算错误的启发点，在随测试实例游历程序逻辑时，在怀疑程序的过程中发现错误
  + 这比几个测试用例本身直接发现的错误要多得多
* 代码走查的缺点
  + 代码走查使用测试用例启发检测错误，人们注意力会相对集中在随测试用例游历的程序逻辑路径上，不如代码审查检查的范围广，错误覆盖面全

# part05-结构性测试

## 各种覆盖准则之间的包含关系



## 程序切片

* + **抽取程序中影响某一位置变量的语句的一门技术**
* **程序切片的思想**
  + **对程序进行裁剪，使得所得到的程序代码仍能反映原程序的部分特征**
  + **程序切片专门针对这类问题，它按切片准则来裁剪程序，使人们能把注意力集中在相关的程序代码上**
* **程序切片的分类**
  + **静态切片**
    - **与程序的输入无关**
    - **Weiser提出的程序切片属于此类**
  + **动态切片**
    - **与程序的输入相关**
    - **B. Korel**

### 静态切片的计算方法

* + **基本概念**
  + **有向图**
  + **控制流图**
  + **路径**
* **程序依赖图通过合并控制依赖图和数据依赖图而获得**
* **根据程序依赖图计算静态切片**
  + **在程序依赖图中求出所关心结点的逆向传递闭包就可以得出它所数据依赖、控制依赖的结点**
  + **由以上计算所得结点对应的程序语句所组成的程序段即为所求静态切片**

## 用于方法评估的指标（解决）

# part06-集成测试

## 什么是集成测试

* + **集成测试亦称组装测试**
  + **集成测试将经过单元测试的模块逐步进行组装和测试**
  + **集成测试验证程序和概要设计说明的一致性**
  + **集成测试在模块组装后查找模块间接口的错误**

## 集成测试的必要性

* + **良好的软件测试充分性准则应具有的基本性质**
    - **非复合性：即使一个测试数据集对于软件所有的成分是充分的，也不能保证它对于整个软件是充分的**
    - **非分解性：即使一个测试数据集对于整个软件是充分的，也不能保证它对于软件所有的成分是充分的**
  + **一个模块可能对另一个模块产生不利的影响**
  + **可能会发现单元测试中未发现的接口方面的错误**
  + **将子功能合成时不一定产生所期望的主功能**
  + **独立可接受的误差，在组装后可能会超过可接受的误差限度**
  + **在单元测试中无法发现时序问题（实时系统）**
  + **在单元测试中无法发现资源竞争问题**

## 集成测试的方法

* + **基于分解的集成**
  + **基于调用图的集成**
  + **基于路径的集成**

### 基于分解的集成

* **模块组装的策略**
  + **非渐增式**
  + **渐增式**

#### 非渐增式

* **非渐增式集成测试：急于求成**
  + **大爆炸集成、莽撞测试（big bang testing）**
  + **一下子把几十个甚至上百个模块和莽撞地联接在一起**
  + **“big bang”描述了这种联接的声音效果“咣”**
  + **目的：通过少数测试运行检测整个系统来论证系统的稳定性**
* **非渐增式集成测试的特点**
  + **优点**
    - **迅速完成集成测试**
    - **测试用例较少**
  + **缺点**
    - **错误难以定位**
    - **即使通过测试，许多接口错误也会隐藏**
  + **应用**
    - **小的、良构的系统，其模块已接受了充分的测试**
    - **一个已经存在的系统，只是做了少量的修改**
    - **通过复用可信赖的模块构造系统**

#### 渐增式集成测试

* + **功能分解树**
  + **自顶向下**
    - **从树根结点开始向下**
  + **自底向上**
    - **从树叶结点开始向上**
  + **三明治**
    - **自顶向下与自底向上相结合**

### 基于调用图的集成

* **基于调用图的集成**
  + **基于分解集成的缺点之一，是以功能分解树为基础**
  + **如果改用调用图，则可以减缓这种缺陷，并且也向结构性测试方向发展**
  + **由于调用图是一种有向图，可以利用前面介绍过的图论知识，像使用程序图那样地使用调用图**
  + **基于调用图的集成测试方法**
    - **成对集成**
    - **相邻集成**

#### 基于调用图的集成测试的优点

* + **免除了驱动器/桩的开发工作**
  + **接口关系测试充分**
  + **测试集中于衔接的功能性**
  + **测试和集成可以并行开始**

#### 基于调用图的集成测试的缺点

* + **缺点**
    - **调用或协作的关系可能是错综复杂的**
    - **参与者没有被单独测试，要充分测试底层模块较困难**
    - **特定的调用或协作可能是不完全的**
    - **缺陷隔离**
  + **适用范围**
    - **尽快论证一个可运行的调用或协作**
    - **被测系统已清楚定义了模块的调用和协作关系**

### 基于路径的集成