SOFTWARE TESTING

Chapter 6
Unit and Integration Testing

School of Data & Computer Science
Sun Yat-sen University

Approaches & Technologies

OUTLINE

- 6.1 测试用例设计
- 6.2 单元测试
  - 单元测试概述
  - 单元测试的的技术要求 (GB/T 15532-2008)
  - 单元测试内容
  - 单元测试的基本流程
  - 单元测试用例设计
  - 单元测试报告
- 6.3 集成测试

概述

- 软件单元 (Software Unit)
  - Computer Software Unit is the lowest group of software code created to perform a specified function or functions, an element of a Computer Software Component (CSC) that is separately testable.
  - Software Unit is an element in the design of a software item; for example, a major subdivision of a software item, a component of that subdivision, a class, object, module, function, routine, or database. Software units may occur at different levels of a hierarchy and may consist of other software units. Software units in the design may or may not have a one-to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities. (ISO 12207-2017)

概述

- 软件单元
  - 软件单元是软件设计说明中一个可独立测试的元素，是程序中一个逻辑独立的 (最小) 部分。
- 单元测试 (组件测试/模块测试)
  - 单元测试由一组独立的测试构成，每个测试针对软件中的一个独立的软件单元。单元测试由开发工程师或开发团队内部测试工程师确定测试计划，编写测试代码，用于检验被测代码所实现的一个独立功能的正确性，是代码级别的测试。
    - 例：判断在特定条件或场景下某函数的行为是否合规。
  - 单元测试是软件生命周期中初始级别的测试。
    - 单元测试是编码完成后，首先要实施的测试。
  - 单元测试方法：静态测试、白盒测试、黑盒测试 (需要提供环境条件)；人工方法、自动工具方法。

概述

- 单元测试的重要性
  - 单元测试是软件测试阶段的初始测试，其测试效果将直接影响软件的后期测试 (集成测试和系统测试)，在很大程度上影响到产品的最终质量。
  - 单元测试是代码级别的测试，测试过程中能够发现集成测试和系统测试很难发现的问题。
    - 单元测试中容易发现的问题如果到后期测试才被发现，其纠正成本将成倍数上升。
  - 单元测试不仅仅用于证明被测代码实现了规定的功能，重要的是明确被测代码如何实现这些功能，是否做了它该做的事情而没有做它不该做的事情。

概述

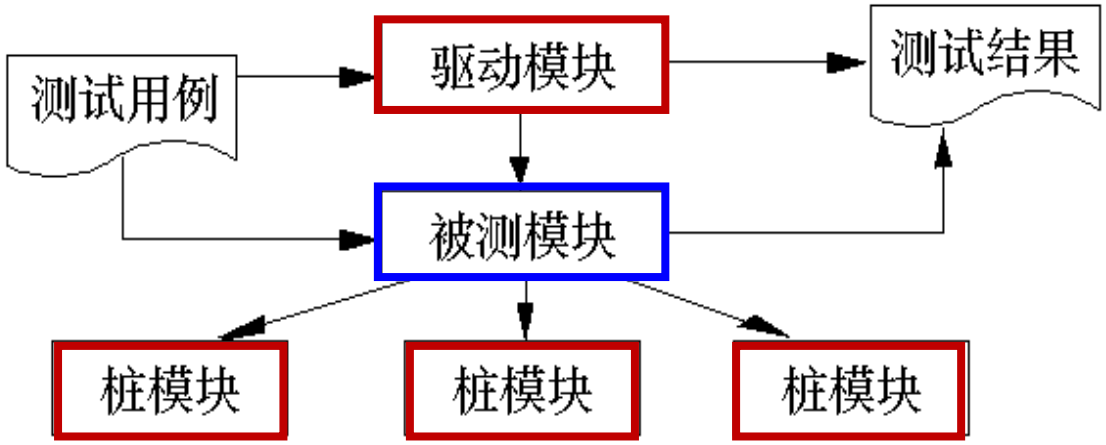
- 单元测试的特点
  - 验证行为
    - 单元测试验证程序中每一项功能的正确性。
  - 设计行为
    - 编写单元测试的要求使得开发者能够多从调用者的角度对代码进行观察。
      - 设计易于调用和测试的程序模块，降低其耦合度。
    - 开发者在编码时进行单元代码测试可以尽量降低程序的缺陷数目。
  - 测试文档
    - 单元测试文档是展示函数或类如何使用的最佳文档。
  - 回归性
    - 自动化的单元测试设计避免了代码级别的回归。软件单元代码编写完成后，可以随时进行快速测试。

单元测试的的技术要求

- 软件单元测试一般应符合以下技术要求： (GB/T 15532-2008)
  - (1) 对软件设计文档规定的软件单元的功能、 性能、 接口等应逐项进行测试；
  - (2) 每个软件特性应至少被一个正常测试用例和一个被认可的异常测试用例覆盖；
  - (3) 测试用例的输人应至少包括有效等价类值、无效等价类值和边界数据值；
  - (4) 在对软件单元进行动态测试之前，一般应对软件单元的源代码进行静态测试；
  - (5) 语句覆盖率达到 100%；
  - (6) 分支覆盖率要达到 100%；
  - (7) 对输出数据及其格式进行测试。
- 对具体的软件单元，可根据软件测试合同 (或项目计划) 以及软件单元的重要性、完整性级别等要求对上述内容进行裁剪。

单元测试内容

- 软件单元通常不是可运行的程序。
  - 单元测试需要编写额外的测试驱动程序和桩模块。
    - 驱动模块 Driver：模拟被测模块的主程序。它接收测试数据，把这些数据传送给被测模块，最后再输出实测结果。
    - 桩模块 Stub (测试存根、连接程序)：代替被测模块调用的子模块。

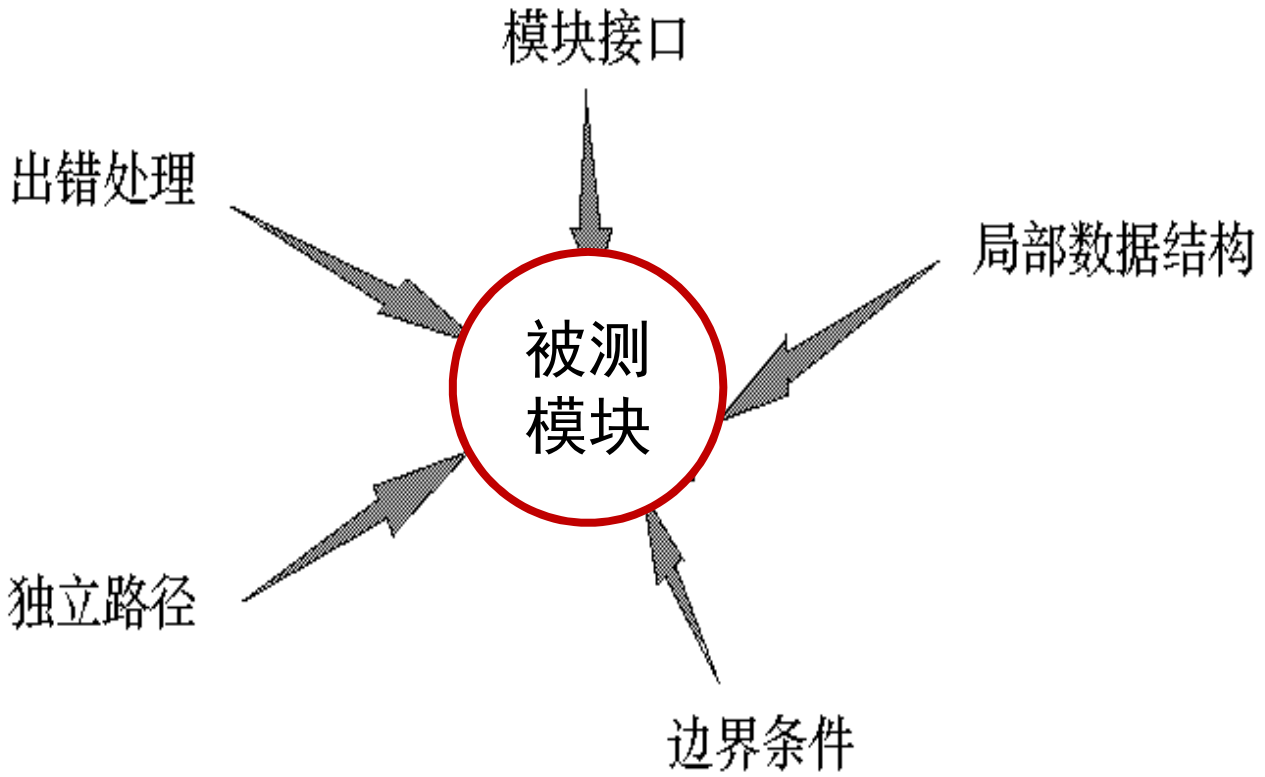


单元测试内容

- 单元测试内容的设计依据是详细设计文档。
  - 根据详细设计说明书和源程序清单，了解被测模块的 I/O 条件和模块的逻辑结构。
  - 静态测试：对所有的局部和全局的数据结构、外部接口和程序代码的关键部分进行桌面检查和代码评审。
  - 动态测试：主要采用白盒测试用例，辅之以黑盒测试用例。
    - 对任何合理和不合理的输入都要能够鉴别和响应。

单元测试内容

单元测试的具体内容



单元测试内容

- 模块接口测试
  - 模块接口测试的主要内容
    - 模块实际输入与定义的输入是否一致。
      - 输入参数的个数、属性、顺序
    - 调用被测模块时的输入参数与模块的形式参数是否匹配。
      - 输入参数的个数、 属性、顺序
    - 是否修改了只做输入用的形式参数。
    - 被测模块中对于非内部/局部变量是否合理使用，全局变量的定义在各模块中是否一致。
    - 使用外部资源时，是否检查了可用性并及时释放资源。
      - 外部资源包括内存、文件、硬盘、端口等

单元测试内容

- 模块接口测试 (续)
  - 在做内外存交换时要考虑：
    - 文件属性是否正确；
    - open 与 close 语句是否正确；
    - 缓冲区容量与记录长度是否匹配；
    - 在进行读写操作之前是否打开了文件；
    - 在结束文件处理时是否关闭了文件；
    - 正文书写/输入错误；
    - I/O 错误是否检查并做了处理。

单元测试内容

- 局部数据结构测试
  - 局部数据结构测试检查局部数据结构能否保持完整性。
    - 不正确或不一致的数据类型说明；
    - 使用尚未赋值或尚未初始化的变量；
    - 错误的初始值或错误的缺省值；
    - 变量名拼写错误或书写错误：使用了外部变量或函数；
    - 不一致的数据类型；
    - 全局数据对模块的影响；
    - 数组越界；
    - 非法指针。

单元测试内容

- 路径测试
  - 检查由于计算错误、判定错误、控制流错误导致的程序错误。
    - 根据白盒测试和黑盒测试用例设计方法设计测试用例，对模块中重要的执行路径进行测试；
    - 设计测试用例查找由于错误的计算、不正确的比较或不正常的控制流而导致的错误；
    - 对基本执行路径和循环进行测试可提高发现路径错误的概率。

- 单元测试内容
  - 路径测试 (续)
    - 检查内容
      - 死代码 dead code；
      - 计算优先级错误；
      - 算法错误；
      - 初始化不正确；
      - 精度错误：
        - 比较运算错误、赋值错误；
      - 表达式的符号不正确：
        - 各类运算符号 (算术、关系、逻辑、位操作、复制、条件、逗号、指针、字节数、其它) 的检查；
      - 循环变量的使用错误：例如错误的赋值；
      - 其他。

单元测试内容

- 路径测试 (续)
  - 常见的比较和控制流错误有：
    - 不同数据类型的比较；
    - 不正确的逻辑运算符或优先次序；
    - 因浮点运算精度问题而造成的浮点数比较不相等；
    - 关系表达式中不正确的变量和关系运算符；
    - “差 1 错”：不正确地多循环或少循环一次；
    - 错误的或不可能的循环终止条件；
    - 当遇到发散的迭代时不能终止循环；
    - 不适当地修改了循环变量等。

单元测试内容

- 错误处理测试
  - 检查内部错误处理设施是否有效。
    - 出错的描述是否难以理解，是否能够对错误定位；
    - 显示的错误提示与实际的错误是否相符；
    - 对错误条件的处理是否正确；
    - 错误被处理之前，错误条件是否已经引起系统的干预。
  - 检查内容：
    - 是否检查了错误出现情况：
      - 资源使用前后、其他模块使用前后；
    - 出现错误后是否进行了错误处理：
      - 引发错误 → 通知用户 → 记录错误
    - 错误处理是否有效：
      - 是否在系统干预前处理错误；
      - 错误报告和记录是否真实详细。

单元测试内容

- 边界测试
  - 检查临界数据是否得到正确处理：
    - 选择测试用例，重点检查数据流、控制流中等于、大于或小于确定的比较值时出错的可能性。
    - 如果对模块性能有要求，还要专门进行关键路径测试。
      - 确定最坏和平均情况下影响模块运行时间的因素。
  - 检查内容：
    - 普通合法数据、普通非法数据是否正确处理；
    - 边界内最接近边界的 (合法) 数据是否正确处理；
    - 边界外最接近边界的 (非法) 数据是否正确处理等；
    - 在 n 次循环的第 0 次、1 次、n 次是否有错误；
    - 运算或判断中取最大最小值时是否有错误；
    - 数据流、控制流中等于、大于、小于确定的比较值时是否出现错误。

- GB/T 15532-2008 单元测试内容
  - 总则
    - 当静态测试时，所测试的内容与选择的测试方法有关。如采用代码审查方法，通常要对寄存器的使用 (仅限定在机器指令和汇编语言时考虑)、程序格式、入口和出口的连接、程序语言的使用、存储器的使用等内容进行检查；采用静态分析方法，通常要对软件单元的控制流、数据流、接口、表达式等内容进行分析。详细内容可参见各种静态测试方法的描述。当动态测试时，通常对软件单元的功能、性能、接口、局部数据结构、独立路径、出错处理、边界条件和内存使用情况进行测试。通常对软件单元接口的测试优先于其他内容的测试。对具体的软件单元，应根据软件测试合同 (或项目计划)、软件设计文档的要求及选择的测试方法确定测试的具体内容。

- GB/T 15532-2008 单元测试内容
  - 接口
    - 测试接口一般应包括以下内容：
      (1) 调用被测单元的实际参数与该单元的形式参数的个数、 属性、量纲、顺序是否一致；
      (2) 被测单元调用子模块时，传递给子模块的实际参数与子模块的形式参数的个数、属性、量纲、顺序是否一致；
      (3) 是否修改了只作为输入值的形式参数；
      (4) 调用内部函数的参数个数、属性、量纲、顺序是否正确；
      (5) 被测单元在使用全局变量时是否与全局变量的定义一致；
      (6) 在单元有多个入口的情况下，是否引用了与当前入口无关的参数；
      (7) 常数是否当作变量来传递；
      (8) 输入/输出文件属性的正确性；

- GB/T 15532-2008 单元测试内容
  - 接口
    - 测试接口一般应包括以下内容：(续)

(9) OPEN 语句的正确性；

(10) CLOSE 语句的正确性；

(11) 规定的输入/输出格式说明与输人/输出语句是否匹配；

(12) 缓冲区容量与记录长度是否匹配；

(13) 文件是否先打开后使用；

(14) 文件结束条件的判断和处理的正确性；

(15) 对输入/输出错误是否进行了检查并做了处理以及处理的正确性。

- GB/T 15532-2008 单元测试内容
  - 局部数据结构
    - 测试软件单元内部的数据能否保持其完整性，包括内部数据内容、格式及相互关系。
    - 应设计测试用例以检查如下差错：
      (1) 不正确或不一致的数据类型说明；
      (2) 错误的变量名，如变量名拼写错或缩写错等；
      (3) 使用尚未赋值或尚未初始化的变量；
      (4) 差错的初始值或差错的缺省值；
      (5) 不一致的数据类型；
      (6) 下溢、上溢或是地址差错；
      (7) 全局数据对软件单元的影响。

- GB/T 15532-2008 单元测试内容
  - 独立路径
    - 独立路径是指在程序中至少引进一个新的处理语句集合或一个新条件的任一路径。在程序的控制流图中，一条独立路径是至少包含有一条在其他独立路径中从未有过的边的路径。应设计适当的测试用例，对软件单元中的独立路径进行测试，特别是对独立路径中的基本路径进行测试。基本路径指在程序控制流图中，通过对控制构造的环路复杂性分析而导出的基本的、可执行的独立路径集合。
  - 边界条件
    - 应测试软件单元在边界处能否正常工作，如测试处理数组的第一个和最后一个元素；测试循环执行到最后一次；测试取最大值或最小值；测试数据流、控制流中刚好等于、大于或小于确定的比较值。

- GB/T 15532-2008 单元测试内容
  - 差错处理
    - 测试软件单元在运行过程中发生差错时，其出错处理措施是否有效。
    - 良好的单元设计要求能预见到程序投入运行后可能发生的差错并给出相应的处理措施。这种出错处理也应当是软件单元功能的一部分。一般若出现下列情况之一，则表明软件单元的出错处理功能包含差错或缺陷：
      (1) 差错的描述难以理解；
      (2) 在对差错进行处理之前，差错条件已经引起系统的干预；
      (3) 提供的差错描述信息不足以确定造成差错的位置或原因；
      (4) 显示的出错提示与实际差错不符；
      (5) 对差错条件的处理不正确；
      (6) 意外的处理不当；
      (7) 联机条件处理 (即交互处理等) 不正确。

- GB/T 15532-2008 单元测试内容
  - 功能
    - 应对软件设计文档规定的软件单元的功能逐项进行测试。
  - 性能
    - 按软件设计文档的要求，对软件单元的性能 (如精度、时间、容量等) 进行测试。
  - 内存使用
    - 检查内存的使用情况，特别是动态申请的内存在使用上的错误 (如指针越界、内存泄露等)。

单元测试的基本流程

- 单元测试的基本流程：
  - 确认测试用例运行的前置条件已经具备；
  - 配置被测模块/被测单元所需环境 (进行全局变量赋值或实体初始化)；
  - 启动测试驱动模块；
  - 设置桩模块；
  - 调用被测模块；
  - 设置预期输出条件判断；
  - 恢复环境 (包括桩的清除)。

- 单元测试用例设计
  - 白盒测试用例设计：覆盖测试用例
    - 作用：找出单元内部控制结构和数据使用可能存在的问题。
    - 陷阱：测试中误判死代码或者冗余代码，并将其删除。
      - 原因：可能是测试用例不够，或者测试者缺乏对整体结构的把握 (尤其测试的是其它开发人员编写的代码)。

- 单元测试用例设计
  - 黑盒测试用例设计：接口测试用例
    - 功能初步测试用例
      - 作用：证明被测单元至少在某种正常情况下能够运行。
    - 功能正面测试用例
      - 作用：找出被测单元对于设计要求的正确输入可能做出的不正确处理。
    - 功能反面测试用例
      - 作用：找出被测单元对于设计要求的错误输入可能做出的不正确处理。
    - 性能测试用例
      - 作用：找出被测单元对于设计要求的性能可能无法实现的错误。

- 单元测试用例设计
  - 单元覆盖测试要求
    - 语句覆盖达到 100%
    - 分支覆盖达到 100%
    - 错误处理路径达到 100%
    - 单元的软件特性覆盖
    - 各种数据特性覆盖

单元测试报告

- 编写目的
  - 对单元测试结果进行整理和汇总，形成正式的测试文档；
  - 为软件单元的评审验收提供依据；
  - 测试报告纳入软件产品配置管理库。
- 软件单元描述
  - 简单描述被测试单元或与之相关单元的产品项目名称、所属子系统、单元要完成的功能、需求和设计要求等。
- 单元结构
  - 根据本单元的控制结构或操作时序，画出其大致的程序流程。
- 测试过程
  - 简要描述本程序单元的测试过程。
- 测试结果
  - 包括代码审查结果、测试用例统计结果。

Thank you!