软件测试 Software Testing

14. 集成测试

程适

cheng@snnu.edu.cn

计算机科学学院

2016 年 10 月 28 日

陕西师範大學
SHAANXI NORMAL UNIVERSITY

Outline

- 系统集成的模式与方法
- 回归测试

集成测试

- 所有功能基本独立的模块经过严格的单元测试以后，接下来需要进行集成测试 (integration test)。
- 集成测试是将已分别通过测试的单元按设计要求组合起来再进行的测试，以检查这些单元之间的接口是否存在问题。

系统集成的模式与方法

- 集成测试前的准备
- 集成测试的模式
- 自顶向下和自底向上集成方法
- 大爆炸与三明治集成方法
- 持续集成

集成测试前的准备

- 人员安排
- 测试计划
- 测试内容
- 集成模式
- 测试方法

集成测试前的准备

- 人员安排：集成测试既要求参与的人熟悉单元的内部细节，又要求能够从足够高的层次上观察整个系统。一般由有经验的测试人员和软件开发者共同完成集成测试的计划和执行。
- 测试计划：集成测试计划在系统设计阶段就开始制定，随着系统设计、开发过程不断细化，最终在系统实施集成之前完成。在这份计划里主要包含的内容有测试的描述和范围、测试的预期目标、测试环境、集成次序、测试用例设计思想、时间表等。

集成测试前的准备

- 测试内容：在经过了单元测试后，需要将所有单元集成到一起，组成一个完整的软件系统。其重点测试内容包括各单元的接口是否吻合、代码是否符合规定的标准、界面标准是否统一等。
- 集成模式：集成方式的选择，要么把所有模块按设计要求一次全部组装起来后进行测试，要么测试是在模块一个一个地扩展下进行，其测试的范围逐步增大。

集成测试前的准备

- 测试方法：集成测试阶段是以黑盒测试为主。在自底向上集成的早期，白盒测试占有较大的比例，随着集成测试的不断深入，这种比例在测试过程中将越来越少，渐渐地，黑盒测试占据主导地位。

集成测试的模式

- 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，如大爆炸模式。
- 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。

集成测试的模式

- 渐增式测试模式需要编写的软件较多，工作量较大，而非渐增式测试开销小。
- 渐增式测试模式发现模块间接口错误早，而非渐增式测试模式晚。
- 非渐增式测试模式发现错误，较难诊断，而使用渐增式测试模式，如果发生错误则往往和最近加进来的那个模块有关。

集成测试的模式

- 渐增式测试模式测试更彻底。
- 渐增式测试模式需要更多的机器时间。
- 使用非渐增式测试模式，可以并行测试。

自顶向下和自底向上集成方法

- 自顶向下法 (Top-down Integration)
- 自底向上法 (Bottom-up Integration)
- 混合策略 (Modified Top-down Integration)

自顶向下法 (Top-down Integration)

- 自顶向下法，从主控模块（“主程序”）开始，沿着软件的控制层次向下移动，从而逐渐把各个模块结合起来。在组装过程中，可以使用深度优先的策略，或宽度优先的策略，其具体步骤是：
  1. 对主控模块进行测试，测试时用桩程序代替所有直接附属于主控模块的模块。
  2. 根据选定的结合策略（深度优先或宽度优先），每次用一个实际模块代替一个桩程序（新结合进来的模块往往又需要新的桩程序）。
  3. 在结合下一个模块的同时进行测试；
  4. 为了保证加入模块没有引入新的错误，可能需要进行回归测试（即全部或部分地重复以前做过的测试）。

自底向上法 (Bottom-up Integration)

- 自底向上测试用“原子”模块（即在软件结构最底层的模块）开始集成以进行测试，具体策略是：
  1. 把底层模块组合实现某个特定的软件子功能的族；
  2. 写一个驱动程序（用于测试的控制程序），协调测试数据的输入和输出；
  3. 对由模块组成的子功能族进行测试。
  4. 去掉驱动程序，沿软件结构自下而上移动，把子功能族组合起来形成更大的子功能族 (Cluster) 。

混合策略 (Modified Top-down Integration)

- 在具体测试中，采用混合策略：
  - 改进的自顶向下法：基本使用“自顶向下”法，但在测试早期，使用“自底向上”法测试少数的关键模块。
  - 混合法：对软件结构中较上层，使用的是“自顶向下”法；对软件结构中较下层，使用的是“自底向上”法，两者相结合。

大爆炸与三明治集成方法

- 大爆炸集成方法（Big-bang Integration）
- 三明治集成方法（Sandwich Integration）
- 改进的三明治集成方法（Modified Sandwich Integration）

大爆炸集成方法（Big-bang Integration）

- 采用大爆炸集成方法，先是对每一个子模块进行测试（单元测试阶段），然后将所有模块一次性地全部集成起来进行集成测试。因为所有模块是一次集成的，所以很难确定出错的真正位置、所在的模块、错误的原因。
- 这个方法并不推荐在任何系统中使用，适合在规模较小的应用系统中适应。

三明治集成方法（Sandwich Integration）

- 三明治方法由两头向中间集成；
- 优点：将自顶向下和自底向上的集成方法有机地结合起来，不需要写桩程序，因为在测试初自底向上集成已经验证了底层模块的正确性。
- 缺点：在真正集成之前每一个独立的模块没有完全测试过。

改进的三明治集成方法

- 改进的三明治集成方法，不仅自两头向中间集成，而且保证每个模块得到单独的测试，使测试进行得比较彻底。

集成方法性能比较

	自底向上	自顶向下	混合策略	大爆炸	三明治	改进的三明治
集成	早	早	早	晚	早	早
基本程序能工作时间	晚	早	早	晚	早	早
需要驱动程序	是	否	是	是	是	是
需要桩程序	否	是	是	是	是	是
工作并行性	中	低	中	高	中	高
特殊路径测试	容易	难	容易	容易	中等	容易
计划与控制	容易	难	难	容易	难	难

持续集成

- 通常系统集成都会采用持续集成的策略，软件开发中各个模块不是同时完成，根据进度将完成的模块尽可能早地进行集成，有助于尽早发现 Bug，避免集成中大量 Bug 涌现。

回归测试

- 无论在进行系统测试还是功能测试时，当发现一些严重的缺陷而需要修正时，会构造一个新的软件包 (Full Build) 或新的软件补丁包 (Patch)，然后进行测试。
- 这时的测试不仅要验证被修复的软件缺陷是否真正被解决了，而且要保证以前所有运行正常的功能依旧保持正常，而不要受到这次修改的影响。

回归测试

- 不希望已发现的程序缺陷被修复了，但出现新的软件缺陷，甚至是更严重的缺陷，没有被发现，结果产品发布出去了。

回归测试

1. 目的
2. 策略及其方法

目的

- 回归测试的目的是在程序有修改的情况下保证原有功能正常的一种测试策略和方法，因为这时的测试不需要进行全面测试，从头到尾测一遍，而是根据修改的情况进行有效测试。
- 程序在发现严重软件缺陷要进行修改或版本升级要新增功能，这时需要对软件进行修改，修改后的程序要进行测试，这时要检验软件所进行的修改是否正确，保证改动不会带来新的严重错误。

软件修改的正确性

- 所做的修改达到了预定的目的，如错误得到了改正，新功能得到了实现，能够适应新的运行环境等；
- 不影响软件原有功能的正确性。

策略及其方法

- 在软件生命周期中，即使一个得到良好维护的测试用例库也可能变得相当大，使得每次回归测试都重新运行完整的测试包变得不切实际，时间和成本约束也不允许进行一个完全的测试。
- 需要从测试用例库中选择有效的测试用例，构造一个缩减的测试用例组来完成回归测试。

回归测试基本过程

- 识别出软件中被修改的部分。
- 从原基线测试用例库 T 中，排除所有不再适用的测试用例，确定那些对新的软件版本依然有效的测试用例，其结果是建立一个新的基线测试用例库 T0。
- 依据一定的策略从 T0 中选择测试用例测试被修改的软件。

回归测试基本过程

- 如果回归测试包不能达到所需的覆盖要求，必须补充新的测试用例使覆盖率达到规定的要求，生成新的测试用例集 T1，用于测试 T0 无法充分测试的软件部分。
- 用 T1 执行修改后的软件。

回归测试方法

- 选择回归测试策略应该兼顾效率和有效性两个方面，下面几种方法，在效率和有效性方面其侧重点是不同的。
  - 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。

小结

- 系统集成的模式与方法
- 回归测试

致谢

谢谢，欢迎提问!