1. 集成测试回顾和总结

1.1概述

总体上来说体系结构阶段开发的测试用例我们认为并不十分成功，在构造阶段，我们参照构造阶段对体系结构设计进行了大量的验证和部分调整，这就导致开发的Stub很多是失效的。所以在真正的体系结构定下来之后，项目小组重新开发了Stub来保证持续集成的正常进行，最后的结果还不坏，可以正常集成，不同的程序员构造的模块可以有效的做为系统的组件。

1.2详细情况----问题与解决方案：

1.2.1

Qusetion:

体系结构设计的结果需要修正。

Reason:

对于“在项目没有真正进入下一阶段之前，无法真正验证当前工作的有效性”，深以为然。想象和实现有较大的差距，可能的原因包括经验的缺乏，设计阶段的思想上不重视，能力不足等。

Solution:

在构造阶段发现体系结构存在问题，虽说体系结构关注的主要是非功能性需求的实现，但是在工程中产生漏洞导致部分功能的实现难度加大，显然是不合适的。所以对体系结构进行了调整，主要调整的内容包括层与层之间的调用接口，进行了更精细的考量，主要以实现为依据。

因此凡是修改了层与层之间接口的情况，重新开发了Stub。

1.2.2

Question:

体系结构原来开发的Stub程序对单元测试阶段造成了困难，进而影响了集成。

Reason:

原有的Stub程序在可测试性上存在较大的问题，导致在单元测试编码期间，发现很难通过返回值判断是够正常执行。

Solution：重新编写的过程中做了改进，在Stub中增加了更多的逻辑判断，方便验证程序的正确性。

1.3总结与经验：

①体系结构设计阶段不能纸上谈兵，一定要形成相对稳定的接口，方便开发Stub和Driver，也可以尽可能保证他们的有效性。

②Stub不能开发的太过于简单，如果方法接口调用全部返回一样的值，那么会对单元测试造成较大的困难，至少成功失败需要区分。

③为了隔离复杂度，可以单独对远程调用，或者是事件驱动这种整体框架进行验证。

1. 单元测试回顾和总结

2.1概述：

诚如作业批改结果所言，mockObject在详细设计阶段基本没怎么写，后来发现这是个失误。原因是老师讲的时候说只需要书写很复杂的类的Mock object，比如选课。但是在真实执行的过程中发现需要的地方还是挺多的。真正构思了每个类的具体实现甚至写好了属性和方法接口之后，才开始开发mockObject比较合适，也有可能是我们代码写得太晚了。还好，最后还是基本上实现了Test-Driver,当然主要是针对较为复杂的逻辑，只有getter和setter的那种类就没有测试了。结果可以接受吧。

2.2测试用例的有效性评估

测试用例大多是两个思路，一个是测试主干正常执行，另一个是边界值的专门测试。应该是颇为有效的。

缺点是只是做到了代码覆盖，可能逻辑组合是会出问题。

2.3总结与经验：

①MockObject的开发尽可能简单，调试MockObject这种事情真是不能忍。

②测试驱动开发会让人迅速体验到成就感，看到那个绿色的状态条颇为高兴，Nice。

③构造要尽量构造可测试的代码，会为测试驱动创造有利的条件。We need testable java.

④测试用例的设计问题，边界测试很好用，首先问题大多在那里出现，其次是往往后面调用出错就恰恰调用的是边界值。