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尚算成功，在构造阶段，体系结构与文档保持一致，我们参照构造阶段对体系结构设计进行了大量的验证和极小部分调整，但详细设计的类定义和小部分类协作，构造阶段进行了较大改动，导致了较多mock失效，最后虽可以正常集成，但耗时耗力，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:

详细设计的部分类定义及类协作的结果需要修正。

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”。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，之前的部分类定义不能满足实现需求。

Solution:

在构造阶段发现详细设计的类协作存在问题，还是在设计阶段并未很好的重视起来，加上经验不足，导致按详细设计搭建的类及类协作不能满足实际开发的需求，。所以对详细设计类协作及部分类定义进行了调整，主要调整的内容包括类成员方法，方法参数与类与类之间的依赖调用逻辑，进行了更精细的考量，主要以实现和设计原则为依据。

其中，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:

详细设计原来为类协作开发的Mock　Object对单元测试阶段造成了困难，进而影响了集成。

Reason:

原有的Object程序基于原有的类定义开发的，而与后期构造重新定义的类及协作不能很好的匹配适用，在可测试性上存在较大的问题，导致在单元测试编码期间，需要重新写Mock Object ,虽然程序简单，但工作量较繁。

Solution：重新编写的过程中做了改进，对类及协作做了认真的思考，较好的利用Mock进行测试隔离，方便验证程序的正确性。

1.3总结与经验：

①详细设计阶段不能纸上谈兵，一定要形成相对稳定的定义和协作，方便开发MockObject，也可以尽可能保证他们的有效性。

②Mock要开发的十分简单，方便程序员定位错误代码。

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：

虽然明白Mock Object的使用可以帮助单元测试隔离待测试单元和其他单元，易于定位出错点，但由于前期大部分针对协作测试而写的Mock Object，在后期由于构造中代码结构，类定义的变化中，不能满足使用，。虽然测试时间较短，但经过努力，Mock Objec还是发挥了一定的作用。所以详细设计的类定义，接口定义等需要深思熟虑，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。还好，最后还是基本上实现了Test-Driver或者Junit,主要是针对较为复杂的逻辑，。

2.2测试用例的有效性评估

测试用例分两步走，先进行了主干正常执行的测试，再进行异常输入及边界值输入的专门测试，检查出了不少漏洞。

缺点是只是做到了代码覆盖，而且前期的系统测试用例还是有部分不能使用，幸好其中大部分都可转为界面测试，没白写，但需求阶段的系统测试用例测试覆盖范围较小，不能满足测试需求。

2.3总结与经验：

1.前期，需求，详细设计一定要好好做，不然构造，测试阶段会很痛苦

2.对于已有的设计，不要轻易改动，要做决策，不然测试时，前期的苦心写的文档可能白费，

3.重视stub,mock的使用，隔离测试模块，便于定位出错点，同时方便集成测试

4.每天都要jenkins跑一跑，代码测试不通过要及时修改

5在构造或测试阶段中，对于出现过的缺陷或错误要及时整理备份，便于以后出现同样的问题做参考，及做项目分析.