**快递物流系统**

**(Express Logistics System)**

**测试回顾文档**



**学 院：南京大学软件学院**

**团 队：伪装者W小组**

**成 员：张词校 张家盛 王丽莉 魏彦淑**

**完成日期：2015年12月20日**

目录

[1.单元测试回顾 2](#_Toc438417274)

[1.1概述 2](#_Toc438417275)

[1.2测试用例的有效性分析 2](#_Toc438417276)

[1.3采取的弥补手段 2](#_Toc438417277)

[1.4小结 2](#_Toc438417278)

[2.集成测试回顾 3](#_Toc438417279)

[2.1概述 3](#_Toc438417280)

[2.2测试用例的有效性分析 3](#_Toc438417281)

[2.3采取的弥补手段 3](#_Toc438417282)

[2.4小结 3](#_Toc438417283)

# 1.单元测试回顾

## 1.1概述

总体上来说，我们组使用了TestDriver的思想指导开发。在构造阶段，对于有较为复杂的逻辑处理和内外部接口的类，编写JUnit测试用例或者使用mock object以隔离外界环境的影响进行开发。开发完成之后，先进行了JUnit自动化测试，再通过人工与界面进行交互测试。测试的方法包括随机测试和边界值测试，部分复杂逻辑采用了白盒测试中的语句覆盖测试。

## 1.2测试用例的有效性分析

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。测试结果比较有效，能够找到一些比较隐蔽的错误。

但是由于前期所写的单元测试只涉及部分复杂的逻辑结构，且对应要用的mock object 写的也比较少，这就导致了后期测试的困难。并且一开始考虑的不是很周到，导致前期写的一些测试到构造后期无法使用。实际上在真正构思好了属性和方法接口后再去开发 mock object比较合适。另外，许多之前写的单元测试到后期并没有实际的应用，这说明针对测试编程做的还不是很好。

## 1.3采取的弥补手段

对于一些复杂的逻辑结构，在后期进行了代码评审。

在构造后期对复杂的逻辑结构重写了能够使用的单元测试。

## 1.4小结

1、在对整个项目有一个成熟的详细设计之后再进行mock object和单元测试JUnit的开发，不然会对后期的测试造成很大的不便。

2、要尽量构造可测试的代码，为测试取得创造有利条件，由于在构造阶段未考虑测试的因素，因此导致有些代码片段测试起来比较困难。

3、测试用例的设计中，关于边界值的测试往往能够发现隐藏的问题。在设计测试用例时，应多考虑这方面测试方法的使用。

# 2.集成测试回顾

## 2.1概述

在体系结构设计阶段结束的时候，为每一层开发了Stub和部分层开发了Driver。在构造阶段采用自底向上的构造方式，每开发一个层次，就使用上一个层次的driver，进行maven build。逐步将上层集成至已经开发完成的代码当中。

## 2.2测试用例的有效性分析

总体上来说，体系结构阶段开发的测试用例并不十分成功。在构造阶段，我们对体系结构的设计进行了部分调整，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，这就导致开发的stub与driver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和driver来保证持续集成的正常进行。我们组的持续集成仅仅集中在一个很短的时间段，因此持续集成的效果并不是特别明显，测试用例的有效性并不高，而且刚开始的时候还出现了集成大爆炸现象。此外，由于一开始对于桩和驱动的理解并不是很透彻，不明白该如何在实际操作中利用桩和驱动进行集成测试，因此在构造初期和中期并没有真正进行持续集成。

## 2.3采取的弥补手段

在对体系结构设计进行更精细的考量与调整后，再重新开发stub和driver。

## 2.4小结

1、在体系结构阶段一定要对设计做充分的考虑，形成相对稳定的接口，这样才能使在体系结构阶段写的桩和驱动在后期能真正被用来集成。

2、Stub开发不能过于简单，不能简单靠惯性思维。如果方法接口调用全都返回一样的值，会对单元测试造成较大的困难。许多stub和driver到后期直接被舍弃，就是因为一开始想的简单，给后期集成带来了困惑。