目录

[一、 单元测试回顾 2](#_Toc469867235)

[(一) 概述 2](#_Toc469867236)

[(二) 测试用例的有效性分析 2](#_Toc469867237)

[(三) 采取的弥补手段 2](#_Toc469867238)

[(四) 度量数据 3](#_Toc469867239)

[(五) 小结 3](#_Toc469867240)

[二、 单元测试回顾 4](#_Toc469867241)

[(一) 概述 4](#_Toc469867242)

[(二) 测试用例的有效性分析 4](#_Toc469867243)

[(三) 采取的弥补手段 4](#_Toc469867244)

[(四) 度量数据 4](#_Toc469867245)

[(五) 小结 5](#_Toc469867246)

# 单元测试回顾

## 概述

大体上使用了DriverTest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外部内部接口的类，先编写JUnit测试用例。并且编写了mock object以隔离外界环境对此类的影响，而后进行开发，开发完成之后，先进行JUnit自动化测试，通过之后提交。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。

## 测试用例的有效性分析

测试用例大多是两个思路，一个是测试主干的正常执行，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。

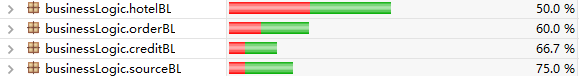
但是由于前期所写的单元测试只涉及部分复杂的逻辑结构，且对应要用的mock object 写的也比较少，这就导致了后期测试的困难。并且一开始考虑的不是特别周到，导致前期写的一些测试到构造后期作用甚微。实际上在真正构思好了属性和方法接口后再去开发 mock object比较合适。另外，所做的单元测试只是做到了代码覆盖，所以导致部分逻辑组合的问题未发现。

## 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并进行了路径覆盖测试。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。

## 度量数据

（单元测试覆盖度）使用EclEmma工具进行度量。由于主要测试的部分为bl层，因此所测试的覆盖率为bl层的单元测试覆盖率。度量结果如下图示例。



2

3

4

5

6

## 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。

# 单元测试回顾

## 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 install。逐步将上层集成至已经开发完成的代码当中。

## 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。但其实真正的构造阶段也是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。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，因此在构造初期只是进行简单的持续集成。

## 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。

## 度量数据

（单元测试覆盖度）100%

## 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，如果方法接口调用全都返回的值过于简单，会对单元测试的开发造成较大的困难。