**进销存系统**

**软件测试报告**

**（V1.2）**

**南京大学软件学院**

**羽见青柠工作组**

金翠黄涵倩王宁(79)严顺宽

**2014年12月15日**

**目录**

目录

[1.引言 3](#_Toc406437495)

[1.1文档标识 3](#_Toc406437496)

[1.2范围 3](#_Toc406437497)

[1.3参考文献 3](#_Toc406437498)

[1.4术语和缩略语 3](#_Toc406437499)

[2.详细情况 4](#_Toc406437500)

[2.1总测试结果概述 4](#_Toc406437501)

[2.1.1测试用例设计 4](#_Toc406437502)

[2.1.2测试环境与配置 5](#_Toc406437503)

[2.1.3 测试方法和工具 6](#_Toc406437504)

[2.2详细测试结果 8](#_Toc406437505)

[2.2.1.系统测试 8](#_Toc406437506)

[2.2.2.集成测试 11](#_Toc406437507)

[2.2.3.单元测试 14](#_Toc406437508)

[2.2.3评估产品质量 16](#_Toc406437509)

[2.2.4度量数据总结 16](#_Toc406437510)

[2.3决策理由 16](#_Toc406437511)

[2.4总结与建议 17](#_Toc406437512)

[2.4.1总体性评价 17](#_Toc406437513)

[2.4.2安装环境 18](#_Toc406437514)

[2.4.3经验总结 18](#_Toc406437515)

[3.其他 19](#_Toc406437516)

[3.1文档修订历史 19](#_Toc406437517)

# 1.引言

## 1.1文档标识

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文件状态：  [√] 草稿  [ ] 正式发布  [ ]正在修改 | 报告编号： | 01 | | |
| 当前版本 | 1.0 | | |
| 编写人 | 金翠，黄涵倩，王宁，严顺宽 | 编写日期 | 2014.12.13 |
| 审批人 |  | 审批日期 |  |
| 保密级别 | 灰常高！！ | | |

## 1.2范围

本测试报告为进销存项目的测试报告，目的在于总结测试阶段的测试以及分析测试结果，描述系统是否符合进销存系统的需求。预期参考人员包括用户、测试人员、开发人员、项目管理者、其他质量管理人员和需要阅读本报告的高层经理。

## 1.3参考文献

1.[IEEE-829-2008]

2.进销存系统用例文档

3.进销存系统需求规格说明文档

4.进销存系统测试用例文档

## 1.4术语和缩略语

|  |  |
| --- | --- |
| **术语或缩略语** | **全意** |
| HCI | 持续集成 |
| 严重缺陷 | 系统发生死机，异常，终止，返回错误信息的情况。 |
| LOC | 代码行数 |
| NULL Pointer Exception | 空指针异常，一般产生于没有正确初始化对象或者是对象赋值有误 |

# 2.详细情况

## 2.1总测试结果概述

进销存系统的测试从2013-12-4开始持续到2013-12-16，共持续12天，按时完成测试计划。

### 2.1.1测试用例设计

1.系统测试

※概述：系统测试以功能性为主。(详见进销存系统需求测试用例及回顾)

1. 系统实现的主要功能

商品分类管理、商品管理、库存查看、客户管理、制定进货单、制定进货退货单、制定销售单、制定销售退货单、账户管理、制定收款单、制定付款单、用户管理、促销策略制定、单据审批、库存盘点、库存赠送、 库存报警、制定现金费用单、查看销售明细表、查看经营明细表、查看经营情况表日志查看、红冲操作、期初建账、库存报溢报损

②系统为了安全性实现的功能

登录功能

系统管理人员设置用户的密码，可查看系统日志，系统的状态转换和权限控制。

※测试用例选择：主要是随机测试，应用等价类划分法和边界值分析法对需求阶段制品中的系统测试用例进行了一定的调整和修正。

2.集成测试

※概述：使用持续集成，体系结构设计阶段为每一层编写了Stub和Driver,用自动化集成工具完成，在构造阶段完成。

※测试用例选择：

随机选择+边界值分析

3.单元测试

※概述:

①测试了复杂的业务逻辑，包括销售，进货，收款，商品管理，单据审批，系统的状态控制，在构造阶段对业务逻辑层的逻辑进行了单元测试。

③数据文件的读写进行了单元测试，确保数据层的操作正确。

※测试用例选择：

随机选择+边界值分析

### 2.1.2测试环境与配置

#### 2.1.2.1软硬件环境

表2.1.2.1 软硬件环境

|  |  |  |
| --- | --- | --- |
| 硬件环境 | 应用服务器 | 客户端 |
| 硬件配置 | CPU:Intel®Core™i5-4200U CPU@1.60GHZ 2.30GHZ  Memory:4GB  HD:Intel GMA HD 3000 | CPU:Intel®Core™i5-4200U CPU@1.60GHZ 2.30GHZ  Memory:4GB  HD: Intel GMA HD 3000 |
| 软件配置 | OS:windows7旗舰版  JDK: 1.7  Tomcat :8.0.14  Maven:3.2.3 | OS:windows7旗舰版  OS:Ubuntu12.1.3  JDK: 1.7  JRE: 1.7 |
| 网络环境 | 10M LAN | 10M LAN |

#### 2.1.2.2网络拓扑



图2.1.2.2应用服务器端和客户端（分布式系统）

### 2.1.3 测试方法和工具

表2.1.3 测试方法和工具

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 分类标准 | 类别 | 测试方法 | 测试工具 | 备注 |
| 测试对象 | 单元测试 | 黑盒与白盒测试相结合。  部分使用TestDriver进行开发。对较为复杂的逻辑使用了语句覆盖的白盒测试，其他大多使用随机测试。 | Junit4.11.0 |  |
| 集成测试 | CI | Jenkins |  |
| 系统测试 | 参照软件需求规格说明，手工测试 |  |  |
| 测试目标 | 功能 | 黑盒、手工、回归 | 随机测试 |  |
| 性能 | 黑盒、手工、回归 |  |  |
| 安全性 | 黑盒、手工、回归 |  |  |

## 2.2详细测试结果

### 2.2.1.系统测试

1.测试活动总结：

根据学习内容和系统需求变更，对需求阶段形成的需求测试用例进行了一定程度的修改。以功能性测试为主，对进销存系统进行了系统测试。这也是测试阶段主要完成的任务，测试活动时间较为紧张，通过增加工作时间，顺利完成预定的测试计划。

2.测试任务的结果总结：

对比系统需求规格说明书，对每项需求进行了测试。就已测试的部分而言，系统能够完成规格说明提出的所用功能性需求和其他的非功能性需求。

3.缺陷和辨析总结：

表2.2.1-1 系统测试重要缺陷分析表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块名称** | **缺陷编号** | **简要描述** | **分析结果** | **测试通过(pass/partially pass/fail)** |
| stockdata | 1 | 修改商品信息（进货价格）时报空指针异常 |  |  |
| userui | 2 | 系统日志功能不能用 |  |  |
| receiptui | 3 | 单据审批之后没有刷新表格 |  |  |
| stockui | 4 | 点击刷新按钮后表格没有重新渲染 |  |  |
| stock | 5 | 当审批通过后库存报损单显示在了库存报溢单表格中 |  |  |
| promotionbl | 6 | 执行代金券促销策略时报空指针异常 |  |  |
| mainui | 7 | 同一账号可以同时登录 |  |  |
| financeui | 8 | 收付款单，现金费用单转账金额没有进行负数检查 |  |  |
| receiptui | 9 | 勾选复选框并不能进行批量审批 | 没写监听 |  |
| financedata | 10 | 创建现金费用单时会报错 |  |  |
| mainui | 11 | 多条信息产生时，只显示了一条信息 |  |  |
| financeui | 12 | 查看三表界面，三表没有显示 |  |  |
| finance | 13 | 期初建账功能不能使用 |  |  |
| receiptui | 14 | 查看三表界面，刷新，按日期查看，各种筛选，搜索框不能使用 | 没有监听 |  |
| memberui | 15 | 添加客户时，信息没写全也可以通过 |  |  |
| memberui | 16 | 修改客户时可以修改客户的类型，这是应当被禁止的 |  |  |
| purchaseui | 17 | 允许重复创建进货退货单 |  |  |
| purchaseui | 18 | 查看进货单详情没问题，但查看进货退货单详情时会报错 |  |  |
| saleui | 19 | 不能查看详情，不能搜索，不能刷新 | 没写监听 |  |
| saleui | 20 | 创建销售退货单时会报空指针 |  |  |
| purchaseui | 21 | 创建进货单成功后，返回主界面一个单据都没有 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

表2.2.1-1系统测试残留问题和未解决问题列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块名称** | **缺陷编号** | **性质** | **简要描述** | **原因分析** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

4.产品质量评估

概述：总体来说，系统基本完成了所有系统需求规格说明书中对功能性方面要求的内容，对于其他的非功能性需求，做出了一定的努力，比如为了安全，将设置了系统日志和用户密码；为了易用性，更新了人机交互设计；但是易用性，性能，都没有经过测试工具的验证，需要持续改进。

备注：系统测试中发现的缺陷数目并不多，我们认为这是由于在集成测试和单元测试中消除了很多缺陷。

结论：达成预定目标。

5.度量数据

1.系统测试缺陷数据：

严重缺陷：?

一般缺陷：?

无影响缺陷：?5(+-1)（主要体现在界面提示信息设置的合理性，界面跳转的合理性等方面与软件工程艺术性相关的东西，上节缺陷分析中省略）

2.系统测试覆盖度：

对用例的覆盖度：100%，详见系统测试用例文档中的覆盖情况度量。

### 2.2.2.集成测试

1.测试活动总结

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，除了对数据文件的读写外，其余全部采用自顶向下的开发结构，每开发一个层次，就使用下一个层次的Stub，进行maven build。逐步将下层集成至已经开发完成的代码当中。

2.测试任务的结果总结

任务1：

描述：集成Data层对数据文件的操作。

编写了Data层对数据文件读写进行封装和实现的FileUtility以及其实现类，用FileUtilityDriver对其进行了驱动测试，以便以后集成。

结论：通过。足以完成需求。

问题：无

任务2：

描述:集成UI与BL层。

完成业务逻辑层之后将UI层中使用Stub的部分替换为实现代码将BL层集成进系统。

结论：最终通过。

任务3：

描述：集成Data层

完成Data层之后将BL层中使用Stub的部分替换为实现代码，将Data层集成进系统。

结论：最终通过。

3.缺陷和辨析

表2.2.2-1 集成测试严重缺陷分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块名称** | **缺陷编号** | **简要描述** | **分析结果** | **测试通过(pass/fail)** |
| BL | 2.1 | ui层与bl层之间采用controller和controllerFactory进行交互，但是ui层采用角色进行划分，bl层使用功能进行划分，导致层与层之间的依赖关系过于复杂不符合分层风格的设计原则。 | 在BLService一个子层次，专门用来处理ui层任务的分配。 | pass |
| AccountBL  &AccountData | 2.2 | Build报错，显示NullPointereException。 | 系统应该在BL层添加对结果返回值为NULL的处理。 | fail->pass |
| GoodsPO | 2.3 | GoodsPO对象，经由rmi通信后部分数据丢失 | Client与server两者的GoodsPO对象，成员名不一致，导致data层数据丢失 | fail->pass |
| CommodityPO | 2.4 | serializable 类串行版本号不一致，导致rmi通信失败 | Client与server端的序列化文件应该有着相同的串行版本号 | fail->pass |
|  |  |  |  |  |

表2.2.2-2 集成测试残留缺陷及未解决问题列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块名称** | **缺陷编号** | **性质** | **简要描述** | **原因分析** |
| ReceiptData | 2.11 | 无影响缺陷 | 筛选算法有不科学性，但是，总归是可以使用的 | 应该考虑时间复杂度和空间复杂度来提升效率 |
| GiftBL | 2.12 | 无影响缺陷 | 获得历史库存赠送支出需要遍历所有库存赠送单据，计算出总的支出 | 当系统单据较多时，耗时不能忍 |
|  |  |  |  |  |
|  |  |  |  |  |

4.产品质量评估：

概述，在集成过程中使用Jenkins进行自动化构建，最终构建结果是SUCCESS。期间出现build失败的情况，查找缺陷，定位缺陷并更正。

结论：达成预期目标，测试通过。

5.度量数据

1.集成测试缺陷数据：

严重缺陷：?

一般缺陷：?

无影响缺陷：?

2.集成测试覆盖度：

被测试的模块总数：?

模块总数：?

模块覆盖度：100%

备注：模块java/main/二级目录下包的个数进行统计。

### 2.2.3.单元测试

1.测试活动总结：

大体上使用了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。

2.测试任务的结果总结：

保证所有进行单元测试的类，方法自动化测试通过。也即保证了该方法就有系统所期待的输入输出和不变式。

3.缺陷和辨析

表2.2.3-1 单元测试重要缺陷分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块名称** | **缺陷编号** | **简要描述** | **分析结果** | **测试通过(pass/fail)** |
| stockui | 2.3 | 商品分类使用jtree表示，将jtree对象整体写入文件，需要的时候整体读出。新建一个树节点然后立即删除，报数组越界异常 | UI与底层数据不一致，treemodel不匹配，刷新jtree时越界 | fail->pass |
| GiftReceipt | 2.4 | 对象声明时，因构造函数参数较多，有两个参数位置反了，导致生成的对象不正确 | 需要仔细匹对构造函数参数是否顺序一致，类型一致 | pass |
| GoodsClassData | 2.6 | Data层文件对出显示空指针异常 | 没有考虑系统使用之初是没有数据文件的，加上没有没有数据的判断（对返回值为null的处理） | fail->pass |
|  |  |  |  |  |

表2.2.2-2 单元测试残留缺陷及未解决问题列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块名称** | **缺陷编号** | **性质** | **简要描述** | **原因分析** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

4.产品质量评估：

1.概述：单元测试全部测试用例Junit自动化测试SUCCESS，达到预定的目标，对复杂逻辑基本实现代码覆盖。

2.备注：上表只显示了系统筛选算法的缺陷分析，主要有两个原因。①这是系统中最复杂的逻辑，而且复杂度基本集中在构造阶段，对外接口较为简单，实现较为复杂。②在开发过程中没有记录debug的情况，没有历史记录。

3.结论：质量达到预期标准，可以进行后续的打包安装工作。

5.度量数据：

1.缺陷度量：

缺陷数目难以找到真实可靠的数据。

2.覆盖度度量：

使用Maven Cobertura工具进行度量。度量结果如下图示例

？

图2.2.3.5 覆盖度度量

总体来说对于有较为复杂业务逻辑（这里指不是只含有getter和setter）的BL层和Data层基本实现了方法的全覆盖。

### 2.2.3评估产品质量

经过对系统测试，集成测试以及单元测试结果的分析总结以及对测试过程的回顾和反思。按照系统仍然存在的缺陷数目大致与最后一次测试缺陷数目持平的原则，我们认为系统的质量在功能性，部分非功能性上达到了预期，符合客户需求，可以交付使用。

### 2.2.4度量数据总结

单元测试的覆盖率较低（低于对10000+LOC）的要求（50%）：原因在于大量LOC分布在ui层，尤其是使用可视化界面编辑工具产生的不可修改代码当中，对这些代码，难以同时也没有必要开发测试用例。而系统的主要复杂度只要集中在BL层，对BL层的单元测试测试用例覆盖度更有意义。所以我们认为测试的质量是得到保证的。

## 2.3决策理由

1.系统测试：

通过：完全按照测试用例设计的输入，得到预期的输出，不变式（如果有的话）符合要求。

不通过：对于所有的输入，系统没有给出预期的输出，或者发生严重出现严重错误（死机，提示错误等）

有条件通过：①在一定条件下，产生符合预期的输出，在其他条件下产生错误的输出或者无输出。②对于部分输入，产生符合预期的输出，对于其他输入产生错误的输出或者无输出。

2.集成测试：

通过：maven clean build输出结果为SUCCESS

不通过：maven clean build 输出结果为FAILTURE

有条件通过：无

3.单元测试：

通过：junit自动测试执行结果为全部通过SUCCESS

不通过：junit自动测试执行结果为全部FAILTURE

有条件通过：junit自动测试执行结果部分SUCCESS部分FAILTURE

## 2.4总结与建议

### 2.4.1总体性评价

产品是进销存系统，业务目标是提高用户满意度，总体来说我们任务业务目标实现的可能性很大。首先用户满意度大半取决于人机交互设计，为此，专门进行了二次人机交互设计，对艺术性，可用性，易用性，易学性做了专门的考量。功能齐全，覆盖了系统用例文档中的全部用例。

同时，项目失败的可能性是存在的，比如对单据审批没有做到完全的仿真。

另使用文件系统对当下打包安装有利，但是为后续大批量数据的积累埋下了隐患，可能会对系统的性能造成一定的影响。没有进行压力测试，并发性是可能达不到实际要求。

### 2.4.2安装环境

安装环境为windows7（经过测试），客户下载压缩包，解压之后直接使用单击.exe文件就可以使用。

### 2.4.3经验总结

①重视需求，尽可能的在需求阶段发现所有的需求（实际上认为不可能在需求阶段发现所有的需求），并形成清晰的说明，有助于后续的开发。反之，如果在需求中存在没有发现的重要需求，会严重影响系统的逻辑完备性，功能全面性，大大加大后续工作量，并且直接影响设计的质量。项目进行中，在体系结构设计阶段新增了需求，感受到工作量远比需求阶段发现需求多，认为需要提高需求工作的质量。

②体系结构设计阶段最好由核心程序员为主导，（如果可能的话，负责最复杂业务逻辑的程序员应该是核心程序员），以实现为依据进行规划设计。明确层与层之间的调用接口。

③制定规约，所有人在体系结构设计完成之后，必须按照体系结构设计产生的项目框架进行详细设计和后续的构造。

④详细设计阶段中可以更多的加入构造阶段的东西，我们发现在真正构造之前，所有详细设计的结果基本都是无效的不现实的，但是为了避免出现build& fix的情况，这个阶段的构造应该以编写接口为主。

⑤在构造阶段一定不要轻易修改接口，会导致一系列严重的问题，依赖于此的程序可能会崩溃，最直观的是配置管理中的版本控制会出现版本冲突，持续集成会中断。在实现的时候可以考虑只增不改，增量式开发。

⑥任务的分配要科学合理，量体裁衣，按照个人的优缺点和劳动生产力进行分配。

⑧一致性的问题，提前商量命名规范遵循一定的命名规范，出于可读性的考量。

# 3.其他

## 3.1文档修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **版本信息** | **变更人** | **主要内容** | **备注** |
| 1.0 | 黄涵倩 | 创建 | 2014-12-13 |
| 1.1 | 严顺宽 | 完善部分缺陷分析记录 | 2014-12-14 |
| 1.2 | 王宁，金翠 | 完善缺陷记录，测试用例，编制目录，完善整个文档 | 2014-12-15 |