**NJWU选课系统**

**(Course Selection System)**

**总测试报告**



**南京大学软件学院Aurora小组**

2013-12-9

目录

[1．引言 3](#_Toc374400404)

[1.1文档标识 3](#_Toc374400405)

[1.2范围 3](#_Toc374400406)

[1.3参考文献 3](#_Toc374400407)

[1.4术语和缩略语 3](#_Toc374400408)

[2.详细情况 4](#_Toc374400409)

[2.1总测试结果概述 4](#_Toc374400410)

[2.1.1测试用例设计 4](#_Toc374400411)

[2.1.2测试环境与配置 4](#_Toc374400412)

[2.1.3 测试方法和工具 5](#_Toc374400413)

[2.2详细测试结果 6](#_Toc374400414)

[2.2.1 测试活动总结 6](#_Toc374400415)

[2.2.2测试任务的结果总结 6](#_Toc374400416)

[2.2.3缺陷与辨析 6](#_Toc374400417)

[2.2.4评估产品质量 6](#_Toc374400418)

[2.2.5度量数据总结 6](#_Toc374400419)

[2.3决策理由 6](#_Toc374400420)

[2.4总结与建议 6](#_Toc374400421)

[3.其他 6](#_Toc374400422)

[3.1文档修订历史 6](#_Toc374400423)

# 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： | 刘璟 | 编写日期 | 2013.12.2 |
| 审批人： |  | 审批日期 |  |
| 保密级别： |  | | |

## 1.2范围

本测试报告为CSS(南京某大学选课系统)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.NJWU选课系统用例文档

3.NJWU选课系统需求规格说明书

## 1.4术语和缩略语

|  |  |
| --- | --- |
| **术语或缩略语** | **全意** |
| NJWU | 南京某大学 |
| CSS | 选课系统 |
| 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

CSS系统的测试从2013-12-2开始持续到2013-12-15，共持续13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NJWU选课系统需求测试用例及回顾)

1. 系统实现的主要功能

教务处老师发布教学框架，查看教学计划，发布通识，通修课程。

院系教务员发布课程，录入教学计划。

教师完善课程信息，录入成绩。

学生选课，退选，补选。

1. 系统为了安全性实现的功能

登陆功能

教务处老师设置选课时间，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，包括选课，排课算法，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 酷睿i5 2520M  CPU2.5GHZ stepping 01  Memory:4GB  HD:Intel GMA HD 3000 | CPU Intel 酷睿i5 2520M  CPU2.5GHZ stepping 01  Memory:4GB  HD: Intel GMA HD 3000 |
| 软件配置 | OS:windows7旗舰版  JDK: 1.7  Tomcat :7.0 | 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。 | Junit3.8 |  |
| 集成测试 | CI | Hudson.3.1.0 |  |
| 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CSS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)** |
| Plan.observe | 1.1 | 非教务员角色查看教学计划出现NULL Pointer Exception | 由于没有PlanController没有在构造函数中对PLanVO做，而是在其他函数中做了初始化，这就导致查看的逻辑对其他逻辑的依赖，出现错误。 | Fall->pass |
| Course.LaunchGeneral | 1.2 | 教务处老师发布通识通修课程的结束后，无法查看。 | 由于系统没有及时写回文件，而查看逻辑设计的是直接从Data层调用数据，直接调用查看的接口。会出现问题。 | Fall->pass |
| Processmng.State.SetState | 1.3 | 教务处老师设置教学进度的具体landmark时，只能设置无法修改。 | 不允许进行修改，如果进行修改，而状态转换已经发生，会对系统造成不可预计的后果。 | Partially pass |
| Procesmng.NoticeReceiver.getNotice | 1.4 | 教务处老师设置发送通知后，各个角色的用户都接收了一样通知。 | 在调用的接收接口方法的实现中，没有对角色进行判断。 | Fail->pass |
| Procesmng.NoticeController.sendNotice | 1.5 | 教务处老师发通知之后如果选择Role.ALL，但教务员接不到通知。 | 教务员老师的NoticeReceiver中没有添加Role.ALL的识别。 | Fail-pass |
| Course.Select | 1.6 | 当没有选中任何课程的时候，提示选课成功，并导致排课发生异常错误。 | 当没有选中任何课程的时候，系统仍然根据系统时间，生成一条课程内容为空的选课记录。 | Fall->pass |
| Course.Reselect | 1.7 | 补选没有立即给出已经选中的提示。 | 设计中补选采取的是先到先得的算法。  而在实现中为了可变更性，把算法封装，做了Strategy所以应该给对外接口中增加显示即时提示的方法。 | Fall->pass |
| Course.Drop | 1.8 | 学生可以退选必修课程。 | 选课模块的权限判断有问题。 | Fall->pass |
| Course.Select | 1.9 | 超过系统规定的选课时间，仍然可以成功生成选课记录。（更为严重的是即使选落后系统状态跳转后任意时间也有可能生成选课记录）。 | 由于对系统状态的判断逻辑处于选课逻辑的初始位置，没有在生成选课记录的时候更细客户端系统状态的副本，使得仍然看似可以选课。需要在生成选课记录的时候加入对系统状态的判断。 | Fail->pass |
| Course.Launch | 1.10 | 可以发布课程号相同的课程 | 1. 缺少对课程号是否已经存在的课程判断。 2. 存在及时性的问题，即不同客户同时发布课程，此时课程还没有写回，服务器端会会认为不存在此课程好可以创建。可以使用及时写回的策略。 | Fail->pass |
| Course.addCourseInfo | 1.11 | 任课老师添加课程信息的时候，可以覆盖教务员老师的部分内容。 | 没有关于是否覆盖的逻辑判断。 | Fail->pass |
| Course.Select.ReachBound | 1.12 | 对选课门数上限的控制存在问题，当发生同一账号不同客户机登陆的时候，会出现问题。 | 把对门数控制的逻辑设置为单件。 | Fail->pass |
| Grade.GPA | 1.13 | 分学期计算的GPA有问题 | 方法调用的参数列表有问题 | Fail->pass |
| Teacher.lookStu | 1.14 | 任课教师查看自己某门课程的学生列表出现列表不完整的情况 | 补选选中的学生没有包含在内。 | Fail->pass |

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** |
| Processmng.State.SetState | 1.3 | 一般（可忽略） | 教务处老师设置教学进度的具体landmark时，只能设置无法修改。 | 不允许进行修改，如果进行修改，而状态转换已经发生，会对系统造成不可预计的后果。  预计的改善包括对教务处老师的修改时间做一定的限定，及处于当前系统状态时可以修改以后状态的标志时间。 |
| Processmng.notice.sendNotice | 无 | 一般（可忽略） | 教务处老师不能同时选中多个角色发送通知，易用性有问题。 | 数据存储和读取的编码存在难度。 |

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，将用户密码编码；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：14（发现并移除）

一般缺陷：2（发现但没有移除）

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 |
| courseBl&  courseUI | 2.2 | Build报错，NULL Pointer Exception | 原因是ConrseController没有正确初始化CourseController中的List属性，导致插入出错。 | Fail->pass |
| couresBL&  courseUI | 2.3 | Build报错，系统显示格式不正确。 | 系统ui层的VO中Date没有使用系统指定的统一的编码方式，少写了一次规范格式的函数调用。 | Fail->pass |
| CourseBL  &courseData | 2.4 | 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 |
| CourseBL  &courseData | 2.5 | Rmi 和Remote Exception。 | 在BL层的 DataServic类对象的初始化(应该通过RMI从远程获取本地代理)。CourseController中实现采用的是本地创建，发生异常。 | Fall->pass |
| Course.BL和Selection.BL | 2.6 | NullPointerException | CourseBL调用SelectionBL的时候，Record没有正确初始化。 | Fail->pass |
| DataService | 2.7 | 没有办法创建DataService的本地对象。 | 客户端服务器端目录结构有一定差别，类的相对路径不匹配 | Fail->pass |
| FileUtility | 2.8 | Ui层得到的数据中文出现乱码。 | 应该加上使用UTF-8类读取。  文件读取的测试用例只使用了英文。 | Fail->pass |
| Ui | 2.9 | ui层登陆界面之后界面跳转有问题。 | 界面跳转的MainFrameController没有得到正确的参数做为属性 | Fail->pass |
| gradeBL | 2.10 | Grade模块调用选课记录模块的dataService层提供的接口是出现NullPointerException | SelectionDataService给出的接口不足以初始化grade类中的所有属性，应该自行初始化， | Fail->pass |
| CourseBL | 2.11 | 安排课程的算法科学性不够，可能会出现大四学生不能毕业的情况。 | 需要写一个调整因子更多的算法，当前这个可以作为测试版本。 | Partially pass |
| SelectionBL&  User | 2.12 | 选课逻辑调用用户信息的DataService接口时，出现异常错误。 | SelectionBL中调用的其他方法修改了对User调用的属性的状态和权限信息。 | 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** |
| CourseBL | 2.11 | 无影响缺陷 | 选课算法有不科学性。  但是，总归是可以使用的。 | 应该考虑更多的调整因子以影响排课的结果。 |

4.产品质量评估：

概述，在集成过程中使用hudson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：12

一般缺陷：0

无影响缺陷：1

2.集成测试覆盖度：

被测试的模块总数：65

模块总数：65

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)** |
| **CourseAssignment** | 3.1 | 排课算法返回的列表超过上限人数 | 循环判断多了一次 | Fail->pass |
| **CourseAssignment** | 3.2 | 排课算法无限循环 | 算法中使用每次删除优先队列中 第一个元素，但是没有给表示队头的header变量赋值。 | Fail->pass |
|  |  |  |  |  |

表2.2.3-2 单元测试残留于未解决问题列表

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

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：上表只显示了系统排课算法的缺陷分析，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

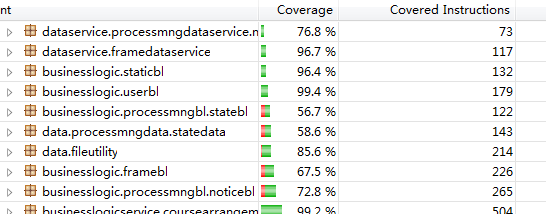


图2.2.3.5 覆盖度度量

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

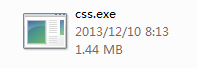
产品是为NJWU开发的选课系统，业务目标是提高用户满意度，总体来说我们任务业务目标实现的可能性很大。首先用户满意度大半取决于人机交互设计，为此，专门进行了二次人机交互设计，对艺术性，可用性，易用性，易学性做了专门的考量。功能齐全，覆盖了系统用例文档中的全部用例。

同时，项目失败的可能性是存在的，比如对选课没有做到完全的仿真，没有考量校区。

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** |
| 1.0.1 | 刘璟 | 创建 | 2013-12-2 |
|  |  |  |  |