**NBA信息查询平台**

**软件测试报告**

**（V1.2）**

**南京大学软件学院**

**羽见青柠工作组(15)**

金翠黄涵倩王宁(79)严顺宽

**2015年4月19日**

目录

[1.引言 3](#_Toc406495461)

[1.1文档标识 3](#_Toc406495462)

[1.2范围 3](#_Toc406495463)

[1.3参考文献 3](#_Toc406495464)

[1.4术语和缩略语 4](#_Toc406495465)

[2.详细情况 4](#_Toc406495466)

[2.1总测试结果概述 4](#_Toc406495467)

[2.1.1测试用例设计 4](#_Toc406495468)

[2.1.2测试环境与配置 24](#_Toc406495469)

[2.1.3 测试方法和工具 25](#_Toc406495470)

[2.2详细测试结果 26](#_Toc406495471)

[2.2.1.系统测试 26](#_Toc406495472)

[2.2.2.集成测试 35](#_Toc406495473)

[2.2.3.单元测试 39](#_Toc406495474)

[2.2.3评估产品质量 42](#_Toc406495475)

[2.2.4度量数据总结 42](#_Toc406495476)

[2.3决策理由 43](#_Toc406495477)

[2.4总结与建议 44](#_Toc406495478)

[2.4.1总体性评价 44](#_Toc406495479)

[2.4.2安装环境 44](#_Toc406495480)

[2.4.3经验总结 44](#_Toc406495481)

[3.其他 45](#_Toc406495482)

[3.1文档修订历史 45](#_Toc406495483)

# 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 | 1.2 | | |
| 编写人 | 黄涵倩 | 编写日期 | 2015.04.19 |
| 审批人 | 金翠 | 审批日期 | 2015.04.20 |
| 保密级别 | 灰常高！！ | | |

## 1.2范围

本测试报告为NBA信息查询平台项目的测试报告，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NBA信息查询系统的需求。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.NBA信息查询平台项目系统用例文档

3. NBA信息查询平台项目系统需求规格说明文档

4. NBA信息查询平台项目系统测试用例文档

## 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

NBA信息查询平台项目系统的测试从2015-04-10开始持续到2015-04-19，共持续10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NBA信息查询平台项目系统需求测试用例及回顾)

※系统实现的主要功能

查询球员信息，查询球队信息，查询比赛信息

※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，用自动化集成工具jenkins，在项目的每个阶段都一直在集成。

※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，包括查询球员信息，查询球队信息，查询比赛信息，对业务逻辑层的逻辑进行了单元测试。

②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

4.测试用例表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试用例标识符 | | 测试用例名称 | | 状态 | 测试结果 | | 备注 | |
| 业务测试 | | | | | | | | |
| Testcase001 | | 查询球队的赛季数据 | | 已执行 | 测试通过 | |  | |
| Testcase002 | | 查询球队的场均数据 | | 已执行 | 测试通过 | |  | |
| Testcase003 | | 对球队的赛季数据排序 | | 已执行 | 测试通过 | |  | |
| Testcase004 | | 对球队的场均数据排序 | | 已执行 | 测试通过 | |  | |
| Testcase005 | | 查看某支球队的比赛信息 | | 已执行 | 测试通过 | |  | |
| Testcase006 | | 查看某支球队的球员信息 | | 已执行 | 测试通过 | |  | |
| Testcase007 | | 查看球员的基本信息 | | 已执行 | 测试通过 | |  | |
| Testcase008 | | 按首字母筛选球员 | | 已执行 | 测试通过 | |  | |
| Testcase009 | | 按所属球队筛选球员 | | 已执行 | 测试通过 | |  | |
| Testcase010 | | 查看球员的赛季数据 | | 已执行 | 测试通过 | |  | |
| Testcase011 | | 查看球员的场均数据 | | 已执行 | 测试通过 | |  | |
| Testcase012 | | 对球员的赛季数据排序 | | 已执行 | 测试通过 | |  | |
| Testcase013 | | 对球员的场均数据排序 | | 已执行 | 测试通过 | |  | |
| Testcase014 | | 根据球员的出场时间进行排序 | | 已执行 | 测试通过 | |  | |
| Testcase015 | | 按照位置，分区和排序条件筛选出球员 | | 已执行 | 测试通过 | |  | |
| Testcase016 | | 查看某一日期的比赛信息 | | 已执行 | 测试通过 | |  | |
| Testcase017 | | 按照赛季，日期，主队和客队筛选出比赛 | | 已执行 | 测试通过 | |  | |
| Testcase018 | | 查看今日热点球员 | | 已执行 | 测试通过 | |  | |
| Testcase019 | | 查看赛季热点球员 | | 已执行 | 测试通过 | |  | |
| Testcase020 | | 查看赛季热点球队 | | 已执行 | 测试通过 | |  | |
| Testcase021 | | 查看进步最快球员 | | 已执行 | 测试通过 | |  | |
| Testcase022 | | 根据输入值模糊查找球员 | | 已执行 | 测试通过 | |  | |
| Testcase023 | | 根据输入值模糊查找球队 | | 已执行 | 测试通过 | |  | |
| 功能测试 | | | | | | | | | |
| Testcase024 | 用户输入数值不合法检测 | | 已执行 | | | 测试通过 | |  | |
| Testcase025 | 根据赛季和日期显示比赛信息 | | 已执行 | | | 测试通过 | |
| Testcase026 | 根据赛季显示球队信息 | | 已执行 | | | 测试通过 | |
| Testcase027 | 根据赛季显示球员信息 | | 已执行 | | | 测试通过 | |
| Testcase028 | 球员信息是否按球员名字首字母显示 | | 已执行 | | | 测试通过 | |
| Testcase029 | 球队信息是否按球队名字首字母显示 | | 已执行 | | | 测试通过 | | 账户管理 | |
| Testcase030 | 查询信息时输入空值检测 | | 已执行 | | | 测试通过 | |
| 系统测试部分 | | | | | | | | | |
| Testcase031 | 大数据量测试 | | 已执行 | | | 测试通过 | | 性能测试 | |
| Testcase032 | 负载测试 | | 已执行 | | | 测试通过 | |
| Testcase033 | 压力测试 | | 已执行 | | | 测试通过 | |
| Testcase034 | 调整界面大小、界面还能完整显示，按比例缩放 | | 已执行 | | | 测试通过 | | 界面测试 | |
| Testcase035 | 是否有错误提示 | | 已执行 | | | 测试通过 | |
| Testcase036 | 提示、警告、或错误说明应该清楚、明了、恰当 | | 已执行 | | | 测试通过 | |
| Testcase037 | 所有跳转均到了该跳转的页面 | | 已执行 | | | 测试通过 | |
| Testcase038 | 按钮状态是否正确 | | 已执行 | | | 测试通过 | |
| Testcase039 | 分辨率测试 | | 已执行 | | | 测试通过 | |
| Testcase040 | 平台测试 | | 已执行 | | | 测试通过 | | 兼容性测试 | |

### 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JunitTest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NBA信息查询平台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) |
| bl.player | 1.1 | 查询场均信息时报空指针异常 | 没有考虑到record球员不在那448个球员中的情况 | fail->pass |
| bl.player | 1.2 | 根据球队找球员报空指针异常 | 有的球队在不同赛季中改了名字 | fail->pass |
| bl.match | 1.4 | 得不到相应赛季的比赛信息 | 查询比赛数据时map对象的key值搞混了 | fail->pass |
| ui.playerPanel | 1.4 | 筛选球员时越筛越少 | 没有重置数组列表，在筛选后的列表里继续筛的 | fail->pass |
| ui.matchPanel | 1.5 | 查询某个日期的比赛的时候第二次不显示比赛信息 | 缓存数据 变动后没有重新获取 | fail->pass |
| ui.bestImprovedPlayer | 1.6 | 获得进步最快球员数据错误 | 调用错方法，传错colum值 | fail->pass |
| ui. teamPanel | 1.7 | 显示球队信息时报空指针异常 | 球队名没有进行中英文转换 | fail->pass |
| ui.playerPanel | 1.8 | 根据某条件筛选时报空指针异常 | 未对筛选的中文信息进行英文转换 | fail->pass |
| bl.player | 1.9 | 保留4位小数的时候出现NumberFormatException | 对数据进行计算的时候没有做除0检查 | fail->pass |
| bl.player | 1.10 | 查询球员信息的时候发现有多个值为null | 忘记set球员基本信息了 | fail->pass |
| ui.bestImprovedPlayer | 1.11 | 显示进步最快球员信息数据显示错误 | 没有将从data传来的数据按顺序正确显示到界面上，忘记clear了 | fail->pass |
| ui.matchPanel | 1.12 | 查看某个日期的比赛信息，一行只有一场比赛时比例失调 | GridBagLayout的属性未设置好 | 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 |
| ui.playerPanel | 1.13 | 一般 | 没有进行可模糊的查找 | 底层实现的是equals而不是contains |
|  |  |  |  |  |

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：6

一般缺陷：6

无影响缺陷：0（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，为数据层和逻辑层提供了需实现的接口。在构造阶段，除了对数据文件的读写外，其余全部采用针对接口的编程，两边同时进行，每实现一个功能，就进行maven build。逐步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用UI层的实现代码将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层之后用BL层的实现代码将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。 | fail->pass |
| teamBL  &teamData | 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 |
| PlayerPO | 2.3 | PlayerPO对象，经由rmi通信后部分数据丢失 | Client与server两者的PlayerPO对象，成员名不一致，导致data层数据丢失 | fail->pass |
| playerui | 2.5 | 刷新球员列表时在bl层的球员排序算法中报空指针异常。 | 界面在构建PlayerVO是参数未初始化及正确赋值，bl层未对vo数据再次检测，导致参数中有null的存在 | fail->pass |
| teamUI | 2.6 | 读取team文件时，报空值错误 | Data层对于team文件的解系错误 | 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 |
| PlayerData | 2.6 | 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 |
| playerBL | 2.7 | 无影响缺陷 | 遍历所有的比赛，来查询球员所参加的比赛 | 当比赛较多时，计算时间较长 |
|  |  |  |  |  |
|  |  |  |  |  |

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。



图2.2.4.1 Jenkins运行截图

5.度量数据

1.集成测试缺陷数据：

严重缺陷：4

一般缺陷：2

无影响缺陷：2

2.集成测试覆盖度：

被测试的模块总数：3

模块总数：3

模块覆盖度：100%

备注：模块根据main / java/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，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) |
| MatchData | 3.1 | 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 | fail->pass |
| TeamData | 3.2 | Data层文件路径不对显示空指针异常 | 没有考虑系统使用时数据有可能迁移 | fail->pass |
| PlayerData | 3.3 | 排序前num名球员的时候arrayOutOfBounds | 没有对num为负值的情况进行处理 | 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 |
| 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。

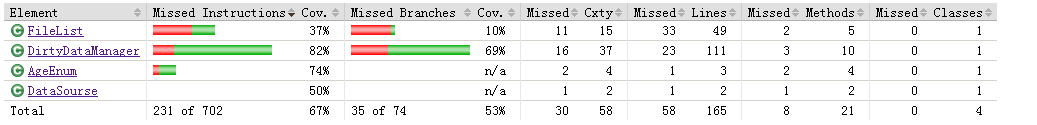
5.度量数据：

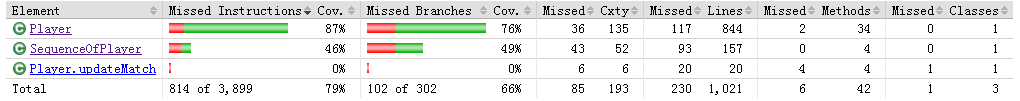
1.缺陷度量：

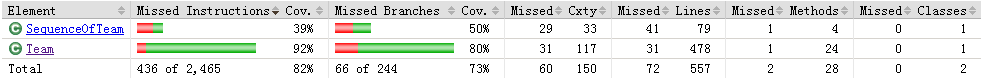
缺陷数目难以找到真实可靠的数据。

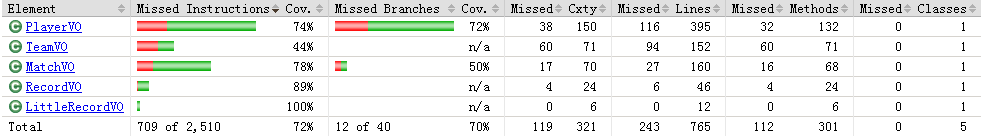
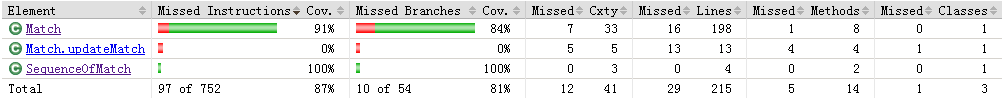
2.覆盖度度量：

使用JAVA EMMA工具进行度量。度量结果如下图示例



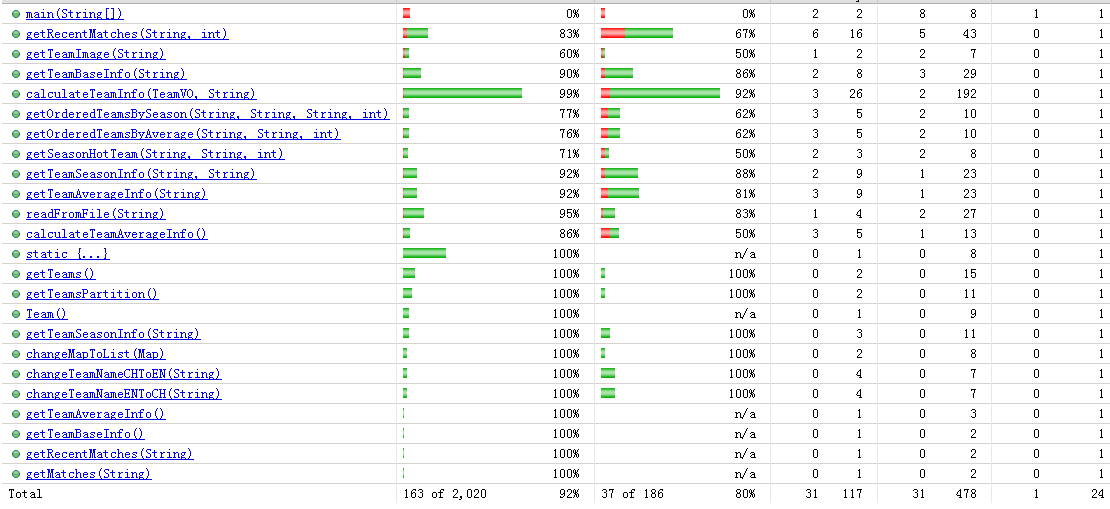












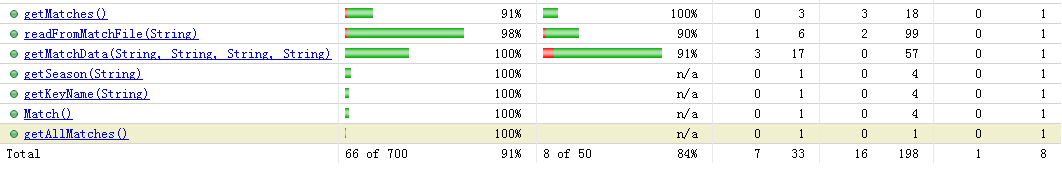


图2.2.3.5 覆盖度度量

（注：度量时Presentation包及businesslogicservice包不计算在内）

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高（79%）： 原因在于将UI层不计入单元测试的覆盖中，因为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NBA信息查询平台系统，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，因为对于篮球迷的所期望看到的数据并不那么了解。

另使用文件系统对数据的获取和分析有利，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。

### 2.4.3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。
7. 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 | 2015-04-19 |
| 1.1 | 严顺宽 | 完善部分缺陷分析记录 | 2015-04-20 |
| 1.2 | 金翠 | 文档审核 | 2015-04-20 |