|  |
| --- |
| 画扇小组 |
| 测试计划 |
| 迭代一 |

|  |
| --- |
| 张仁知、阮威威  2015-3-11 |

目录

[单元测试计划 0](#_Toc445489852)

[单元测试用例 1](#_Toc445489853)

[集成测试计划 3](#_Toc445489854)

[迭代一功能测试 4](#_Toc445489855)

# 单元测试计划

**团队名称**：画扇小组

被测系统名称：GitMining

1. 单元测试计划概述

本文档为在GitMining系统进行单元测试时提供的有关任务安排、方法、资源和进度方面的指导，目标是从代码中发现Bug，以备后续移除Bug参考，提高软件质量，最后达到用户需求。

本文档的读者主要是开发工程师。

参考文献：《需求规格说明文档》和《Gitmining软件详细设计描述文档》

1. 单元测试的范围

单元测试必须能够展示GitMining系统中类的每个方法都能正常工作。单元测试的概念是将一个单元和该系统的其余部分隔离开来单独来独立测试，因此可能需要创建测试驱动（test driver）代码。要求测试驱动代码用JUnit编写，当被测单元被提交准备完成项目构建时，相关的测试驱动代码必须已经被提交到团队的代码库中。在第三部分将给出隔离测试代码的方法指导。

注意：JUnit不能进行GUI测试。

1. 单元测试的策略
2. 对方法（method）的单元测试

·检查对所有调用对象的使用。

·验证对所有数据结构的处理。

·验证对所有文件的处理。

·验证控制流的不变性。

·检查所有循环的正常终止。

·检查所有循环的异常终止。

·验证所有错误条件的处理。

·检查定时和同步。

1. 对类（class）的单元测试
2. 结合方法的执行，推荐对类的测试。

·选择最常见的执行序列。

·需要包括可能导致缺陷的序列。

·手工计算预期的属性值。

1. 集中对每个属性进行单元测试。

·初始化，然后执行会影响属性的方法序列。

1. 验证每个类的不变性。

·验证默认初始值时的不变性。

·执行方法序列。

·验证不变性仍然正确。

# 单元测试用例

**项目名称**：GitMining **团队名称**：LifeIS小组

Unit Test Case L1

被测单元名称：RepositoryService

用例设计人员：阮威威 设计日期：

执行测试人员：张仁知 测试日期：

测试用例列表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例ID | 测试用例描述 | 输入数据 | 预期输出 | 实际输出 | Pass/Fail | 备注 |
| TestRepositoryService-001 | GetAllRepository |  | List<ProjectInfo> | List<ProjectInfo> | Pass |  |
| TestRepositoryService-002 | getRepositoryByNam1 | Null:无效输入 | Throw Error | Throw Error | Pass |  |
| TestRepositoryService-003 | getRepositoryByNam1 | ProjectName=有效输入 | ProjectDetail | ProjectDetail | Pass |  |
| TestRepositoryService－004 | searchRepositorys | String key：“11” | List<ProjectInfo> | List<ProjectInfo> | Pass |  |
| TestRepositoryService－005 | searchRepositorys | String key：“1” | List<ProjectInfo> | Throw Error | Fail | 逻辑出错 |
| TestRepositoryService－006 | searchRepositorys | String key：有效输入 | List<ProjectInfo> | List<ProjectInfo> | Pass | 逻辑纠正 |
| TestRepositoryService－007 | getSortedRepositorys | SortType:Fork | List<ProjectInfo> | List<ProjectInfo> | pass |  |
| TestRepositoryService－008 | getSortedRepositorys | SortType:Star | List<ProjectInfo> | List<ProjectInfo> | pass |  |
| TestRepositoryService－009 | getSortedRepositorys | SortType:Contributor | List<ProjectInfo> | List<ProjectInfo> | pass |  |
| TestRepositoryService－010 | getSortedRepositorys | SortType:General | List<ProjectInfo> | List<ProjectInfo> | pass |  |
| TestRepositoryService－011 | SortSearchRepositorys | SortType:Fork  null:无效输入 | Throw error | Throw error | Pass |  |
| TestRepositoryService－012 | SortSearchRepositorys | SortType:Fork  ProjectName=有效输入 | List<ProjectInfo> | List<ProjectInfo> | Pass |  |
| TestRepositoryService－013 | SortSearchRepositorys | SortType:Star  ProjectName=有效输入 | List<ProjectInfo> | List<ProjectInfo> | Pass |  |
| TestRepositoryService－014 | SortSearchRepositorys | SortType:Contributor  ProjectName=有效输入 | List<ProjectInfo> | List<ProjectInfo> | Pass |  |
| TestRepositoryService－015 | SortSearchRepositorys | SortType:General  ProjectName=有效输入 | List<ProjectInfo> | List<ProjectInfo> | Pass |  |

Unit Test Case L2

被测单元名称：UserService

用例设计人员：阮威威 设计日期：

执行测试人员：张仁知 测试日期：

测试用例列表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例ID | 测试用例描述 | 输入数据 | 预期输出 | 实际输出 | Pass/Fail | 备注 |
| TestUserService-001 | GetAllUsers |  | List<UserInfo> | List<UserInfo> | Pass |  |
| TestUserService-002 | getUserByName | String:null=无效输入 | Throw Error | Throw Error | Pass |  |
| TestUserService-003 | getUserByName | String name:有效输入 | UserInfoDetail | UserInfoDetail | pass |  |
| TestUserService-004 | searchUsers | String:null | List<UserInfo>为空 | List<UserInfo>为空 | pass |  |
| TestUserService-005 | searchUsers | String;“asdfs” | List<UserInfo> | List<UserInfo> | pass |  |
| TestUserService-006 | searchUsers | String;“11” | List<UserInfo> | List<UserInfo> | pass |  |
| TestUserService-007 | searchUsers | String;“1” | List<UserInfo> | List<UserInfo> | pass |  |

# 集成测试计划

**团队名称：**LifeIS小组

**被测系统名称：**GitMining

1. 集成测试计划概述

本文档主要描述GitMining系统的集成测试活动如何进行。包括如何控制集成测试活动、流程以及工作安排等。

本文档只计划针对软件的集成测试，不包括硬件、系统测试和单元测试内容。主要的读者是LifeIS小组内部成员。集成测试的目标是验证系统和《软件架构设计文档》的吻合情况。

1. 集成测试的范围和资源
2. 集成测试的主要任务

在连接各个模块的时候，测试穿越模块接口的数据是否会丢失。

在组合各个模块的子功能后，测试是否能到达预期的父功能。

测试一个模块的功能是否会对另一个模块的功能产生不利影响。

测试单个模块的误差积累是否会放大到不可接受的程度。

1. 集成测试的资源

集成测试人员为4人，分工如下：

阮威威：软件质量保障员（集成测试负责人）

肖安祥：技术经理（架构的团队内部负责人）

张云飞、张仁知：软件工程师

测试执行时间为3个工作日，在完成单元测试的第二天开始。

1. 集成测试环境

Jerkins , eclipse直接运行。

1. 集成测试计划资源分配

阮威威全面控制并负责集成测试的组织，书写测试用例，并负责测试环境的搭建，同时决定提交上来的Bug是否需要修改。

肖安祥负责架构一致性和Bug的移除确认。

张云飞负责按照集成测试用例表进行测试活动，并记录测试结果。

张仁知负责按照集成测试用例表记性测试活动，并记录测试结果。

1. 待测试的清单和顺序
2. 子系统测试顺序

系统划分为项目搜索、项目排序和用户查询子系统。

主要的集成顺序是自底向上。

按照“项目搜索——项目排序——用户查询”的顺序进行集成和测试。

1. 子系统的测试站滚沸

集成子系统，测试相关的UI、业务逻辑和数据单元是否达到子系统的功能要求。

数据传递是否正确，对于传入值的控制范围是否一致。

类之间的调用是否正常。

执行集成测试的初步计划如下。

Day1

阮威威：构建测试环境，并组织测试的组织。

张仁知：执行项目搜索模块内部的集成和测试，并记录结果，反馈Bug。

张云飞：执行项目排序模块内部的集成和测试，并记录结果，反馈Bug。

肖安祥：对与架构一致性进行确认，反馈开发工程师调试。

Day2

阮威威：执行项目搜索和项目排序模块之间的集成和测试，并记录测试结果，反馈Bug。

张云飞：执行用户查询模块内部的集成和测试，并记录测试结果，反馈Bug。

张仁知：执行项目搜索和项目排序模块之间的集成和测试，并记录测试结果，反馈Bug。

肖安祥：对系统测试需求一致性和通过性进行确认。

1. 集成测试结束交付产物
2. 集成测试用例文档的执行情况列表（所有成员填写，阮威威汇总整理）
3. Bug报告表（所有成员填写，阮威威汇总整理）
4. 集成测试报告（张仁知负责完成）
5. 执行集成测试的入口和出口条件
6. 入口条件

单元测试必须完成。

开发出完整的系统。

搭建了类似于实际使用/配置环境的测试环境。

测试人员安排到位。

1. 出口条件（以下条件达到其一就可以进入系统测试）

所有的集成测试用例都被执行，测试结果100%通过。

集成测试过程中发现的Bug已经被定位或者关闭。

1. 如何判断集成测试用例通过
2. 接口集成时，接口提供的功能或者数据正确。
3. 功能点集成时，验证与《软件架构设计文档》中的描述一致。
4. 如何判断集成测试用例失败以及失败后的行为

* 接口集成时，接口提供的功能或者数据不正确。
* 功能点集成时，验证与《软件架构设计文档》中对需求的描述不一致。
* 集成测试用例未通过，提交测试负责人，由测试负责人阮威威判断是否需要转交技术经理肖安祥，定位和移除Bug。

# 迭代一功能测试

Repository的测试用例

|  |  |  |
| --- | --- | --- |
| ID： | 输入 | 预期输出 |
| TSU1－1 | 选择Fork排序 | 呈现Fork排序后的项目信息列表 |
| TSU1－1.1 | 选择Star排序 | 呈现Star排序后的项目信息列表 |
| TSU1－1.2 | 选择Contributor排序 | 呈现Contributor排序后的项目信息列表 |
| TSU1－1.3 | 选择General | 呈现最初的项目信息列表 |
| TSU1－2 | 搜索栏输入“1” | 产生所有相关项目信息列表 |
| TSU1－2.1 | 选择Fork排序 | 呈现Fork排序后的搜索项目信息列表 |
| TSU1－2.2 | 选择Star排序 | 呈现Star排序后的搜索项目信息列表 |
| TSU1－2.3 | 选择Contributor排序 | 呈现Contributor排序后的搜索项目信息列表 |
| TSU1－2.4 | 选择General | 呈现最初搜索项目信息列表 |
| TSU1－3 | 搜索栏输入“11” | 产生所有相关项目信息列表 |
| TSU1－3.1 | 选择Fork排序 | 呈现Fork排序后的搜索项目信息列表 |
| TSU1－3.2 | 选择Star排序 | 呈现Star排序后的搜索项目信息列表 |
| TSU1－3.3 | 选择Contributor排序 | 呈现Contributor排序后的搜索项目信息列表 |
| TSU1－3.4 | 选择General | 呈现最初搜索项目信息列表 |
| **TSU1－4** | 搜索栏输入“a” | 产生所有相关项目信息列表 |
| TSU1－4.1 | 选择Fork排序 | 呈现Fork排序后的搜索项目信息列表 |
| TSU1－4.2 | 选择Star排序 | 呈现Star排序后的搜索项目信息列表 |
| TSU1－4.3 | 选择Contributor排序 | 呈现Contributor排序后的搜索项目信息列表 |
| TSU1－4.4 | 选择General | 呈现最初搜索项目信息列表 |
| TSU1－5 | 搜索栏输入“ab” | 产生所有相关项目信息列表 |
| TSU1－5.1 | 选择Fork排序 | 呈现Fork排序后的搜索项目信息列表 |
| TSU1－5.2 | 选择Star排序 | 呈现Star排序后的搜索项目信息列表 |
| TSU1－5.3 | 选择Contributor排序 | 呈现Contributor排序后的搜索项目信息列表 |
| TSU1－5.4 | 选择General | 呈现最初搜索项目信息列表 |
| TSU1－6 | 选择一个项目，并点击 | 呈现出项目的详细信息 |
| TSU1－6.1 | 选择一个项目相关用户选项并点击 | 呈现出相关信息 |
| TSU1－7 | 选择Home | 呈现初始界面 |
| TSU1－8 | 选择刷新 | 信息进行刷新 |

User的测试用例

|  |  |  |
| --- | --- | --- |
| ID： | 输入 | 预期输出 |
| TSU2－1 | 搜索栏输入“1” | 产生所有相关用户信息列表 |
| TSU2－2 | 搜索栏输入“1” | 产生所有相关用户信息列表 |
| TSU2－3 | 搜索栏输入“a” | 产生所有相关用户信息列表 |
| TSU2－4 | 搜索栏输入“ad” | 产生所有相关用户信息列表 |
| TSU2－5 | 选择一个用户，并点击 | 呈现出用户的详细信息 |
| TSU3－5.1 | 选择一个用户参加的项目并点击 | 呈现出相关信息 |
| TSU3－5.2 | 选择一个用户创建的项目并点击 | 呈现出相关信息 |