N-Gitea

基于Gitea的通知扩展

**综合实验总结与分析**

北京航空航天大学计算机学院

二〇二〇年四月十五日

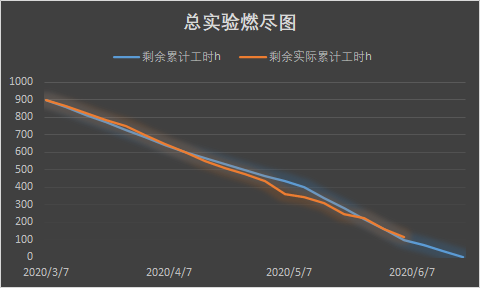
## 一 概述

本项目是基于Gitea的通知模块扩展，是在Gitea原有框架的基础上，进行通知方式、通知内容的扩充。一方面，我们实现了微信通知模块，允许使用者通过绑定微信服务号，实现信息收取和链接跳转。另一方面，我们提供了其他通知模块扩展的接口，允许部署者进一步的扩充通知方式和通知内容。

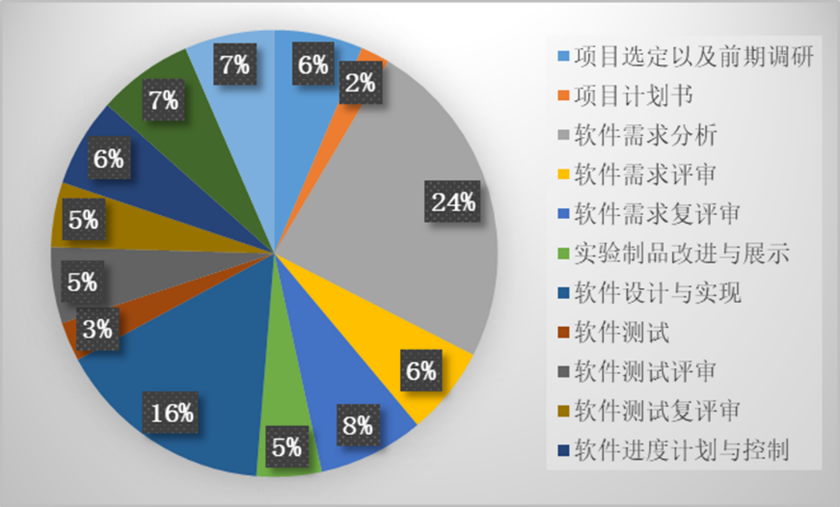
我们依次完成了项目计划、软件需求分析、软件需求评审、软件设计与实现、软件测试、软件测试与评审、软件进度计划、配置管理、工作量估计与分析等多项软件开发流程中的环节。在此期间，我们遇上了许多困难，各环节的制品也有许多不足，但是在老师和其他同学的指导下，克服了这些困难，也改进了我们的制品，对这些环节也有了更进一步的了解。

## 二 总体数据统计

项目总体进度：



项目总工时：



项目总Commit次数：总Commit次数202次，有效提交次数175次，代码类提交102次，文档类提交总数73次。

## 三 各实验情况

### 3.1 实验一：软件需求分析

#### 3.1.1 总体数据

阶段工时：194工时

阶段Commit次数：3.21到4.3期间，总体为46次，文档类21次，代码类24次。

项目制品——需求分析文档：最终定稿字数为17324个字，5个图以及11个表。修订版本数为7次，最终版本号为v1.7。图主要以用例图为主，表主要以RUCM模型为主。

#### 3.1.2 文档质量分析

需求分析说明书经历了多次修改，包括组内修改、根据老师指导意见以及需求初评审、需求复评审的修改，已经得到了较好的完善。

而在撰写需求分析说明书的过程中，使用了用例图、RUCM模型等方式，对需求进行了详细的说明。同时，作为一个对框架进行拓展开发的项目，我们不仅作出了对拓展模块的需求分析，也对原框架的功能需求进行了分析。使得整个需求分析更为合理、完整。

当然，作为项目的开头阶段，其文档肯定有许多考虑不足之处，在后续的实验环节当中，我们也逐渐发现了需求说明书的不合理、考虑不完善的地方。

#### 3.1.3 有效方法

（1）需求分析要从使用者角度分析需求，区分设计文档，设计实现分析是从系统实现角度看待问题的。

（2）使用纯文字描述功能需求会导致难以理解和表述不清，使用用例图、RUCM图，会令整个需求分析更加明确、展示更加清晰、更有条理。

（3）UML模型是开发流程描述的利器，可以根据实际撰写情况，使用不同的模型进行描述。

#### 3.1.4 小结

在需求文档撰写中，首先要确定需求分析的角度。确定角度之后，就要确定分析的模块。然后绘制用例图，区分功能需求和非功能需求，逐一描述。描述完成之后，使用RUCM模型进行辅助描述，使得需求分析更加完善。

### 3.2 实验二：需求评审

#### 3.2.1 总体数据

阶段工时：114工时

阶段Commit次数：总提交次数22次，代码类提交13次，文档类提交次数9次。

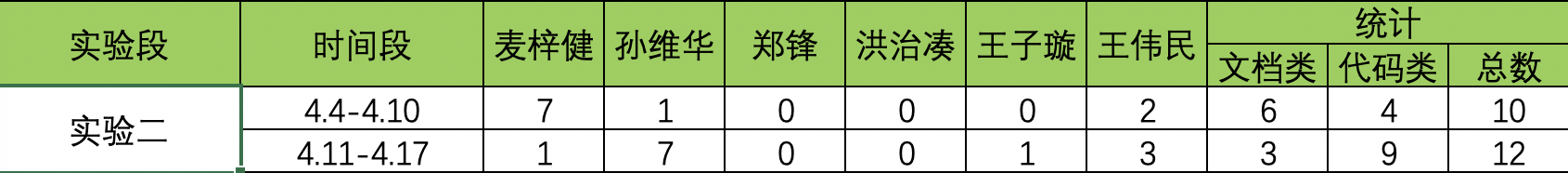


图 1 实验二阶段Commit

收到意见总数：初评审43条（轻微30一般11严重2），接受了其中37条，拒绝了其中六条。复评审46条（轻微33条 一般11条 严重2），接受了其中39条，部分接受3条，拒绝了4条。即总共89条意见，接受了76条，部分接受3条，拒绝了10条。

#### 3.2.2 有效方法

（1）辅助评审：仅仅对问题的描述是不足以完全呈现这个问题的，需要更多的标签辅助我们对问题进行完整的叙述，如“问题位置”、“问题严重程度”、“问题分类”等。这些标签，一方面可以让我们对问题进行分类、组内分工解决，另一方面还允许我们根据问题的优先度进行问题的解决排序。

（2）规范模版：每个评审组的风格和侧重点是不一致的，如果不提供统一的评审模版给到评审组，会导致收到的评审报告表单各不相同，这样对意见的汇总、分类、修改都会带来影响。需要根据自己组的需求，制定一个完善、合理的评审表单模版。

（3）评审交流：指定联系人进行转达，往往比将评审组和被评审组所有人拉进群里进行交流，更有效率。一方面，指定的联系人只要对文档有所了解，就可以针对问题进行快速责任人定位，指定其修改。另一方面，群交流往往会因为相互不熟悉等原因导致交流效率低下。

#### 3.2.3 小结

在初评审和复评审当中，我们分别对四个组进行了评审。在这个过程当中，我们更加意识到，除了字体、格式、修辞等问题外，更应当为评审组找到需求分析的缺漏之处，逻辑不完善的地方。比如，用例图与RUCM图例描述不对应，需求不合理，上下文对同一用户群体的指代不统一等等，这些才更能够给他组完善文档的机会。

同时，还有四个组对我组的文档进行了评审，给我们组提出了许多细致准确的意见，在此再次感谢四个组的评审意见。我们针对这些意见，一一进行了讨论，对接受的意见进行了修改，对不接受的意见也给出了我们的理由，并与评审组进行了交流。

在整个评审工作中，我们不仅对整个流程有了更加清晰的认知，也对自己组的需求分析有了更深刻的了解。

### 3.3 实验三：软件设计与实现

#### 3.3.1 总体数据

阶段工时：128工时

阶段Commit次数：总计6次提交，代码类提交4次，文档类提交2次。



图 2 实验三Commit

实验制品：

（1）概要设计，定稿4516个字，图4个，表5个。定稿版本v0.2。概要设计涉及总体描述、接口设计、数据结构设计、异常处理设计等四个模块。

（2）项目源码：代码总行数1953

#### 3.3.2 Gitea拓展说明

Gitea是一个go的代码自托管平台，其主要的通知方式只有网页通知与邮件通知。但在实际使用当中，这样的通知方式是不够的。因此，我们针对Gitea的通知模块进行了拓展。同时，go语言对于动态加载模块的支持并不完善，而且缺乏跨平台特性的支持，为了更好的进行开发工作，功能扩展使用Python完成模块的加载、运行。Python代码向Gitea提供gRPC远程调用接口，Gitea通过调用Python代码完成通知的设置和分发功能。

#### 3.3.3 总结

我们对功能扩展模块的实现分成了四个部分：数据存储管理、模块加载管理、URL转发控制、Go前端控制，这四个模块各有其设计和实现的难点，分别由开发组的四位同学负责完成。开发的整个过程可以分为总体架构设计、技术选取和验证、模块接口设计、模块实现、总体功能验证这五个阶段。软件设计与实现部分我们总体上完成了需求规格说明书中要求的功能需求和非功能需求，并且修复了软件测试过程中发现的软件问题。

实验三我们做得比较好的地方在于，一开始就明确了分工，因此较早地进入了开发准备阶段，也提高了开发阶段的效率。

本次试验做得不够好的地方有，开发人员配合不密切，因此在对不同开发人员编写的代码的整合过程中就遇到过一些问题，如何提高协作效率是我们本次项目和今后工作中仍需不断思考和探索的方面。

### 3.4 实验四：软件测试

#### 3.4.1 总体数据

阶段工时：23工时

阶段Commit次数：总计Commit34次，文档类两次，代码类32次

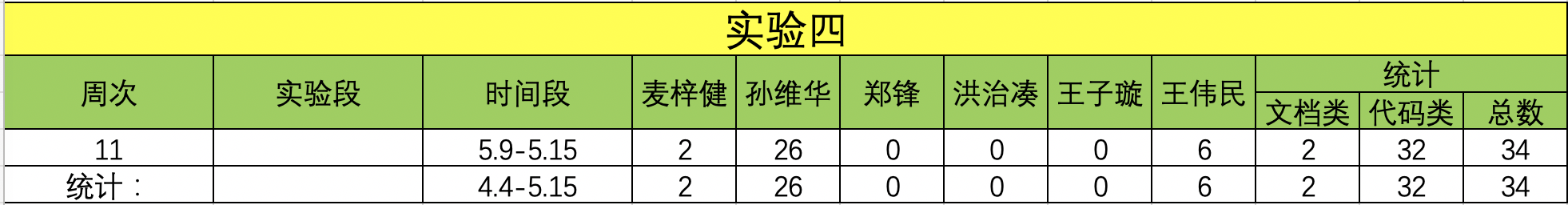


图 3 实验4Commit

项目制品：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 描述 | 字数 | 图 | 表 | 定稿版本 |
| 测试需求规格说明书 | 用于阐述测试工作流程和明确测试阶段工作 | 12683 | 0 | 27 | v1.9 |
| 单元测试报告 | 用于说明单元测试工作的工具、流程和结果 | 2462 | 10 | 0 | v1.2 |
| 测试报告 | 用于说明测试用例的自测结果 | 8263 | 0 | 22 | v1.1 |
| 缺陷报告 | 用于详细说明测试缺陷定义和处理流程 | 1244 | 2 | 0 | v1.1 |

#### 3.4.2 文档分析

在软件自测当中，我们撰写比较多的临时文档，然后最终的制品为上述四个文档。

测试需求规格说明书是本实验的主要制品，其用于阐述测试工作流程和明确测试阶段的详细工作。其定义了对Gitea的原功能及拓展通知功能模块进行各种测试时的工作流程和输入输出用例的形式、内容。在初评审和复评审后，我们又根据评审意见，增加了实验一需求规格说明书和测试需求说明书中需求的对应表，以达成前后相对应。同时，根据评审意见，我们对测试用例进行了小规模的补充、增删和完善，使测试文档逻辑自洽，上下文对应，模块描述清晰、完整。

单元测试报告，是软件自测的一种形式。因为整个项目使用了两种开发语言Go和Python，所以我们需要针对两种语言，选择不同的单元测试框架。在报告中我们具体的给出了测试的工具说明、使用方式和使用结果。然后，根据初评审意见和复评审意见，我们又对单元测试报告进行了一些修改，使得报告更为完善。

测试报告，准确来说是测试用例报告，是根据初评审意见，从测试需求说明书中分离出来的报告。将设计的测试用例与实际的测试结果分离，使得说明书更加清晰、合理，同时整个实验阶段的文档制品更加完善。

缺陷报告则是用来辅助自测的，通过对需求规格说明书的分析，我们对自己项目的软件缺陷进行了定义，同时根据自己组的情况，对软件缺陷的处理流程进行了定义，这样在自测、初评审和复评审工作当中，可以根据这份报告进行缺陷处理。

#### 3.4.3 有效方法

（1）单元测试是软件自测的一个重要手段，是对一个模块、一个函数或者类进行输入输出的正确性校验的工作，是最小的测试模块。软件自测需要单元测试保证大部分模块本身工作的正确性，再进而测试模块之间工作的正确性。单元测试尽可能做到完整，但是全部覆盖又是难以实现的，需要根据项目情况进行把控。

（2）在设计测试用例过程中，测试方法是一个重要的根据。不同的测试方法会对测试内容有不同程度的影响，白/黑盒测试、场景测试、集成测试等等，测试方法不一定要全，但是需要意识到，测试不同的内容时，有不同的测试方法。

（3）不是所有的内容都需要测试的，这是需要认识到的一点。一个项目从开发到部署再到使用过程，可以测试的东西太多了，在时间限制内，我们需要根据测试需求对重要的地方进行测试。在本次实验当中，我们的开发模块主要是动态加载模块和微信通知模块，以及其下的数据库管理模块、配置管理模块等等。因此，我们将会主要围绕这些模块，从部署到使用设计测试用例。

#### 3.4.4 小结

软件自测是保障软件质量的重要组成部分。在最开始的时候，我们仅仅设计了14个测试用例，而且大多是针对部署的细节进行了测试。在展示和初评审、复评审之后，我们删去了部分过于细节的用例，对部分描述过于简单的测试用例进行了完善，同时增加了可视化部分的测试用例，以实际测试模块功能。而对发现的软件缺陷，我们也对其进行了部分修复。有部分问题由于是原框架的问题，我们的需求说明书和代码都没有涉及，同时也考虑到时间和成本的问题，因而决定不予修复。

软件是很难做到尽善尽美的，在开发过程中，我们认为自己的逻辑是自洽的，因而需要他人的评审工作来发现问题。我们需要与评审组保持良好的沟通，以完成测试过程的问题发现和修复。

### 3.5 实验五：软件测试评审

#### 3.5.1 总体数据

阶段工时：82工时

阶段Commit次数：总计26次，代码类提交18次，文档类8次



图 4 实验五Commit

收到的意见情况：总计意见85条，轻微55条，中等29条，严重1条。接受情况总计接受81条，部分接受2条，拒绝2条。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 收到意见 | 接受情况 | 备注 |
| B组文档评审单 | 12（轻微9 中等3） | 接受10 拒绝2 |  |
| B组软件报告 | 2（轻微1 中等1） | 接受2 |  |
| C组文档评审单 | 12（轻微11 中等1） | 接受12 |  |
| C组软件报告 | 2（中等2） | 拒绝2 |  |
| I组文档评审单 | 23（轻微17 中等6） | 接受22 部分接受1 |  |
| I组软件报告 | 6（轻微1 中等5） | 接受6 |  |
| A组文档评审单 | 28（轻微16中等11严重1） | 接受27部分接受1 |  |

#### 3.5.2 过程分析

在整个对他组测试评审过程当中，我们发现，由于项目的种类繁多，而在短短的时间内想完全熟悉他组的软件是不可能的，甚至想要全面了解都很困难，因此可视化的UI交互界面就显得非常重要。良好的UI界面可以将自己的工作完整的展示给评审组，帮助评审组直观的了解项目的功能，更好地进行评审工作。而不友好的界面或者说没有界面，可能会导致评审工作难以开展，举步维艰。

另一方面，在对测试需求文档的评审过程当中我们发现，无论是他组对本组评审还是本组对他组评审，主要都是从测试用例的完整性、准确性入手，由于对项目了解不够，很难从正确性进行评价，也就是这个测试用例是否恰当，是否应当存在。这方面可能主要就是时间不够，导致对项目理解不足引起的其中一个问题。

#### 3.5.3 有效方法

（1）配置好环境，设计项目的UI界面，提供输入输出的直观表达，是评审工作顺利进行的重要助力。

（2）沟通交流是比看文档直接测试的更有力手段。动辄数万字的文档，全新的源码项目，在几天内一点一点看、一点一点自行理解是完全不现实的，在这种情况下，通过沟通，对不理解的地方马上进行交流，一方面可以加快对评进程，另一方面也可以加深对评审项目的了解。

#### 3.5.4 总结

软件测试评审也是软件质量保障的重要组成部分。在软件自测之后，进行相互的软件评审，通过对其他组的详细评审，横向对比，我们可以了解我们测试手段的不足、测试流程的不完善。而通过他组对我们组的评审，我们又可以通过他组的视角，从新审视自己的测试流程、测试结果。不断地与老师、与同学们交流沟通，我们不断发现自己团队的问题和疏忽，不断改进自己的报告和测试。

### 3.6 实验六：项目进度计划

#### 3.6.1 总体数据

总工时：56工时

项目制品：

（1）个人工作日志：2750字，更新次数14次

（2）项目进度计划：计划任务个数128个，更新次数15次

（3）进度计划与控制分析报告：1770字，1图表

#### 3.6.2 数据分析

我们对影响实验实施进度原因总结如下：

1) 小组成员自身因素：在软件工程项目过程中，人永远是最重要的。小组成员在项目过程中有时会存在这样那样的问题，比如个人突发情况导致时间被占用、情绪低落无法全身心投入到项目中、配合不力等，为了规避这一点，我们会尽量合理安排任务分工，提早沟通与督促，起到提醒的作用，并提高小组成员的工作积极性。

2) 甲方需求的变更：在本次实验中，扮演甲方角色的是任课老师和负责评审的其他组的同学们。在实验的进行中，老师会根据课程要求和进度安排会进行一些细小的任务调整，评审组也会根据他们的理解对我方制品提出改进的点。

我们一方面会有前面所提到的每周定期会议，另一方面会在微信群里及时沟通，根据这些变化适当调整，比较灵活，更好地应对实施过程中的影响因素。

#### 3.6.3 总结

制定项目计划并进行相应的进度控制，不仅仅对小组成员的时间进行了合理高效的利用，还增强项目进度的透明度，当项目进展与项目计划出现严重偏差时，可以进一步采取适当的纠正或预防措施，从而保障项目如期完成。

### 3.7 实验七：项目配置管理

#### 3.7.1 总体数据

总工时：63工时

项目制品：

（1）Commit总表：总字数约为200左右，图12张，表6张。

（2）项目配置管理报告：总字数2651字，图2张，表7张。定稿版本v2.7。

#### 3.7.2 数据分析



图 5 Commit统计总表

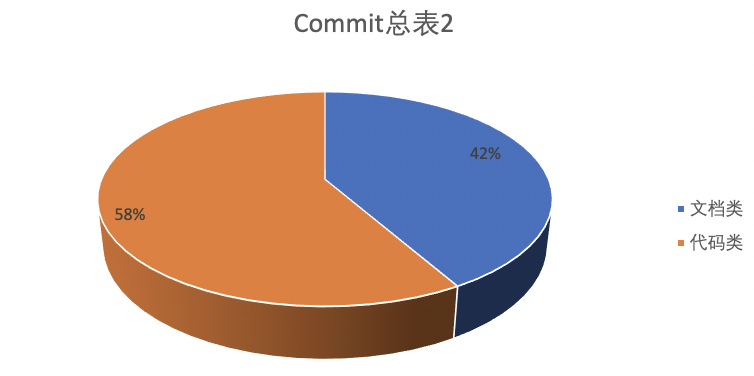


图 6 Commit占比总图

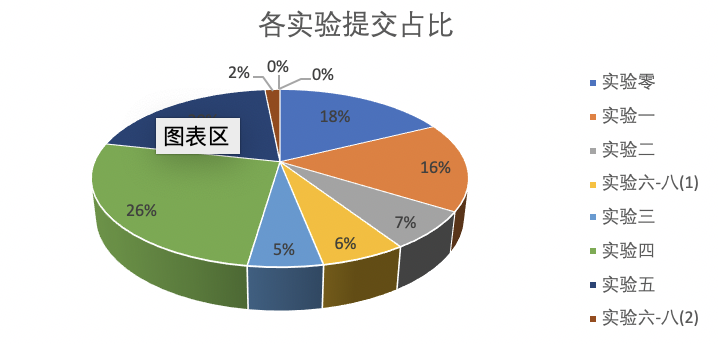


图 7 各实验提交占比

先看下Commit的总体数据，从3.7-6.5，一共提交次数202次（git shortlog | wc -l统计得到），有效提交次数175次（通过命令git log --oneline --author="asd864613087" --since="" --until="" | wc -l统计得到）。可以看到，代码类的总有效提交次数102次，文档类提交总有效提交次数73次，各自占比58%和42%。主要的提交时间段在实验四、实验五和实验一的时间段。但实际上，因为初期的分工已经决定了是代码工作和文档工作双线并行，代码的工作贯穿了整个实验，从实验一到实验五都有代码类的提交，所以这个提交占比只能用作参考。

#### 3.7.3 配置管理分析

配置管理报告主要又两方面的作用，一是记录各种主要文件的提交记录：会议记录、课堂汇报、交流记录以及主要实验制品。二是说明各种配置管理手段，如github的管理方式、Commit的统计方式、文件命名方式等等。

#### 3.7.4 有效方法

（1）github的Commit规范：使用commit的模版，规定Commit必须是以类型+Commit简述+逐条修改阐述这样的形式进行提交，方便后期的查询和管理

（2）github的Tag规范：使用tag进行重要版本管理，对Master分支的提交进行打tag，tag名以重要制品+版本来命名，方便后期的查询，以及当出现重大错误的时候，进行回退操作能够分辨重要的提交时间节点。

（3）文件的命名规范：对文件命名进行规范可以方便后期的管理，以主版本号+次版本号+修订版本号，可以对修改的内容进行简单的区分，方便地找到所需的文件版本。

（4）git log的使用：强大git log的命令，有着丰富的参数，可以对git log进行信息的筛选。常用的git log –graph –all –online 可以进行所有分支的查看，对分支进行管理。对commit进行统计时，可以使用git log –pretty=“”自行定义输出形式，使用—author，--since，--until来进行提交者、时间的筛选。

#### 3.7.5 总结

配置管理是项目文件的总体管理，对过去文件提交的总结，是对现有文件的管理，是对未来提交的规范。一开始我们的配置管理，仅限于统计一些文件的提交，后来发现这样是不够的，对于现有的文件，我们也需要进行一些改变，所以我们进行了tag管理。再进一步的，在与老师和同学们的交流中，我们意识到，还必须对文件管理进行更多的规范，所以我们对未来提交的文件和commit注解进行了规定。

### 3.8 实验八：工作量估计与分析

#### 3.8.1 总体数据

工时：55工时

项目制品：

工作量追踪与统计分析报告：总字数约5200字，图表个数19个

#### 3.8.2 数据分析

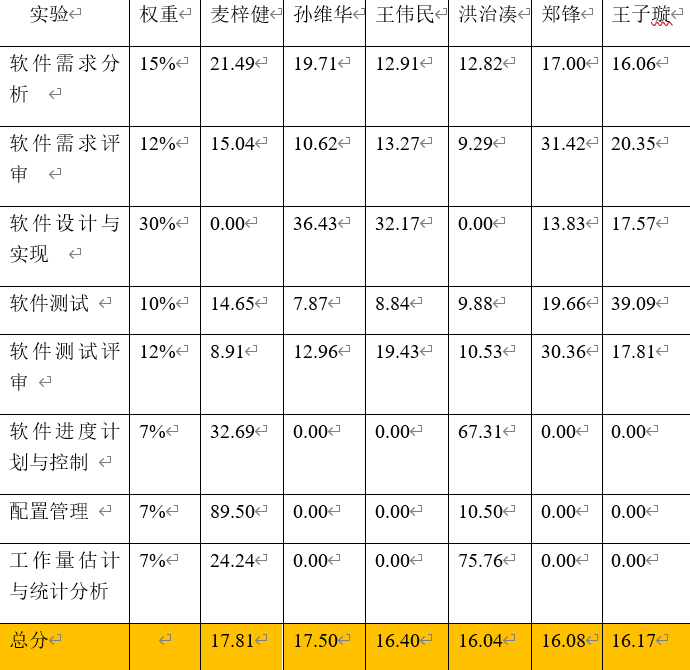


图 8 小组组员贡献

归功于前期根据组员擅长技能的不同进行了比较合理的任务分解与分配，明确了项目组成员职责分工，并采取一些有效措施保证和提高项目组成员间的有效协作，可以看到，虽然在某个具体实验组员间的评分可能有少许差异，但加权总分各成员间都相差无几，不存在特别大的差异。

#### 3.8.3有效方法

明确项目组成员职责分工这一点至关重要，因此，在小组成立初期就应该根据每个成员所擅长的技能经行合理的任务分解与分配，要做到这一点，就必须在前期最大程度的收集组员技能优劣势信息,在这个环节中，不能够谦虚或者夸大自己，应该实事求是。

#### 3.8.4 总结

从组员贡献度总表中可以很容易看出小组成员工作量情况。小组组长麦梓健的工作重点在配置管理和协调组员工作和文档撰写上；小组成员孙维华、王伟民的工作重点在软件设计与实现上；小组成员洪治凑的工作重点在项目进度控制和统计分析上；小组成员郑锋的工作重点在软件需求评审、软件测试评审上；小组成员王子璇的工作重点在文档测试和协助组长撰写文档上。这与小组在项目初期确定的成员职责分工大致相符，并且经过这一学期的良好实践，也说明这是一个比较好的分工。