|  |
| --- |
| 软件工程综合实验H组 |
| 进度控制分析报告v1.1 |
| 基于Scrapy的模板化爬虫程序管理平台 |

|  |
| --- |
| 宋冰晨  2020-6-10 |

**版本变更历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 日期 | 修改人 | 修改章节 | 备注 |
| 1.0 | 2020/6/1 | 宋冰晨 | 全部 | 建立初稿 |
| 1.1 | 2020/6/10 | 宋冰晨 | 表3.1 | 补充部分组员的工作 |

**目录**

[1. 引言 - 1 -](#_Toc41938624)

[1.1 编写目的 - 1 -](#_Toc41938625)

[1.2 项目概述 - 1 -](#_Toc41938626)

[1.3 项目进度控制流程 - 1 -](#_Toc41938627)

[1.4 文档组织结构 - 2 -](#_Toc41938628)

[1.5 数据说明 - 2 -](#_Toc41938629)

[2. 计划变更及其影响因素分析 - 3 -](#_Toc41938630)

[2.1 计划变更情况 - 3 -](#_Toc41938631)

[2.2 影响因素分析 - 4 -](#_Toc41938632)

[3. 工作分配及其影响因素分析 - 5 -](#_Toc41938633)

[3.1 工作分配情况 - 5 -](#_Toc41938634)

[3.2 影响因素分析 - 9 -](#_Toc41938635)

[4. 实验阶段的对比分析 - 11 -](#_Toc41938636)

[4.1 燃尽图分析 - 11 -](#_Toc41938637)

[4.2 工时统计图分析 - 11 -](#_Toc41938638)

[4.2.1 实验准备 - 12 -](#_Toc41938639)

[4.2.2 软件需求分析 - 13 -](#_Toc41938640)

[4.2.3 软件需求评审 - 13 -](#_Toc41938641)

[4.2.4 软件设计与实现 - 14 -](#_Toc41938642)

[4.2.5 软件测试 - 14 -](#_Toc41938643)

[4.2.6 软件测试评审 - 15 -](#_Toc41938644)

[4.2.7 实验6-8 - 15 -](#_Toc41938645)

[5. 成员工时的对比分析 - 17 -](#_Toc41938646)

[5.1 成员工时统计 - 17 -](#_Toc41938647)

[5.2 影响因素分析 - 17 -](#_Toc41938648)

[6. 经验与建议 - 18 -](#_Toc41938649)

[6.1 总结经验 - 18 -](#_Toc41938650)

[6.2 课程建议 - 18 -](#_Toc41938651)

1. **引言**

1.1 编写目的

本文档用于报告基于Scrapy的模板化爬虫程序管理平台(EasySpider)的进度控制情况，将从8个实验和5位团队成员这2个角度进行对比分析，在最后总结经验并给出课程建议。

1.2 项目概述

本项目通过提供一组特定网站的爬虫模板，使用Scrapy框架自动生成每个模板对应的爬虫脚本，从而使用户无需编写代码即可获取爬虫采集数据。同时用户可以在平台上运行和管理自己的爬虫程序。本项目在爬虫脚本中加入应对反爬虫机制来增强爬虫程序的鲁棒性，通过增加分布式爬虫机制来提高爬取效率。

1.3 项目进度控制流程

本项目组通过制定每周计划来控制项目的进度。周计划的内容是根据项目甲方（即软件工程综合实验任课教师）每周在课堂上提出的要求来确定的。为了做到对每周计划执行进度的及时监控，原则上项目组每周召开的需要全组参与的组会次数不少于2次，其中，远程组会沟通工具采用腾讯会议。具体项目每周的进度控制流程如图1.1所示。

图1.1 项目周进度控制流程

项目每周的进度控制流程如下所述：

* 通常情况下本项目组会在每周的周五召开第1次全组会来讨论确定本周的任务，并分配到个人，分配的原则主要是考虑个人意愿，最后由组长落实为会议记录上传至github。
* 全组的第2次组会正常情况下会在每周的周一召开，或者时间由第1次全组会确定。第2次组会的主要内容是组员汇报各自工作情况，全组讨论每个汇报结果，根据讨论结果调整本周计划。
* 每周五13:00前要求项目团队成员采用统一的日志模板填写个人工作日志，并提交至github。13:00之后由项目进度负责人汇总全组的工作日志，使用Microsoft Project项目管理工具形成本周的MPP文件。

1.4 文档组织结构

本文档后面章节的组织结构如下所述：

第2章描述项目计划的变更及其影响因素的分析；

第3章描述项目整体分工及其影响因素的分析；

第4章介绍构成项目的8个实验的进度控制结果及其影响因素的分析；

第5章介绍项目团队内5位成员的工时对比及其影响因素的分析；

第6章总结经验并提出建议。

1.5 数据说明

本文档分析使用的数据包含3部分：个人工作日志、MPP文件、版本管理日志。数据的时间范围为2020年3月6日至2020年5月29日。因为第15周（即5月30日至6月4日）需要完成实验6总结并形成本报告，因此本报告的数据范围只包括教学周第2周至第14周。

1. **计划变更及其影响因素分析**

2.1 计划变更情况

根据版本管理日志，本项目计划变更的具体情况如表2.1所示。

表2.1 项目计划变更记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **更新日期** | **版本** | **更新情况** | **更新原因** | **耗时** |
| 2020/3/13 | 1.0 | 填写第2周任务 | 建立初始文档 | 3h |
| 2020/3/20 | 1.1 | 更新基线的范围为整个项目 | 1.0版本中基线工时为0 | 2h+1h |
| 2020/3/21 | 2.0 | 填写第3周任务 | 日常更新维护 |
| 2020/3/28 | 3.0 | 填写第4周任务 | 日常更新维护 | 1h |
| 2020/4/3 | 3.1 | 更新分支任务和整合任务是FS依赖关系、第3周和第4周是FS依赖关系 | 3.0版本中分支任务与整合任务同时进行、第3周和第4周同时进行 | 1.5h |
| 2020/4/4 | 4.0 | 填写第5周任务 | 日常更新维护 |
| 2020/4/10 | 5.0 | 填写第6周任务 | 日常更新维护 | 1h |
| 2020/4/17 | 5.1 | 更正需求文档更新任务是FS依赖关系；  修改倒数第二级任务名称 | 发现错误：需求文档更新多人同时进行 | 2h |
| 2020/4/18 | 6.0 | 填写第7周任务；  更正部分依赖关系 | 日常更新维护；  部分依赖关系误连 |
| 2020/4/18 | 3.2、4.1、5.2、6.1、7.0 | 更正部分任务层级；  填写第8周任务 | 部分任务层级错误；  日常更新维护 | 5h |
| 2020/5/3 | 8.0 | 填写第9周任务和部分第10周任务 | 日常更新维护 | 1h |
| 2020/5/8 | 9.0 | 填写第10周任务 | 日常更新维护 | 2h |
| 2020/5/15 | 8.1、9.1、10.0 | 更正部分会议序号；  填写第11周任务 | 部分会议序号错误；  日常更新维护 | 3h |
| 2020/5/22 | 11.0 | 填写第12周任务 | 日常更新维护 | 1h |
| 2020/5/29 | 12.0 | 填写第13周任务 | 日常更新维护 | 2h |
| **总计** |  |  |  | **25.5h** |

2.2 影响因素分析

总计结果显示制定项目计划共耗时为25.5h，略高于往届的数据统计结果。主要原因是本届每个小组的大小在5-7人，而往届小组人数在3-4人，由于每周实验6负责人要统计的工作日志数量较多，所以最后总计耗时略高于往届。

经过12周的实验，共计产出了21个版本的MPP文件。每个文件的版本号形如X.Y，其中X按周数递增，Y值从0开始加1递增，X相同的情况下Y值越大表示文件版本越新。之所以版本数量远远超过统计周数，主要原因是不够熟悉Microsoft Project的使用、对其很多功能不甚了解。这导致为了修复很多错误（如依赖关系、基线设置等）对同一周的项目计划更新了多个版本。

从表2.1可以看出，版本1.0是项目计划初始版本，耗时较长是因为Microsoft Project需要一定的学习成本以及工作日志模板的设计工作也需要耗费一定的时间；版本1.1和版本2.0的耗时为2h+1h，其中2h指的是完成2个版本的MPP文件耗费的时间，1h指的是修复工作日志模板CSS显示问题所耗费的时间；其余版本是随着每周的推进逐步增加的，在此过程中如果在某一周发现之前MPP文件存在一定的错误，则实验6负责人会在当周修复错误，更新之前的MPP文件，同时创建当周的项目计划，典型例子如2020/5/15创建的8.1、9.1和10.0所示。

1. **工作分配及其影响因素分析**

3.1 工作分配情况

工作分配的具体情况如表3.1所示。

|  |  |  |  |
| --- | --- | --- | --- |
| **成员** | **职位** | **阶段** | **主要任务** |
| 赵正阳 | 组长 | 实验准备 | * 调研scrapy * 根据调研内容编写项目计划书 |
| 软件需求分析 | * 负责scrapy框架部分的需求分析 * 负责需求文档初版的整合 * 负责需求文档中第1-2章主要内容的编写 |
| 软件需求评审 | * 负责评审G组和A组的需求文档 * 根据评审组意见修改需求文档 |
| 软件设计与实现 | * 负责除节点管理之外，项目其他功能的Django后端的实现 * 负责Django CI中单元测试的编写 |
| 软件测试 | * 负责部分功能性需求测试用例和测试结果的编写 * 负责测试需求文档版本1.0和1.1的整合 * 增加Django后端单元测试和部分场景测试 |
| 软件测试评审 | * 负责评审E组和C组的测试文档 * 根据评审组意见修改测试文档和修复软件缺陷 |
| 实验6-8 | * 负责实验8全部内容（形成工作量统计文档和实验8的输出：实验分析报告） * 负责填写每周个人工作日志 |
| 其他 | * 撰写会议记录 * 制作课堂汇报PPT * 课堂主讲 * 组织召开周组会 * 分配需求评审和测试评审结果给组员 |
| 郭浩隆 | 组员 | 实验准备 | * 调研scrapy可视化项目 * 根据调研内容编写项目计划书 |
| 软件需求分析 | * 负责Web UI部分RUCM的绘制和文字说明 |
| 软件需求评审 | * 负责评审G组和I组的需求文档 * 根据评审组意见修改需求文档 |
| 软件设计与实现 | * 负责Django前端的编写，内容包括：网站首页、用户登录注册、个人信息、管理员模板管理 |
| 软件测试 | * 负责部分功能性需求测试用例和测试结果的编写 |
| 软件测试评审 | * 评审表单设计和评审标准制定 * 负责评审E组和C组的测试文档 * 根据评审组意见修改测试文档 |
| 实验6-8 | * 负责填写每周个人工作日志 |
| 其他 | * 制作课堂汇报PPT并宣讲 * 负责软件工程综合实验总结报告 |
| 沈一聪 | 组员 | 实验准备 | * 调研反爬虫技术 * 根据调研内容编写项目计划书 |
| 软件需求分析 | * 负责需求文档中Web UI的用例图、后端的用例图、后端的RUCM的绘制以及非功能需求中反爬虫的编写 |
| 软件需求评审 | * 评审表单设计和评审标准制定 * 负责评审F组和A组的需求文档 * 根据评审组意见修改需求文档 |
| 软件设计与实现 | * 负责豆瓣网站下音乐、读书、电影3个模板的编写 |
| 软件测试 | * 调研测试产出 * 负责部分非功能性需求测试用例和测试结果的编写 * 增加Scrapy单元测试和部分集成测试 |
| 软件测试评审 | * 负责评审D组和B组的测试文档 * 根据评审组意见修改测试文档和修复软件缺陷 |
| 实验6-8 | * 负责填写每周个人工作日志 |
| 其他 | * 制作课堂汇报PPT并宣讲 * 形成课堂评审意见记录文档 * 负责软件工程综合实验总结报告 |
| 梁远志 | 组员 | 实验准备 | * 调研分布式爬虫技术和scrapyd * 根据调研内容编写项目计划书 |
| 软件需求分析 | * 负责对Scrapyd进行需求分析 * 绘制EasySpider系统架构图、用例图和RUCM图 * 负责编写数据库特性 |
| 软件需求评审 | * 负责评审F组和I组的需求文档 * 根据评审组意见修改需求文档 |
| 软件设计与实现 | * 负责服务器部署 * 负责调度器的实现 * 负责Django CI在github上的配置 * 负责管理员节点管理功能的前后端实现 |
| 软件测试 | * 负责被测软件执行结果分析报告、软件问题报告的编写 * 完成测试需求文档和需求文档的对应表 * 负责部分功能性需求测试用例和测试结果的编写 * 增加调度器单元测试和部分集成测试 |
| 软件测试评审 | * 负责评审D组和B组的测试文档 * 根据评审组意见修改测试文档和修复软件缺陷 |
| 实验6-8 | * 负责填写每周个人工作日志 * 负责实验7的全部内容 |
| 其他 | * 制作课堂汇报PPT并宣讲 |
| 宋冰晨 | 组员 | 实验准备 | * 调研scrapy可视化项目、贡献率计算公式 * 根据调研内容编写项目计划书 |
| 软件需求分析 | * 完成Web UI功能需求和非功能需求的文字描述 * 补充项目背景、用户特点、假定和约束 * 增加Web UI部分的非功能需求 * 设计软件产品原型 |
| 软件需求评审 | * 负责评审G组和I组的需求文档 * 根据评审组意见修改需求文档和修复软件缺陷 |
| 软件设计与实现 | * 负责task、spiderTemplate两个模块的前端实现 * 调研channels，实现基于websocket的前后端消息传递 |
| 软件测试 | * 负责测试计划初版、可用性报告的编写 * 负责部分功能性需求测试用例和测试结果的编写 * 增加部分场景测试和对应测试结果 * 在测试需求规格说明书、被测软件执行结果分析报告中补充可用性测试相关内容 |
| 软件测试评审 | * 负责评审D组和C组的测试文档 * 根据评审组意见修改测试文档 |
| 实验6-8 | * 负责实验6中工作日志模板的设计、MPP文件的制作和进度控制分析报告的编写 * 负责填写每周个人工作日志 |
| 其他 | * 制作课堂汇报PPT并宣讲 |

3.2 影响因素分析

影响表3.1中项目组成员的工作分配情况的主要因素有5个，一是对项目相关的各项内容的熟悉程度，二是前后对应，三是实验阶段，四是课堂点评，五是个人意愿。

对项目相关的各项内容的熟悉程度主要在项目的初期影响分工。具体而言，赵正阳同学对scrapy有使用经验，所以初期项目计划书中scrapy的相关内容由该同学负责；沈一聪同学对selenium等反爬虫工具有使用经验，所以初期项目计划书中反爬虫技术以及项目后端的相关内容由该同学负责；梁远志同学对服务器部署非常熟悉，所以实验准备阶段分配给该同学的任务是调研scrapyd和分布式爬虫部署；宋冰晨同学对javascript、django比较了解，所以主动选择了scrapy可视化作为调研方向；因为项目的Web UI部分任务繁重，所以分配郭浩隆同学也负责前端部分。另外在实验6-8的分工方面，因为宋冰晨同学有Microsoft Project使用经验，所以由该同学做实验6的总负责人；梁远志和赵正阳同学主动提出分别负责实验7和实验8。

上述的分工延续至需求分析与评审阶段，为了做到前后对应。当项目进入设计与实现阶段时，项目组确定了使用Django作为Web应用开发框架，因此重新讨论了成员分工。讨论的结果是：由于赵正阳同学对Django有使用经验，所以由该同学负责大部分Django后端的编写；由于宋冰晨同学对前端知识和Django框架有一定的了解，所以由该同学负责普通用户核心模块的前端；由于沈一聪同学对反爬虫比较熟悉，所以由该同学负责爬虫模板以及反爬虫策略的实现；因为梁远志同学熟悉服务器部署，故由该同学负责调度器的开发、服务器的配置、管理员管理节点部分的前后端开发；由于Web UI部分任务繁重，涉及普通用户和管理员2类用户，故分配郭浩隆同学负责普通用户其他模块以及管理员管理模板部分的前端开发。

当项目进入测试与测试评审阶段的时候，除了沈一聪同学负责非功能性需求测试、赵正阳同学负责Web UI的部分功能性需求测试（减轻宋冰晨同学的测试工作量），剩余成员继续遵循设计与实现阶段的分工，完成各自负责部分的功能性测试，为了做到前后对应。另外，根据甲方（任课教师）在课堂对测试阶段的点评，项目组讨论确定增加项目计划、单元测试、集成测试、场景测试、可用性测试、性能测试。增加的6项内容分配时主要是根据每位成员的个人意愿。

1. **实验阶段的对比分析**

下面先对项目整体燃尽图进行分析，然后从各个实验阶段角度出发，对比不同实验的工时耗费情况。

4.1 燃尽图分析

项目最终燃尽图如图4.1所示。

图4.1 项目最终燃尽图

图4.1中灰线代表的基线剩余累计工时和橙线代表的剩余累计实际工时非常接近，几乎是重叠的关系。这说明为项目计划设置的基线和实际完成情况是比较吻合的。

图4.1中蓝线代表的剩余累积工时与灰线代表的基线剩余累计工时完全重叠，因此并不能分析出什么内容。经过测试，造成这个结果的原因是在维护此文件时，每周都会重新设置一次基线，覆盖整个项目，导致前面的蓝线与灰线完全重合，没有差别。

设置基线实际上代表的是提交一个完整的计划，但是因为在本项目过程中采用的是在每周开始时细化下周计划的方式，如果一直以项目开始时设置的很粗略的计划为基线的话，感觉差别太大，出于这种考虑，才每周设置一次基线，没想到会导致燃尽图出现问题。不过因为所有的基线工时与实际工时都存于每周项目计划文件中，因此问题并不是很大。

4.2 工时统计图分析

各实验阶段耗费工时情况如图4.2所示。

图4.2 工时统计图

从图4.2中可以看出，软件设计与实现阶段耗时最久，这主要是由2个原因造成的。一是因为本项目包含的功能非常多，对应的前后端的代码量都很大，二是因为Django的MVT设计模式致使前后端耦合程度非常紧密，所以前后端负责同学花费在交流沟通上的时间很多。

另一个需要特别说明的地方是软件测试评审阶段耗时高于软件测试需求分析阶段的原因。正常情况下，评审阶段的任务量应该是小于前一个阶段的，如软件需求分析阶段和软件需求评审阶段的关系所示。之所以软件测试评审阶段耗时更高，是因为在课程安排上软件测试阶段只有1周，软件测试阶段课堂展示之后甲方（任课教师）提出了很多修改意见，因此本项目组根据甲方的要求在软件测试评审阶段的第1周内补充完善了测试计划、可用性测试报告、性能测试报告、被测软件执行结果分析报告、软件问题报告、单元测试、集成测试、场景测试等多方面内容，这些改进导致了软件测试评审阶段耗时高于软件测试需求分析阶段。

下面将详细从各个实验阶段对比分析基线工时和实际工时，并对差值做出一定的解释。

4.2.1 实验准备

实验准备阶段的基线工时和实际工时如图4.3所示。

图4.3 实验准备阶段的基线工时和实际工时

从图4.3中可以看到，基线工时差了实际工时7个小时。出现这么大的差距主要原因是对很多调研任务估算的工时过少。

4.2.2 软件需求分析

软件需求分析阶段的基线工时和实际工时如图4.4所示。

图4.4 软件需求分析阶段的基线工时和实际工时

从图4.4中可以看到，基线工时多了实际工时3个小时。两者差距不大，说明经过了实验准备阶段后，对工时的估计更科学合理了。

4.2.3 软件需求评审

软件需求评审阶段的基线工时和实际工时如图4.5所示。

图4.5 软件需求评审阶段的基线工时和实际工时

从图4.5中可以看到，基线工时差了实际工时9个小时。两者差距很大，说明在设置基线时对评审任务的难度和耗时估计过低。

4.2.4 软件设计与实现

软件设计与实现阶段的基线工时和实际工时如图4.6所示。

图4.6 软件设计与实现阶段的基线工时和实际工时

从图4.6中可以看到，基线工时差了实际工时3个小时。两者差距很小，说明经过充分的技术调研之后，软件设计与实现的难度和耗时估计得比较合理。

4.2.5 软件测试

软件测试阶段的基线工时和实际工时如图4.7所示。

图4.7 软件测试阶段的基线工时和实际工时

从图4.7中可以看到，基线工时差了实际工时2个小时。两者差距很小，说明经过了前面5个阶段后，对工时的估计更科学合理了。

4.2.6 软件测试评审

软件测试评审阶段的基线工时和实际工时如图4.8所示。

图4.8 软件测试评审阶段的基线工时和实际工时

从图4.8中可以看到，基线工时差了实际工时2个小时。两者差距很小且差值和上个阶段持平，说明经过了前面6个阶段后，对工时的估计误差能稳定在一个合理的范围内了。

4.2.7 实验6-8

由于在本文档撰写的时候，实验6-8的产出报告尚未完成，故无法对实验6-8的基线工时和实际工时进行对比分析。

1. **成员工时的对比分析**

5.1 成员工时统计

成员工时统计结果如图5.1所示。

图5.1 成员工时统计图

5.2 影响因素分析

从图5.1可以看到，每位成员的基线工时普遍低于实际工时，但差距不大，这说明每周的项目计划制定得还是比较合理的。成员之间的工时耗费出现了一定的差距，主要原因在于分工不同，且知识储备水平不一，导致部分成员投入较多的时间用于调研。工时是个人上报的，存在一定的主观性，这也会导致成员之间存在一定的工时差距。为了避免工时的主观性过分影响成员贡献率的计算，项目组在计算贡献率时更多考虑的是产出制品的工作量比如文档字数。

1. **经验与建议**

6.1 总结经验

在实验6的执行过程中，汇总全组工作日志这个步骤遇到的问题最多，如下所示。

* 每周工作日志的起止时间不一致：有的组员认为每周工作日志记录的任务起止时间为上周五至本周五，有的组员则认为是上周五至本周四。
* 每周全组会的时长不一致：会议记录里没有时长记录，组员各自按照自己的记忆填写会议时长。
* 每周工作日志中包含的任务在时间顺序上是乱序的。

上述3个问题之所以会出现，是因为实验6负责人没有考虑到上述细节，设计的个人工作日志模板包含的信息较少，而且模板里的填写说明和规范有所欠缺。因此导致每周汇总全组工作日志这个步骤耗时较长。

为了解决第1个问题和第3个问题，实验6负责人要求全组在填写个人工作日志时需要按照时间顺序排列各项任务，且在备注一栏注明任务完成日期。为了解决第2个问题，实验6负责人和组长进行沟通，由组长在会议记录上补充一栏，用于填写每次会议的时长，且告知全组在填写个人工作日志时需要核对会议记录文档。

6.2 课程建议

实验6贯穿整个项目，所有相关数据的起始时间应该是第1次课。因此在本学期上完第1次软件工程综合实验课后，实验6的负责人就开始学习Microsoft Project，根据全组工作日志生成MPP文件，但这些MPP文件存在一些问题也不够规范。后面第3周的课堂上老师介绍了实验6，详细说明了任务制定、分解和分配要求，并推荐了Microsoft Project的入门教程。建议老师能在学期第1次软件工程综合实验课上详细介绍实验6-8，这样可以节省掉一些Microsoft Project学习成本和对MPP文件一些不必要的修改与返工。不过如果出于“犯错之后再改会更有记性”的考虑出发，也可以保持现在这种出了错再改的情况，不过可能会不利于学生的积极性。