实验小组总结

**实验一总结**

**具体内容**

软件需求分析分为两部分的内容。首先是撰写软件需求规格说明书。文档内容包括引言、软件总体概述、具体需求和环境需求，具体结构如图1.1所示。文档经过多轮的撰写和迭代，其中针对过程中提出问题展开了多次远程会议讨论。



图1.1 软件需求规格说明书

实验的第二部分是在收到老师在课堂上对本组进展和软件需求规格说明书的评审意见后，对每条意见进行检查和讨论，完成反馈，同时根据意见和相关的反馈，对于本组的软件需求规格说明书进行迭代修改，更新版本号。

## 实验产出

本实验产出一份软件需求规格说明书最终版（v5.0）、四份本组修改过程中的迭代版本软件需求规格说明书（v1.0, v2.0, v3.0, v4.0），均已上传至GitHub，如图1.1所示。同时根据修改结果，对软件需求规格说明书进行了版本迭代。



图1.2 实验一产出图

**实验总结与经验**

1. **总结：**由于项目初期阶段，项目的远程协作带来了比较大的挑战，为此在前期我们进行了多次会议来分配、统一项目进度；
2. **总结：**根据意见的反馈和会议的集中讨论进行多次迭代可以比较好的消除文档的错误和整体逻辑的不协调。

**实验二总结**

**具体内容**

软件需求评审分为三部分的内容。首先是制作评审检查单，为后续评审其他组提供标准和依据。在评审检查单中，从总体上划分了9个类别的检查对象，包括需求的正确性、完整性、实践性、质量属性，文档语义和格式，以及RUCM规范性和需求方案的可行性。

实验的第二部分是依据评审检查单对H、I、B、C四个组的软件需求规格说明书进行评审。为了提高评审的效率，本组成员分成了两个小组，分别负责两个小组的评审任务。在评审结束后，每个小组还负责与被评审组的沟通与协调。

实验的第三部分是在收到其他组对本组软件需求规格说明书的评审意见后，对每条评审意见进行检查，完成评审反馈，同时根据评审意见和相关的反馈，对于本组的软件需求规格说明书进行迭代修改，更新版本号。

## 实验产出

本实验产出一份评审检查单格式、四份本组对其他组（H、I、B、C）的评审检查单、四份评审表单反馈和两份修改结果文档，均已上传至GitHub，如图2.1所示。同时根据修改结果，对软件需求规格说明书进行了版本迭代。



图2.1 实验二产出图

**实验总结与经验**

1. **总结：**由于项目初期阶段，项目组的大部分成员对RUCM不够了解，使用不够准确，导致初版的软件需求规格说明书中RUCM图的规范性被评审指出了较多问题，比如缺少进入Alternative Flow的判定条件等。

**经验：**对于不够熟悉的RUCM，组内各位同学应当先进行相关知识的讨论，由此加深对相关知识的理解，并且从一些RUCM的实例中学习规范的表达。在完成RUCM后，再进行互相检查来减少错误。

1. **经验：**由于不同模块的RUCM是由不同的同学负责实现，在汇总和根据评审意见RUCM后，需要统一RUCM的语言描述等内容风格，增强软件规格说明书的整体性。

# ****实验三总结****

## 具体内容

软件设计和实现阶段我们按照设计文档，每个成员负责开发一个算法模块。在开发开始的前几周，我们已经组内交流好了要什么样的编程语言、接口风格，并且对开发和整合的任务进行了到个人的任务划分，由张崇智负责、秦浩桐、王茵迪、黄涵、赵永驰、吴振赫分别负责开发对抗样本生成、量化模型、阅读理解、目标检测、主动学习算法模块和整体打包测试的工作。

这部分实验的不足在于，一开始大家分头开始写代码的时候，没有按照最初设计文档的用例图上对各个算法模块重叠的功能开发一个统一的数据处理和模型保存功能，而是每个人使用了自己习惯的方式编写了这部分代码，后来合并代码的同学意识到这个问题后，协调大家统一了这部分代码，以保证文档和代码的统一。还有一个通过实验暴露出的问题是我们对瀑布开发的不熟悉，虽然本意是想进行瀑布开发，但是从实验过程看，我们在开发过程中和开发后又对设计文档进行了修改，实际上是一种敏捷开发。

## 实验产出

软件产品和设计文档。软件代码已经上传GitHub，软件架构如图

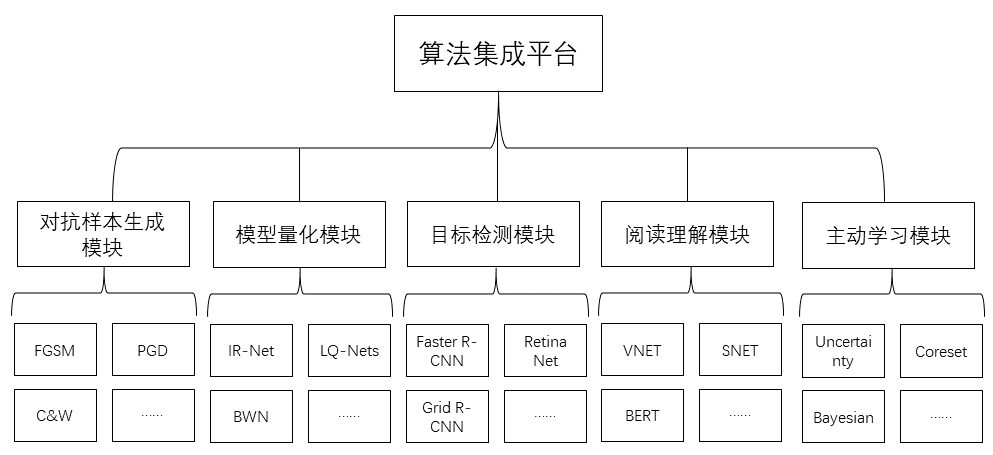


图3.1 软件架构图

# 实验四总结

## 具体内容

本次实验进行完整的软件测试工作，分解软件算法模块，分配给对应的成员。成员需要针对算法模块设计测试样例，完成软件测试和文档撰写，并根据评审意见迭代改进文档和测试样例。主要工作有：

1. 针对对抗样本生成模块、神经网络量化模块、目标检测模块、阅读理解模块和主动学习模块，设计测试样例，开发测试代码。
2. 针对上述模块完成测试需求说明的撰写，对所有的测试用例进行说明。测试用例要覆盖需求分析中的功能性需求和非功能性需求，每个测试用例包含详细的测试步骤，预期结果及评价准则。
3. 针对上述模块完成测试报告的撰写，测试人员针对每个测试用例，运行测试代码，并将运行结果与预期结果进行比对，将测试的详细结果记录在测试报告中。
4. 在两次评审中，分析评审组给出的意见，针对协商后确定有问题的地方仔细修改，完善测试代码、测试需求说明和测试报告，并更新文档的版本号。

## 实验产出

本实验产出测试代码、多个版本的测试需求说明和测试报告。相关内容的统计信息如下：

1. 如图4.1所示，测试代码按照5个软件算法模块分开整理，并提交到GitHub仓库中，代码共831行。
2. 如图4.2所示，完成测试需求说明和测试报告的撰写和改进，并提交到GitHub仓库中，测试报告迭代3版，测试需求说明迭代5版。最终版本的测试报告共106页，38221字，包含对24个单元测试用例、1个集成测试用例、27个功能测试用例、3个非功能测试用例的测试结果。最终版本的测试需求说明报告共92页，36409字，包含对上述所有测试用例的详细描述。

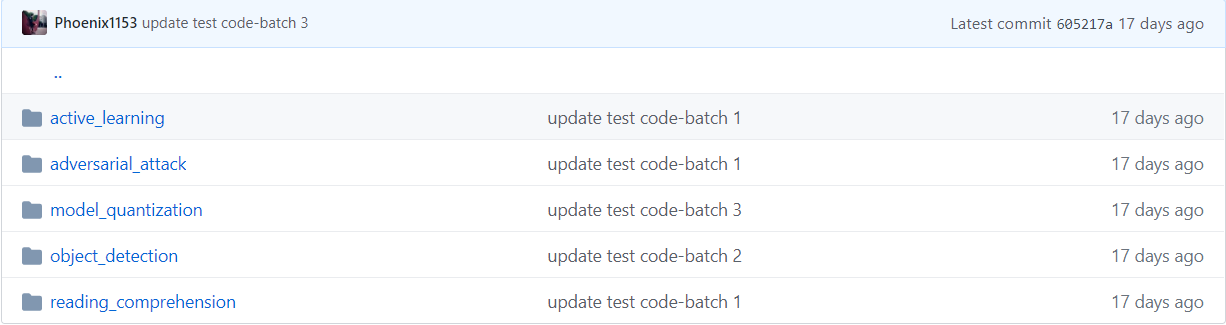


图4.1 测试代码产出



图4.2 测试相关文档产出

## 实验总结与经验

1. 按照CV和NLP的两个开发小组分解测试任务，每个小组有一个小组长，通过组员与小组长汇报，小组长与组长讨论的方式沟通任务进展，完成各个模块内容的合并工作。
2. 每个模块的测试代码开发完之后，组长汇总测试代码和每个模块测试依赖的环境；组长将所有测试代码分发给每个组员，一起测试依赖环境的冲突问题；最后，修改测试代码，解决依赖环境冲突问题。
3. 组员完成各个模块相关文档的撰写之后，组长完成合并工作，并交付专人进行校准。
4. 在测试评审之后，组长首先安排模块对应的负责人与评审组进行沟通讨论；负责人完成对应模块测试文档和代码的完善之后，汇报给组长。

# ****实验五总结****

## 具体内容

软件测试评审分为三部分内容。首先是制作测试评审检查单，为评审其他组提供标准和依据。在测试评审检查单中，本组一共设定了五种检查类别，分别为测试文档语义、文档格式、测试用例完整性和规范性、测试用例准确性、测试方法。

实验的第二部分是依据评审检查单对D、E、F、G四个小组的测试需求说明书和测试报告进行评审。该评审过程通过两周的时间分两次进行，每次评审两个组。本组将组内成员划分为两个小组，每个小组每周负责评审一个其他组的测试需求说明书和测试报告并与该组进行沟通交流。

实验的第三部分是依据D、E、F、G四个小组对本组测试需求说明书与测试报告的评审意见进行反馈与修改。该过程通过两周的时间分两次进行（与第二部分时间同步），每次对两个组进行反馈。反馈与修改包括对对方提出意见的接纳与反驳，接纳的部分在本组的测试需求说明书上进行改进，反驳的部分给出充足理由并与对方组进行及时沟通。

## 实验产出

本实验产出一份测试评审检查单、四份本组对其他组（D、E、F、G）的测试需求说明书评审检查单和测试报告评审检查单、四份对其他组的评审表单反馈，均已上传至Github，如图5.1所示。同时根据修改结果，对本组的测试需求说明书和测试报告进行了内容与版本的修改。



图5.1 实验五产出图

## 实验总结与经验

1. 总结：由于之前有过软件需求评审的的环节，本次测试评审环节在执行上更加有经验了。在制定测试评审检查单时，5个大类34个小类，分条详细而完整。在评审其他组或是对其他组进行反馈时，虽有初始理解的不同，但在沟通后都能很快达成一致。这说明通过评审环节的训练，大家指定细化规则和团队沟通能力皆有所进步。
2. 经验：在评审分工的时候，本组将其他组的报告按段落划分到每一个组员负责，这样做虽然快，但无人复核可能会导致有错、漏问题没看出来。按照评审问题的类别分工至每一位组员或添加最后的复核人员可以解决此问题。

# ****实验六总结****

## 具体内容

软件项目管理和进度控制是按需求确定范围、按目标制定项目计划、按计划执行管理的过程，能够增强对软件开发的控制能力，提升软件开发的质量。本实验主要使用OmniPlan进行任务规划和Excel绘制燃尽图，由高明骏负责。

整体项目的计划为每周进行两次项目会议，在腾讯会议上进行。第一次是在前一周的周六或周日晚，主要针对周五老师对当周工作提出的问题和评价进行修改和总结，其次对接下来一周的目标进行明确，具体任务进行调研和分配；第二次会议是在当周的周三或周四，主要内容是对本周进行的工作进行汇总，并督促未完成的工作，并安排成员对汇总后的结果进行审核，并分配当周汇报任务。

使用微信进行平时交流的主要工具，主要进行一些任务更细节上的交流或者是一些成员进度推迟或提前的报备，或是开发和测试过程中一些问题的交流。

使用omniplan进行任务分解后的规范呈现，对于任务截止时间和前后依赖关系进行清晰的说明。同时设定项目计划的基线和里程碑，在到达里程碑后，能够对当前阶段成果进行检查和确认。每周导出截至下周的进度规划，和组员共享并进行沟通交流，调整各个成员的任务计划。

在中期和实验结束前，对当前进度使用Excel绘制燃尽图，对项目计划进行历时评估，同时调整计划。同时，统计各个成员的工时，并与工作量统计的数据进行对比，观察是否有不统一的项，并分析原因。

## 实验产出

本实验产出每周的任务规划甘特图、一份资源（成员）总结报告、一份任务总结报告、一份项目报告概览、一张项目燃尽图以及一份《进度控制与分析报告》，甘特图和分析报告已上传至GitHub，如图6.1所示，其余报告和图表在分析报告中说明，同时通过Omniplan生成网页版的整体项目报告，可以快速查询任务规划等，如图6.2所示。

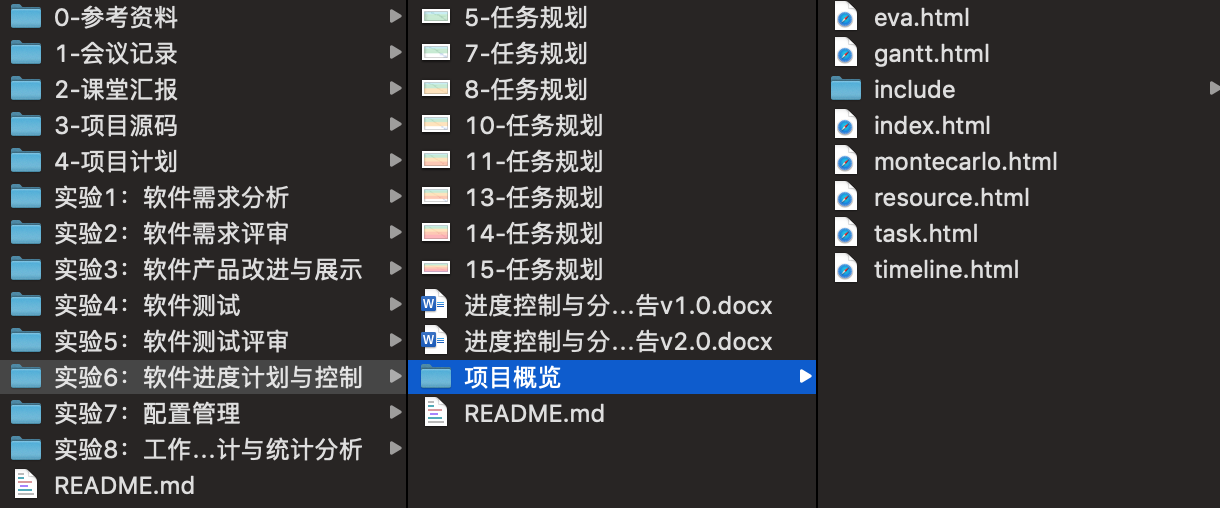


图6.1 实验六产出图



图6.2 项目进度控制概览图

## 实验总结与经验

1. **总结：**在项目初期需求分析阶段，由于小组成员互相不了解，具体的擅长技能和工作节奏不清晰，在会议中不能畅所欲言各尽所长。

**经验：**建议在需求分析阶段之前，对需求分析所需要的基本知识和基本产出有细致的了解划分，在会议上每位成员选择自己擅长或感兴趣的问题，并对文档合并和审核的成员提前安排，否则很容易导致时间节点拖后。

1. **总结：**需求评审阶段，大家对于整个需求分析的要点还不是很明确，但由于团队和其他组之间缺乏沟通等因素，导致评审组与组之间的交流十分拖延。

**经验：**由于需求评审和测试评审阶段都需要对两个组评审和被两个组评审，所以可以根据小组本身的性质将组员划分为两个组，每个组两到三人，小组分别负责一个组的评审和被评审，并为了方便小组与小组沟通，还可以为小组设定一个小组长，小组长和组长进行交流汇报。

1. **经验：**进度控制阶段，如果计划人员使用的是MacOS进行进度控制，那么就可以使用Omniplan，界面美观简洁，支持导出任务报告等，具体介绍和界面在实验六分析报告中给出，缺点是无法自动生成燃尽图，需要手动绘制。同时，在分解任务时，兼顾工作量和难度，不要出现任务都堆积在一个人身上或者是没有人去完成任务。还要同时定期互相提醒各种任务时间节点，不要出现滞后和拖延，避免多等一的现象发生。并且在每周任务分解后，要在当周会议上和成员共享，根据成员自身因素进行调整。

# ****实验七总结****

## 配置管理

在软件开发过程中，从项目的建立到项目最终完善，其版本变更是必须的，而在变更过程中如果没有一套统一的变更控制标准，那么对于团队开发项目进行过程中，很容易造成混乱。因此，软件配置管理(Configuration Management，CM)需要贯穿于整个软件工程实验，在软件研发过程中，采用有效且统一的方法进行软件变更控制和版本管理。

在软件研发过程中，每周对当周在GitHub上的提交数据进行整理，分别以提交的实验制品、所属的实验单元、提交的成员进行统计。最终进行数据的汇总与分析。

## 数据统计

提交次数随天数的变化情况

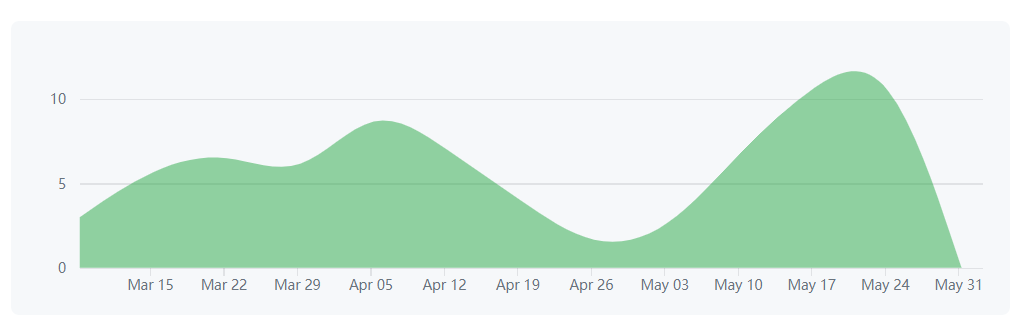


图7.1 提交次数随天数的变化情况

从图中可以看出， commit提交占据了三、四、五月份，提交次数较多的周分别是04/15~04/22（对应实验二-软件需求评审），05/17~05/24（对应实验五-软件测试评审），对于需求评审和测试评审的两周里，提交的次数比较多，符合实际情况。

****一周内的提交变化情况****

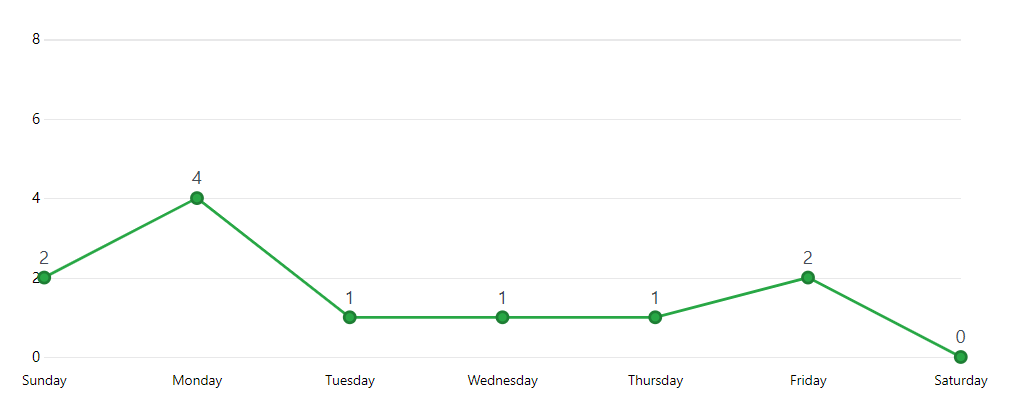


图7.2 05/17~05/24的一周commit分布情况

****各实验阶段提交情况****

表7.1 项目中各实验阶段提交情况

|  |  |
| --- | --- |
| **实验编号** | **相关commit次数** |
| **实验0** | 5 |
| **实验1** | 4 |
| **实验2** | 5 |
| **实验3** | 7 |
| **实验4** | 5 |
| **实验5** | 4 |
| **实验6** | 7 |
| **实验7** | 3 |
| **实验8** | 6 |
| **杂项汇总** | 9 |

综上表观察可知，commit次数在各个实验中分布比较平均，可以展示出各部分制品提交和修改的次数比较一致，证明了工作进展一直比较平均。

## ****版本控制方法****

****Git版本标识****

**将制品初版和修改版提交到远程github仓库的过程由张崇智和高明骏同学完成。commit过程附加的备注信息遵从“第几周-制品名称-修改操作”的命名原则，可以比较准确地反映当次commit完成的操作意义。项目目前的总体情况由图7.3所示。**



图7.3 项目github仓库情况

****文档版本标识****

对于该软件过程中所有的所有文档对象，其规范的版本号采用X.0的标识，以实验一的软件需求规格说明书的版本号变更及变更记录为例，表7.2进行了展示。

表7.2 软件需求规格说明书版本变更情况

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **修改人员** | **审核人员** |
| 2020.03.25 | V1.0 | 无 | 王茵迪，赵永驰，秦浩桐，黄涵，吴振赫，张崇智，高明骏 | 张崇智，高明骏 |
| 2020.03.31 | V2.0 | 针对上周老师的意见进行针对性修改 | 王茵迪，赵永驰，秦浩桐，黄涵 | 张崇智，高明骏 |
| 2020.04.04 | V3.0 | 针对参考文献等细节部分进行了修改 | 张崇智 | 高明骏 |
| 2020.04.12 | V4.0 | 针对第一次评审意见进行了针对性修改 | 王茵迪，赵永驰，秦浩桐，黄涵，吴振赫，张崇智，高明骏 | 高明骏，张崇智 |
| 2020.04.15 | V5.0 | 针对第二次评审意见进行了针对性修改 | 王茵迪，赵永驰，秦浩桐，黄涵，吴振赫，张崇智，高明骏 | 高明骏，张崇智 |

## 总结

在配置管理期间，没有发生文档覆盖、无法解决冲突的情况。

采用了规范化的注释使得提交显得较为清晰，并且在提交期间，大多数提交均能够按照约定提交，能够在后期的数据收集与统计。

# 实验八总结

## 具体内容

本次实验通过追踪项目的实验过程，统计每次实验的工作量，并分析其差异，主要工作有：

1. 每周周五根据当周具体工作内容制作统计表格并发给成员，在周六进行汇总，并上传至GitHub。
2. 在中期和终期进行数据的整理汇总，对每个阶段的产出如报告、代码、图表和所用工时等进行统计，并统计每位成员在各个实验阶段的具体工作，以及所贡献的文档、图表、代码和所用工时，并绘制统计图。
3. 对每个实验阶段的统计图和每位成员的统计图之间存在的差异进行对比寻找问题，并分析其主要原因。
4. 对整体项目各个阶段工作量和成员工作量和实验六任务规划的工时进行对比寻找问题，并分析其主要原因。

## 实验产出

本实验产出每周成员的工作量报告，以及一份工作量汇总与分析报告。分析报告中包括实验目标、各个阶段工作量统计，各个成员工作量统计、最后的各项汇总以及统计图，共7723字，21个表格，2个统计图。全部上传至GitHub，如图8.1所示，具体详细内容见《工作量统计与分析报告v2.1》。

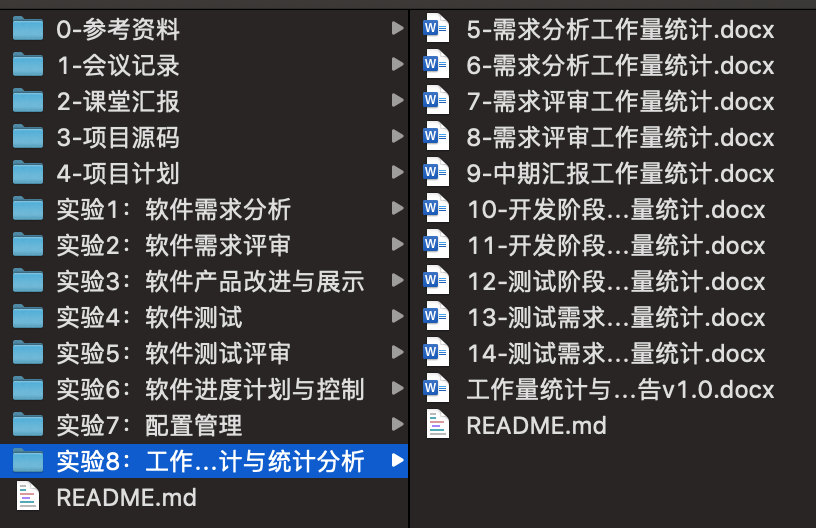


图8.1 实验八产出图

## 实验总结与经验

1. 在每周统计工作量之前提前通过调研或查找往届报告等方式，大致记录当周实验需要统计哪些数据，这些数据之间的关联程度或者在工时方面的相对占比。
2. 每次阶段实验结束后，利用对实验的记忆，对实验中存在的问题进行记录，对成员执行任务的情况进行记录，并进行分析。如果在后期统一分析问题，容易丢失细节，且工作量较大。
3. 每周工作量数据统计应保持公开透明，在每位成员没有异议的情况下再提交至GitHub，保证在最后的整体排名出现后，成员不会存在疑问。
4. 为了保证数据的真实性，应和组长以及进度规划负责人进行沟通，统计结果保证和计划所用工时的一致性。