项目计划与迭代计划

Watermelon Online Judge

组名：Daemon

成员：孙增奎 1707020318

徐维彦 1707020124

张愉飞 1708020112

修订文档历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2020.3.22 | 1.0 | 质量计划与迭代计划 | 徐维彦、张愉飞、孙增奎 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[一、 软件生命周期模型 4](#_Toc35798150)

[二、 WBS 结构 4](#_Toc35798151)

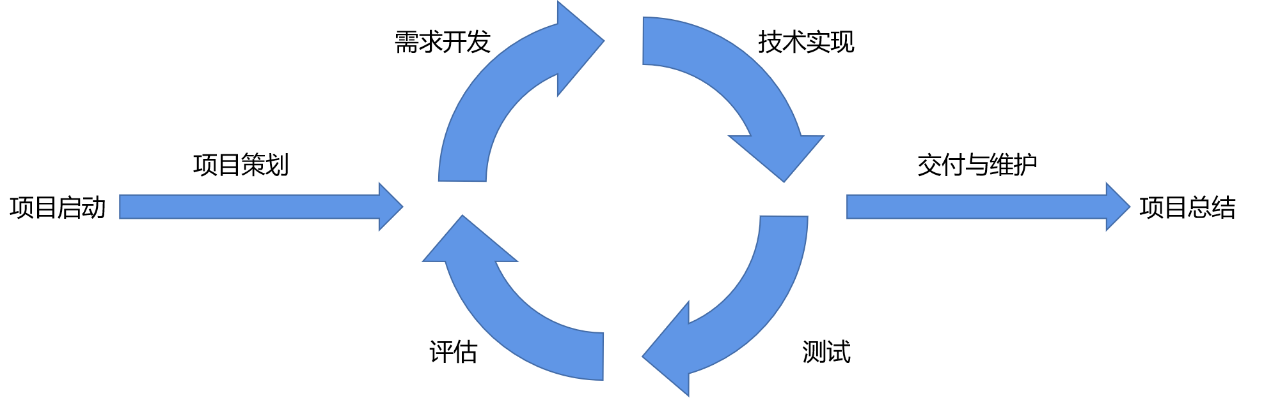
[三、 项目日程及人员日程安排 4](#_Toc35798152)

[四、 质量计划及措施 9](#_Toc35798153)

[五、 风险分析 11](#_Toc35798154)

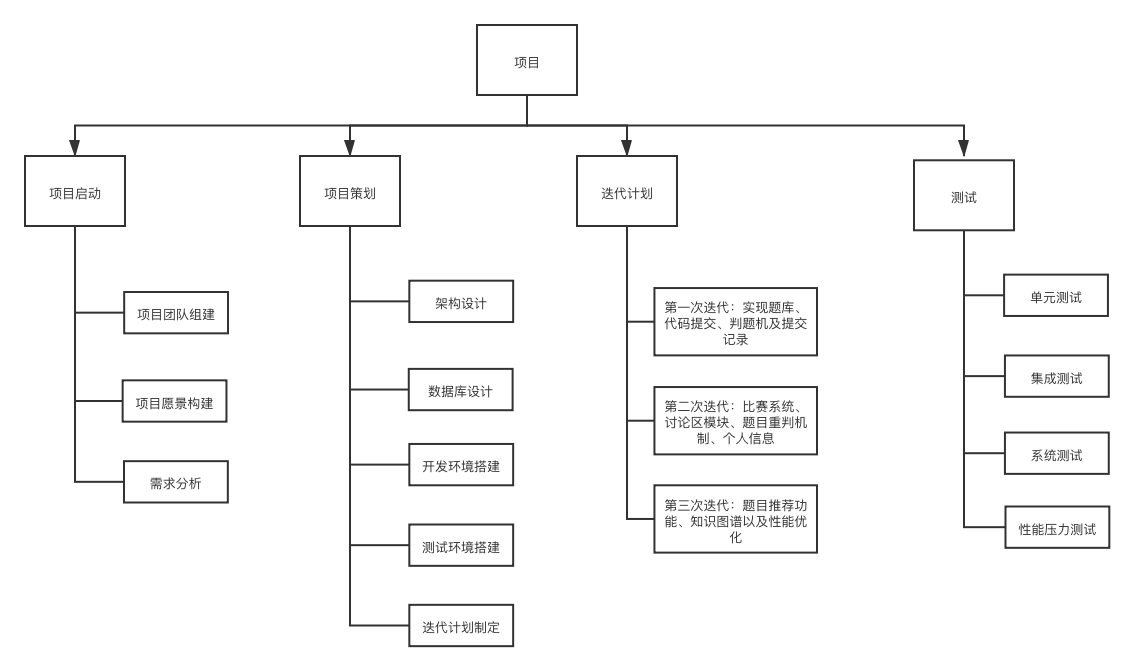
[六、 跟踪计划 12](#_Toc35798155)

## 软件生命周期模型



本项目拟采用迭代式生命周期模型，在每一次的产品迭代都完成部分OJ 功能，产生一个可以发布的产品，这个产品是最终产品的一个子集，并且迭代过程中不断总结评估，对下一个迭代计划的修改更正，推动迭代进行。

## WBS 结构



## 项目日程及人员日程安排

本项目将分三次迭代，每次迭代产生一个可以发布的产品。在第一、二次迭代中，主要目标是将该产品的核心功能完成，在第三次迭代中加入特色功能，并对系统进行压力测试，性能优化等任务，每个迭代中都要进行集成测试。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 阶段 | 项目计划内容 | | 起始时间 | 完成时间 | 所需时间（小时） | 负责人 |
| 项目启动阶段 | 项目组建 | 组建项目团队 | 2.16 | 2.16 | 2 | 孙增奎 |
| 同类项目分析 | 2.18 | 2.18 | 2 | 张愉飞 |
| 可行性分析 | 2.18 | 2.18 | 2 | 徐维彦 |
| 项目愿景构建 | 项目目标定位 | 2.23 | 2.23 | 3 | 张愉飞 |
| 编写愿景文档 | 2.25 | 2.27 | 12 | 张愉飞 |
| 需求分析 | 干系人分析 | 3.1 | 3.1 | 4 | 孙增奎 |
| 编写用户故事 | 3.1 | 3.2 | 8 | 徐维彦 |
| 用例分析 | 3.3 | 3.3 | 6 | 孙增奎 |
| 编写用例规格说明书 | 3.3 | 3.5 | 13 | 徐维彦 |
| 系统架构设计 | 分析设计系统架构 | 3.8 | 3.10 | 7 | 徐维彦 |
| 分析架构可行性 | 3.10 | 3.10 | 4 | 张愉飞 |
| 第一次迭代 | 第一次迭代计划制定 | 分析确定迭代要完成的用户故事 | 3.16 | 3.16 | 3 | 孙增奎 |
| 分配任务编写人员日程安排表 | 3.16 | 3.17 | 6 | 徐维彦 |
| 编写测试计划 | 3.19 | 3.21 | 10 | 徐维彦 |
| 数据库设计 | 分析系统实体类与接口 | 3.22 | 3.22 | 3 | 张愉飞 |
| 设计数据库表 | 3.22 | 3.22 | 2 | 孙增奎 |
| 编写数据库与Java属性对应关系 | 3.22 | 3.22 | 1 | 徐维彦 |
| 编写实体类代码统一格式 | 3.22 | 3.23 | 15 | 张愉飞 |
| 开发环境搭建 | 服务器搭建 | 3.18 | 3.19 | 6 | 孙增奎 |
| 集成环境搭建 | 3.20 | 3.21 | 6 | 孙增奎 |
| 统一编码格式 | 3.21 | 3.21 | 2 | 徐维彦 |
| 题库搭建 | 查看同类网站 | 3.20 | 3.20 | 2 | 张愉飞 |
| 利用爬虫获取题目数据 | 3.20 | 3.22 | 6 | 张愉飞 |
| 数据整合，存入数据库 | 3.22 | 3.23 | 10 | 徐维彦 |
| 搜索题目功能实现 | 设计编写题目列表界面 | 3.23 | 3.30 | 14 | 徐维彦 |
| 编写后端代码 | 3.23 | 3.30 | 16 | 孙增奎 |
| 进行功能测试、单元测试 | 3.31 | 3.31 | 4 | 孙增奎 |
| 查看题目功能实现 | 设计编写题面界面 | 3.23 | 3.30 | 12 | 徐维彦 |
| 编写后端代码 | 3.23 | 3.30 | 16 | 张愉飞 |
| 进行功能测试、单元测试 | 4.1 | 4.1 | 4 | 张愉飞 |
| 查看提交记录功能实现 | 设计编写提交状态界面 | 4.2 | 4.9 | 18 | 徐维彦 |
| 编写后端代码 | 4.2 | 4.9 | 15 | 孙增奎 |
| 进行功能测试、单元测试 | 4.10 | 4.10 | 4 | 孙增奎 |
| 提交代码功能实现 | 设计编写题目列表界面 | 4.2 | 4.8 | 12 | 徐维彦 |
| 使用Markdown对界面格式进行优化 | 4.9 | 4.9 | 4 | 徐维彦 |
| 编写后端代码 | 4.2 | 4.9 | 20 | 张愉飞 |
| 进行功能测试、单元测试 | 4.10 | 4.10 | 4 | 张愉飞 |
| 判题系统 | 编写判题机 | 3.23 | 3.30 | 24 | 孙增奎 |
| 设计编写判题系统 | 3.23 | 3.30 | 24 | 孙增奎 |
| 测试系统可行性 | 3.31 | 3.31 | 4 | 徐维彦 |
| 第一次集成测试 | 集成各功能后端代码进行集成测试 | 4.11 | 4.11 | 3 | 张愉飞 |
| 集成前端与后端代码进行线上的功能测试 | 4.11 | 4.11 | 3 | 徐维彦 |
| 第一次迭代的验收及总结 |  | 4.11 | 4.11 | 2 | 孙增奎 |
| 第二次迭代 | 第二次迭代计划制定 | 分析确定迭代要完成的用户故事 | 4.12 | 4.12 | 2 | 张愉飞 |
| 分配任务编写人员日程安排表 | 4.12 | 4.12 | 2 | 孙增奎 |
| 编写测试计划 | 4.12 | 4.12 | 3 | 徐维彦 |
| 比赛系统实现 | 比赛系统功能分析设计 | 4.13 | 4.13 | 2 | 张愉飞 |
| 比赛系统前端实现 | 4.13 | 4.15 | 4 | 徐维彦 |
| 赛题题目提交实现 | 4.13 | 4.15 | 4 | 孙增奎 |
| 比赛系统单元测试 | 4.14 | 4.16 | 3 | 徐维彦 |
| 比赛榜单实现 | 比赛榜单功能与接口设计 | 4.14 | 4.16 | 2 | 张愉飞 |
| 榜单前端实现 | 4.15 | 4.18 | 10 | 徐维彦 |
| 榜单数据实时更新实现 | 4.15 | 4.18 | 10 | 张愉飞 |
| 榜单单元测试 | 4.18 | 4.20 | 3 | 孙增奎 |
| 查看个人信息功能实现 | 查看个人信息功能分析与设计 | 4.18 | 4.18 | 2 | 徐维彦 |
| 个人信息前端渲染 | 4.18 | 4.21 | 3 | 徐维彦 |
| 个人信息后端实现 | 4.18 | 4.21 | 4 | 张愉飞 |
| 个人信息功能单元测试 | 4.22 | 4.22 | 3 | 孙增奎 |
| 题目重判机制 | 题目重判功能与接口设计 | 4.22 | 4.22 | 3 | 张愉飞 |
| 题目数据格式验证 | 4.22 | 4.25 | 6 | 孙增奎 |
| 重判后端实现 | 4.22 | 4.25 | 10 | 孙增奎 |
| 题目重判单元测试 | 4.26 | 4.26 | 3 | 徐维彦 |
| 讨论区模块设计与实现 | 讨论区功能与接口设计 | 4.21 | 4.22 | 3 | 张愉飞 |
| 讨论区前端实现 | 4.27 | 4.30 | 6 | 徐维彦 |
| 讨论区后端实现 | 4.27 | 4.30 | 6 | 张愉飞 |
| 讨论区单元测试 | 5.1 | 5.2 | 3 | 孙增奎 |
| 第二次集成测试 | 集成各功能后端代码进行集成测试 | 5.2 | 5.6 | 10 | 孙增奎、张愉飞 |
| 集成前端代码与接口进行线上的功能测试 | 5.2 | 5.6 | 10 | 徐维彦 |
| 第二次迭代的验收及总结 |  | 5.7 | 5.8 | 4 | 张愉飞、孙增奎、徐维彦 |
| 第三次迭代 | 第三次迭代计划制定 | 分析确定迭代要完成的用户故事 | 5.9 | 5.9 | 3 | 孙增奎 |
| 分配任务编写人员日程安排表 | 5.9 | 5.9 | 3 | 张愉飞 |
| 编写测试计划 | 5.9 | 5.9 | 3 | 徐维彦 |
| 题目推荐功能实现 | 引入Python中sklearn | 5.10 | 5.11 | 3 | 张愉飞 |
| 使用已知数据进行训练 | 5.10 | 5.15 | 8 | 徐维彦 |
| 能力分析功能实现 | 后端具体实现 | 5.10 | 5.14 | 6 | 孙增奎 |
| 前端引入视觉脚本以实现 | 5.12 | 5.15 | 4 | 徐维彦 |
| 评测队列优化 | 优化评测队列策略 | 5.10 | 5.15 | 8 | 张愉飞 |
| 对新策略进行单元测试 | 5.16 | 5.17 | 3 | 孙增奎 |
| 后端缓存优化 | 引入Redis与Memcache | 5.16 | 5.17 | 3 | 徐维彦 |
| 优化后端代码 | 5.17 | 5.20 | 6 | 徐维彦 |
| 对新接口进行单元测试 | 5.21 | 5.22 | 3 | 张愉飞 |
| 引入分布式文件存储 | 引入MonogoDB | 5.18 | 5.19 | 2 | 孙增奎 |
| 修改相应文件储存代码 | 5.20 | 5.24 | 10 | 张愉飞 |
| 对新代码进行单元测试 | 5.25 | 5.26 | 3 | 孙增奎 |
| 知识图谱实现 | 爬取相应数据 | 5.16 | 5.19 | 6 | 张愉飞 |
| 构建知识图谱 | 5.20 | 5.25 | 10 | 徐维彦 |
| 性能（压力）测试 | 对服务器进行高并发优化 | 5.24 | 5.27 | 7 | 孙增奎 |
| 测试服务器对Ip访问数量的性能反馈 | 5.28 | 5.30 | 5 | 张愉飞 |
| 系统测试 |  | 5.28 | 6.2 | 10 | 张愉飞、孙增奎、徐维彦 |
| 项目总结 | 编写项目总结报告 | 6.1 | 6.4 | 10 | 张愉飞、孙增奎、徐维彦 |

## 质量计划及措施

#### 软件质量目标

|  |  |  |
| --- | --- | --- |
| 度量 | 目标值 | 目标值设定原因 |
| 测试覆盖率 | 80% | 尽可能多的识别软件的缺陷，以保证软件质量 |
| 软件开发进度估计准确性 | 90% | 较准确的估计开发进度以确保项目能在按时完成或提前完成 |
| 测试有效性 | 95% | 确保测试用例的有效性以减少无效测试的资源浪费 |
| 代码缺陷密度（个/KLOC） | 30 | 保证代码质量 |
| 软件完成率 | 95% | 应完成软件的基本功能以及绝大多数用户常用功能，其他功能需根据项目进度而定 |

#### 质量计划及措施

| **任务** | **主要活动** | **措施及结果** | **完成情况** |
| --- | --- | --- | --- |
| 项目筹备 | 构思项目、组建团队 | 确定小组成员，开发项目、并分工 | 良好 |
| 项目愿景 | 构思与分析项目情况 | 小组讨论并完成项目愿景文档 | 良好 |
| 质量属性 | 系统架构设计 | 小组讨论并完成项目4 + 1视图 | 良好 |
| 开发规范 | 指定代码规范、接口规范 | 讨论后拟采用Alibaba代码规范、RESTful API 接口规范 | 良好 |
| 需求分析 | 分析项目所需功能 | 经讨论后，拟定开发题库、比赛、讨论区三大功能及附属特色功能等 | 良好 |
| 系统分析 | 分析系统框架 | 拟采用分层结构，拟采用mvc框架 | 良好 |
| 技术分析 | 讨论系统各技术所采用技术 | 对不同的模块进行技术、框架、算法等分析评估 | 良好 |
| 迭代计划 | 讨论制定项目迭代计划 | 小组成员分析讨论，拟分三次迭代，对功能开发进行划分 | 良好 |
| 开发环境 | 确定系统的开发环境 | 拟采用 Ubuntu 云服务器作为服务器，采用Github进行版本管理，采用自动部署 | 良好 |
| 代码编写 | 对模块进行开发 | 部分功能采用前后端分离开发，部分功能采用全栈开发 | 进行中 |
| 代码审查 | 代码编写规范、接口规范检查 | 每次向代码仓库合并前，需至少一名小组成员进行审查 | 进行中 |
| 运行检测 | 在新旧代码集成前检测运行情况 | 通过 Continuous integration 进行检测，检测通过后通过 CD 进行自动部署 | 进行中 |
| 模块测试 | 对模块功能进行测试 | 采用单元测试、集成测试、回归测试、系统测试 | 进行中 |
| 模块优化 | 对模块性能进行优化 | 通过改进技术方案与优化运行逻辑、算法对模块进行性能优化 | 进行中 |
| 开发会议 | 对开发过程中的事项进行讨论 | 在每个阶段开始与结束时总结会议；并通过例会及时调整进度及开发方案 | 进行中 |
| 文档编写 | 对所进行环节记录文档 | 在开发环节中用文档记录开发细节 | 进行中 |

## 风险分析

#### 风险评估分值表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 评估值=损失程度\*可能性 | | 损失程度 | | |
| 严重（3） | 中度（2） | 轻微（1） |
| 发生可能性 | 高（3） | 9 | 6 | 3 |
| 中（2） | 6 | 4 | 2 |
| 低（1） | 3 | 2 | 1 |

在项目软件的开发周期内，不断的循环执行风险识别、风险评估、风险处置、风险跟踪和监控，识别并预防风险，保证软件开发过程稳定进行。

#### 风险列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 项目风险 | 发生可能性 | 风险评估分值 | 风险说明 | 预防及应对策略 |
| 1 | 项目进度安排不合理 | 中 | 4 | 在项目开发时出现某些任务实际花费时长超出计划时长的情况 | 对计划及时进行调整，团队成员合理调整开发时间 |
| 2 | 技术实现困难 | 高 | 6 | 某些功能的所采用的技术无法达到预期的效果 | 团队成员及时交流思路或者开发者浏览相关技术博客 |
| 3 | 操作逻辑问题 | 中 | 6 | 不同功能集成后出现某些操作不合理、无效操作、逻辑矛盾等问题 | 指定计划时尽可能考虑该情况，若未考虑到应及时讨论，修改逻辑，测试用例应尽可能全面 |
| 4 | 数据评测问题 | 低 | 3 | 由于上传数据文件的格式问题，导致评测机评测错误 | 限制编码格式，尽可能预支持主流操作系统之间的格式转换 |
| 5 | 代码评测速度慢 | 高 | 9 | 同一时间段内，大量代码为待评测状态或部分代码需消耗大量资源 | 对提交代码运行进行资源限制，优化评测队列处理逻辑，提高评测机效率 |
| 6 | 系统响应慢 | 中 | 6 | 大量用户同时操作 | 加大服务器带宽，采用分布式，进行缓存处理等 |
| 7 | 成员协作问题 | 中 | 4 | 由于不可抗力限制，开发人员之间交流不及时 | 可采用线上会议、规定硬性工作时间段等方法 |

## 跟踪计划

#### 计划方案

1. 保持对所确定工作任务的追踪，及时更新项目跟踪相关报表并定期报告项目进展情况
2. 定期召开例会，团队成员汇报自己的工作记录及进度
3. 按照个人特长分配任务
4. 及时对模块进行测试，并提交测试文档
5. 提前配置好系统运行与测试环境，并提交配置文档
6. 合理规划项目工作量
7. 指定里程碑，并每周进行评审

#### 里程碑

| **周次** | **里程碑** |
| --- | --- |
| 第一周 | 项目构思 |
| 第二周 | 项目愿景与团队组建 |
| 第三周 | 项目需求分析 |
| 第四周 | 项目架构设计与技术可行性分析 |
| 第五周 | 迭代计划与冲刺计划指定 |
| 第六周 | 创建github仓库，搭建完成开发环境 |
| 第七周 | 题库功能完成，评测功能实现 |
| 第八周 | 题目界面及后端实现，题目提交功能实现 |
| 第九周 | 查看提交记录及其前端界面实现，完成判题系统 |
| 第十周 | 比赛功能、讨论区功能实现 |
| 第十一周 | 个人信息功能实现 |
| 第十二周 | 完成集成测试、找出潜在Bug |
| 第十三周 | 推荐功能实现 |
| 第十四周 | 完善知识图谱 |
| 第十五周 | 压力测试 |
| 第十六周 | 进行压力测试、解决发现的Bug |