1. 风险管理计划
   1. 风险评估

风险是指在项目进行过程中可能发生的时间，这些时间将会对项目按预期时间、资源和预算完成产生重大影响。风险分析的目标是识别这些事件，设法避免这些事件的发生并制定一旦这些事件发生后的处理措施。

表 8‑1发生概率程度表

|  |  |
| --- | --- |
|  | 概率程度 |
| 发生概率高 | 发生概率>80% |
| 发生概率中 | 发生概率30%~80% |
| 发生概率低 | 发生概率<30% |

表 8‑2影响程度描述表

|  |  |
| --- | --- |
|  | 不利影响程度 |
| 极高影响程度 | 可以造成整个工程的瘫痪，直接导致项目最终的不成功的发生，无法得到补救。 |
| 高影响程度 | 给整个工程带来非常大的不利影响，间接导致项目的不成功，需要长时间的补救。 |
| 中影响程度 | 给整个工程造成不利影响，是可以补救的影响，不会对项目最终的成功造成影响。 |
| 低影响程度 | 给整个工程造成一些影响，可以通过已知的手段结束这种影响。 |

表 8‑3项目风险程度表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 风险等级 | 极高影响程度 | 高影响程度 | 中影响程度 | 低影响程度 |
| 发生概率高 | 高风险 | 高风险 | 高风险 | 中风险 |
| 发生概率中 | 高风险 | 中风险 | 中风险 | 中分线 |
| 发生概率低 | 中风险 | 中分线 | 底风险 | 低风险 |

下面的这个风险列表就是通过一系列的风险识别、风险评估、风险应对，最后得出项目风险计划表。

表 8‑4项目风险计划表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 风险管理计划表 | | | | | |
| 风险发生概率的判断准则： | | | | | |
| 发生概率程度定义：如表8.1  不利影响程度定义：如表8.2  风险等级定义：如表8.3 | | | | | |
| 序号 | 风险描述 | 发生概率 | 不利影响程度 | 风险等级 | 风险响应计划 |
| 1 | 由于时间紧张最后无法按期完成 | 低 | 高 | 中 | 充分考虑各种潜在因素，适当留有余地；任务分解要详细，便于考核；如果出现必须延期的情况，组长需及时同相关负责人沟通，并申请延期时间。 |
| 2 | 开发软件结构体系存在问题，使完成的软件产品未能实现项目预定目标 | 中 | 极高 | 高 | 选用正版软件开发 |
| 3 | 质量不符合用户要求 | 低 | 高 | 中 | 经常和用户交流工作成果、认真组织对产出物的检查和评审、计划和组织严格的独立测试等 |
| 4 | 组员成员因意外无法参与设计 | 中 | 中 | 中 | 实现同用户商量解决办法 |
| 5 | 计划过于乐观 | 中 | 中 | 中 | 使用多种方法进行估计、主动加班 |
| 6 | 项目周期过长，影响项目组员情绪，效率低下 | 中 | 中 | 中 | 缩短开发周期、制定相应激励政策、鼓励创新，提高人员积极性 |
| 7 | 项目原定采用原型开发，实际开发过程中逐渐偏离原型，无法满足客户需求 | 低 | 高 | 中 | 项目开始前与 |
| 8 | 使用新的技术，使用经验不足 | 中 | 低 | 中 | 认真评估使用新技术的预期效果、加强培训、以老带新 |
| 9 | 使用旧的技术，难以满足某些特定的功能、性能要求 | 低 | 低 | 低 | 评估特定功能、性能要求；完善设计 |
| 10 | 需求描述有二义性，开发人员理解错误 | 高 | 高 | 高 | 加强需求评审；每次项目组例会中需求问题沟通；加强客户沟通 |
| 11 | Git远程仓库奔溃导致文档缺失 | 低 | 极大 | 中 | 周期性地使用Git，检查和及时备份所有的文档 |
| 12 | Git远程仓库空间不足 | 低 | 高 | 中 | 周期性的使用Git，检查使用空间的大小 |
| 13 | 项目文件结构不符合要求 | 中 | 高 | 中 | 配置管理员修改文件结构 |
| 14 | 开发人员经验不足 | 中 | 中 | 中 | 开发测试人员固定学习新知识 |
| 15 | 组内信息回复的不够及时 | 低 | 低 | 低 | 为微信群开启通知功能 |
| 16 | 组内成员对于项目开发积极性不够 | 中 | 中 | 中 | 每周举办TeamBuild加强组内人员关系，组长积极做心理工作。 |

1. 配置系统管理指南
   1. 配置项

包括可行性报告、项目计划书、需求工程计划书、软件需求规格说明计划书、软件需求变更计划、系统设计与实现计划、软件概要设计说明、测试和运维计划书、会议记录等受控文档。

* 1. 配置命名

Git个人提交命名规则：PRD-2018-G14-文件名（名字首字母小写）

Git整合文档提交命名规则：PRD-2018-G14-文件名

* 1. 版本管理
     1. 版本格式要求

文档版本为[主版本号.子版本号.修正版本号]

例如：0.1.1

文档的起始版本为：0.1.0

* + 1. 版本详细流程说明

1. 第一步在github上建立期七个分支，作为配置数据库，分别是：master, intergration, zhugezx, chengtk, dengx, zhuangyx, chenwf。它们的用途分别是：放置最终文件，放置整合文件，之后是小组成员的名字首字母，存放相应小组成员的个人作业。
2. 第二步是在master目录中建立受控文件和非受控文件，其中受控文件指的是可以通过版本迭代后更改的文件，非受控号文件指的是一些外部文件，用作一些特定的用途。Intergration分支和master分支类似，但是放置的文件都是未经过评审的文件，是一个暂时的文件放置地。个人分支主目录只有受控文件，第二层目录是Code，Diragram,Document,Minutes,Powerpoint,作用分别是：放置代码，放置图表，放置会议记录，放置展示用PPT文件。详细构造图如下所示。
3. 在项目开发的某个里程碑结束时，通过了该评审阶段的这些开发文档交配置管理员保存到分支master，并且作为正式版本版本号为：0.1.0。
4. 在以后的开发中，如果受控文档需要更改迭代时，更新它的版本号，将它的修正版本号+1处理。每增加到8的周期，子版本号+1处理。
   1. 变更控制
      1. 正常改正时的变更控制

1.在评审或测试后发现的问题由项目经理或业务分析师通知配置管理员，并且及时更正。

2.由配置管理员将需要修改的地方反应出来，然后通知开发人员执行修改。

3.修改完毕后，组内进行测试，当测试完成最终检查，通知配置管理员去修改文件的版本号，上传至github，当测试错误累积到一定的量或者测试时间超过一个星期，通知配置管理员测试无法完成，通知项目经理以及业务分析师去修改更改重新进行估量可行性。

4.配置管理员修改《软件配置状态表》和《软件变更记录表》，以使其他相关开发人员及时了解软件变化情况。

* 1. 配置审核

为保证各项产品在技术上和管理上的完整性，根据杨枨老师在课堂上的要求和候老师的评审计划表，在软件开发过程中的详细设计阶段和测试阶段完成时，对配置情况进行抽查。先提出要审核的内容和各项指标，逐项审核完成后要作好记录。

* 1. 配置项管理方法

这里只统一文档的命名，以下是项目文档命名规范：

|  |  |
| --- | --- |
| 文件种类 | 文档名称 |
| 需求工程项目计划 | PRD2018-G14-项目计划 |
| 可行性研究报告 | PRD2018-G14-可行性研究报告 |
| 需求工程计划 | PRD2018-G14-需求计划-初步 |
|  |  |
|  |  |
|  |  |
|  |  |

* 1. Git配置
     1. 关于分支

1. master：主分支
2. zhugezx
3. zhuangyx
4. chengtk
5. chenwf
6. dengx
   * 1. 关于文件
7. 受控文档：用于可以迭代修改版本的文档
8. 非受控文档：用于不可迭代修改版本的文档
   * 1. 关于文件格式
9. 自己分支上，PRD2018-G14-文件名称(名字拼音)，例：PRD2018-G14-项目计划(dengx)
10. 主分支上，PRD2018-G14-文件名称
    * 1. 关于文件存放

每类文件都有固定的文件夹，请严格按照文件夹分类。

1. document：存放项目文档，如这次的项目计划。
2. diagram：存放项目各种工具图，如这次的OBS图，甘特图，WBS图等。
3. minutes：存放会议纪要以及会议录音。
4. powerpoint：存放ppt。
5. code：存放代码。

平时个人完成的部分请切换到自己分支并放在非受控文档中，并按照文件夹分类。

Git目录结构如下图所示



Github分支结构图

* + 1. 关于提交

提交信息统一写提交的内容，后面括号自己的名字

* + 1. 关于推送

平时的文件均切换到个人的分支推送，避免推送到主分支上，主分支只存放最终成果。