**第二组-锟斤拷烫烫烫-信息安全系统-项目风险管理**

**（撰稿人：戚正；审稿人：罗承勇、高凡）**

|  |  |  |  |
| --- | --- | --- | --- |
| 风险列表 | 风险描述 | 风险优先级（高、中、低） | 应对办法 |
| 任务比我们想象中的要困难许多，原来制定的日程太乐观。 | 软件固有的特性，人员缺乏估算能力和经验，项目日程没有充分讨论。 | 高 | 让团队多做估算练习 ，老员工给新员工传授经验；分析类似项目的日程；  让团队充分参与项目估算流程.  尽早和项目的利益相关者沟通，尽早提出项目延期的问题.  把长期项目规划为短期里程碑，能很快看到项目延期的问题。 |
| 需求爆炸，我们要做愈来愈多的功能 | 随着项目的进展，我们发现在项目初期没有发现的问题必须要解决，从而影响了原定的日程。 | 高 | 承认这是软件项目经常出现的事实  确保意外出现的需求不影响当前的里程碑  制定长度合适的里程碑，确保问题能尽快浮现  经常和客户+开发人员交流，确认项目的需求.. |
| 人员流失（或请假） | 关键人物离开团队，项目的关键信息丢失，人力资源减少，导致项目困难。 | 高 | 在团队持续鼓励合作和分享，所有代码必须在源代码服务器上，源代码的每一次修改都有记录，所有任务必须在项目管理工具中  鼓励结对编程，代码复审，共同拥有代码，经常报告各种进展和问题。  让多人参与关键模块的工作.  短期的里程碑保证某功能在较短时间得到改进，不让功能处于 “半完成” 状态。 |
| 低质量的规格说明书 | 当开始具体实现和集成的时候，发现spec 包含了相互冲突的需求，或者说明不完备。 | 高 | 让客户或者客户的代表，代理来参加需求分析和spec 复审.  运用场景驱动的方法尽早模拟几个模块集成后能给用户带来什么好处.  尽早用 design by contract 的方法来保证模块之间能相互支持。 |
| 团队效率底下 | 在长期项目中，团队成员失去了紧迫感，缺乏真实世界的反馈，大部分时间浪费在不紧急的事情上。 | 高 | 短期的里程碑和冲刺阶段 – 人们 （学生）通常等到deadline 临近的时候才开始行动.  不断发布/收集反馈，让团队感受真实的需求。 |
| 任务分配不均衡 | 计划和产品定义完全依赖于部分代码能力强的同学，计划虽然是优化过的，但是这是不现实的，实现上有较大的困难。 | 中 | 根据任务完成的实际情况，使分配任务相对较少的同学进行较多的非编码任务，如测试与报告的撰写及协调的工作。  还可以对一些模块任务进行拆分与重组，并进行重新的分配。 |
| 设计方向出现偏差 | 设计不够全面，忽略了必要的任务，导致后期开发困难。设计过于简单或复杂，需要重新设计和实现或导致了一些不必要的工作，影响效率。 | 高 | 全组会议讨论设计出现偏差的具体环节，结合自身能力与具体使用场景，对架构设计进行重新的确认。  适当缩减边缘功能，还可以对用户场景进行适当的修改以适应程序的具体表现形式。 |
| 超出交付时间 | 在限定时间内无法交付成已定规模的模块。 | 中 | 分析超出时间的具体原因，若为超出成员能力范围，可以将相关知识进行相互指导或进行拓展学习，若超出太多可考虑缩减其功能。 |
| 开发难度较大 | 开发规模比我们所估计的要大，人员能力不足。 | 中 | 进行组内的拓展学习，若超出能力太多，可以适当考虑削减其功能的重要性。 |
| 缓冲时间较少 | 交付日期前，项目编码耗时较多，没有相应可用的调整时间。 | 中 | 确定测试用程序的完成期限，有意识地给调整留下较多的缓冲时间。 |
| 陌生领域学习成本较高 | 涉足不熟悉的领域，花费在设计和实现上的时间比预期的更多。 | 中 | 将学习内容进行模块化拆分，分配给相应的成员进行学习，若学习成本过高，则可以适当对程序架构或需求进行调整，绕开较难的需求实现。 |
| 项目结构混乱 | 低效率的项目结构，导致研发进度缓慢，沟通交流困难。 | 中 | 由于程序的架构要对我们的具体能力进行对接，结构的删改是常有现象，需要提前对项目进行合理的拆分，并确认好接口的表现形式。 |
| 适应开发环境时间较长 | 开发工具不如期望那样有效，开发人员需要时间创建工作环境或者切换工具。 | 中 | 对所需要的开发环境进行提前的确定，方便成员对相关环境进行尽早的熟悉。  查找资料对开发工具进行提早的了解，找到学习适应成本尽可能低的开发工具。 |
| 需求分析不清晰 | 需求定义欠佳， 而进一步的定义会扩展项目的范畴,需要大量添加额外的需求。 | 高 | 在发现需求分析不明晰后尽早在组内进行讨论，找到合适的解决措施，对程序应用场景进行深入分析后结合成员能力对需求进行重新确认。 |
| 测试时间超时 | 错误发生几率高的模块需要比预期更多的测试工作，或者由于软件功能的错误，需要重新设计和实现。 | 中 | 留下较多的缓冲时间，以防止出现测试时间不足的情况。  对测试大致需要的时间进行提前预估，对重要模块分配较多的测试时间。 |
| 额外功能开发超时 | 开发额外不需要的功能，极大延长了计划进度。 | 低 | 分析其功能对于整体的程序结构的重要程度或者特色的彰显程度，结合成员能力与已经投入的时间与精力，进行综合的取舍，决定是否要进行保留。 |
| 任务交接融合出现问题 | 接口等问题的确定不明晰，功能模块与主体程序的接入出现问题。 | 中 | 在分配模块任务时对融合的相关事项进行提前的确认，避免因难以融合造成模块的返工，造成时间与精力的浪费。 |
| 进度跟踪不明确 | 进度跟踪不准确，无法预知项目是否落后于计划进度；验收标准不够严，质量跟踪不准确，导致无法得知影响进度的质量问题。 | 高 | 及时对已完成的部分进行存档与上传，以周报的形式对项目进度进行确认与追踪，对验收的标准进行明晰，防止产生较为重大的质量或进度问题。 |
| 系统兼容出错 | 要求与现有系统兼容，需要进行比预期更多的工作。 | 中 | 提前对要使用的不同系统进行兼容性的分析，尽量减少对难以兼容的系统的使用。 |

参考：课件“09-项目管理”P40-P44页