**成都信息工程大学**

**可行性分析报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 学院名称 | 计算机学院 | | |
| 专业名称 | 计算机科学与技术 | | |
| 所属学期 | 2017-2018(1) | | |
| 所属课程 | 基于Java的应用软件开发 | | |
| 项目名称 | Incident Reports | | |
| 小组序号 | 1 | | |
| 小组名单 | 班级 | 学号 | 姓名 |
| **计算机应用153** | **2015051152** | **张龙** |
| **计算机应用153** | **2015051158** | **林能** |
| **计算机应用153** | **2015051144** | **黄香源** |
| **计算机应用153** | **2015051146** | **吴佩明** |
| 指导教师 | **魏 乐** | | |
| 提交日期 | 2017年 10月 15日 | | |

目 录

[1 引言 1](#_Toc498938701)

[1.1 编制目的 1](#_Toc498938702)

[1.2 背景及现状分析 1](#_Toc498938703)

[1.3 参考资料 2](#_Toc498938704)

[2 可行性研究的前提 3](#_Toc498938705)

[2.1 业务分析 3](#_Toc498938706)

[2.2 项目目标 4](#_Toc498938707)

[2.3 系统特点 5](#_Toc498938708)

[2.4 条件、假设与限制 5](#_Toc498938709)

[2.4.1 限制 5](#_Toc498938710)

[2.4.2 解决方案 6](#_Toc498938711)

[2.5 评价尺度 6](#_Toc498938712)

[3 业务与技术可行性 7](#_Toc498938713)

[3.1 对所建系统的说明 7](#_Toc498938714)

[3.2 业务可行性与评估 7](#_Toc498938715)

[3.3 技术可行性与评估 8](#_Toc498938716)

[4 成本模型分析与估算 9](#_Toc498938717)

[4.1 估算方法简介 9](#_Toc498938718)

[4.1.1 用例数估算前提条件 9](#_Toc498938719)

[4.1.2 用例数估算步骤 9](#_Toc498938720)

[4.1.3 总结 11](#_Toc498938721)

[4.2 成本估算实施步骤 11](#_Toc498938722)

[4.2.1 Actor的权重计算 11](#_Toc498938723)

[4.2.2 UseCase权重的计算 12](#_Toc498938724)

[4.2.3 UUCP计算(UUCP：Unadjusted Use Case Point) 12](#_Toc498938725)

[4.2.4 技术复杂度（TCF）计算 13](#_Toc498938726)

[4.2.5 环境复杂度（EF）权重的计算 13](#_Toc498938727)

[4.2.6 UCP(Use Case Point)的计算 14](#_Toc498938728)

[4.2.7 计算项目风险系数 14](#_Toc498938729)

[4.2.8 估算工作量 14](#_Toc498938730)

[4.2.9 估算总时间 14](#_Toc498938731)

[5 项目风险可行性研究 15](#_Toc498938732)

[5.1 风险管理方法 15](#_Toc498938733)

[5.2 风险预测与识别 15](#_Toc498938734)

[5.2.1 定义阶段（可行性研究，需求分析）： 15](#_Toc498938735)

[5.2.2 开发阶段（概要设计，详细设计，实现，测试）： 16](#_Toc498938736)

[5.2.3 运行维护阶段（运行，维护）： 16](#_Toc498938737)

[5.3 风险评估与分析 16](#_Toc498938738)

[5.4 风险及规避措施 18](#_Toc498938739)

[6 可行性分析结论 21](#_Toc498938740)

# 引言

## 编制目的

从现有的技术条件，软硬件资源、管理者与使用者的操作方式以及易用性的角度来·对IncidentReports系统的可行性和必要性进行详细的分析，有助于IncidentReports系统的实施和对系统整体的把握，该系统的实施，将在很大的程度提高用户对自己周边信息的了解程度，帮助用户及时做出决策和判断。

该文档将会帮助开发人员明确业务结构，业务风险和发展现状，提高开发和决策人员认识和把握项目整体，出色的完成项目。

## 背景及现状分析

当前，以互联网为代表的信息技术日新月异，引领了社会生产新变革，创造了人类生活新空间，拓展了国家治理新领域，极大提高了人类认识世界、改造世界的能力，同时，互联网及其相关技术的发展，极大的提高了人民群众的信息获取效率和信息量。同时，互联网也为居民了解自己身边的事件提供了新思路：将居民周围发生的事件一个个都标注在地图上，令居民掌握身边的情况，同时通过允许居民上报事件来提高人民群众的参与度，共同建设美好家园。基于地理信息系统（GIS）的事件地图应用将成为实现事件信息化（更为精细、智能、综合）的新进路。所谓“一图胜万言”，这种寓数于图的事件数据可视化分析不仅在决策、风险评估、轨迹追踪、人口管理中发挥基础性作用，还能以事件地图公开的形式服务社会。

这类事件地图应用主要是基于现有地图构建，在现有地图上添加事件详细信息，并通过GPS数据来标记事件位置。这类系统最核心的两方面即为事件地图的构建与事件数据库的管理。目前国际上已经成型的系统还与各地政府部门和警察、法院等合作，建立基于地图、地理位置的犯罪记录数据库。目前CrimeReports已经和北美地区的500多家执法机构进行合作，获取相关信息。我国早在2013年，就有上海市政协委员提案要求公开犯罪地图，但被公安机关以“担心犯罪地图可能被犯罪分子利用，造成安全防范薄弱区域案件高发，进而影响市民的安全感”为由否决。所以说这类地图应用在国内的发展截至目前是有一定的阻力的，地方执法机构的不可配合导致了虚假信息的风险。但是即便如此，犯罪地图的可视化表达机制、精细化探测效果、智能化分析能力、综合化管理模式依然光彩夺目。2014年，北京、上海、杭州、四川自贡等地公安及检察机关主动公开部分城区的治安地图、反扒地图、盗窃密度地图，这种举措受到好评，但也不乏质疑和担忧。

本系统期望在基本的犯罪信息之外再增设更多类型的事件报告，同时不同类型的事件在地图上的显示生命周期有限（但历史可查），这样聚合各种事件的信息，可以极大的提高居民对自己周围事件的了解程度。

截至目前，已经有越来越多的类似系统出现在欧美国家，而我们国家还没有一个持续运行，公众参与的成型的类似系统。但这些信息的公开，从长远看来都是利大于弊的：他可以作为民众安全生活的行动参考，公众参与犯罪治理的数据基础，以防控知识消除恐惧，以信息化助力执法机构与居民的互动。

况且随着现在地方政府的公信度下降，社会中难以置信的黑暗面仍然存在，公众的参与是非常有必要的，越早建成这样一个系统，人民群众就越能从这个系统中受益，以后无论是出游，出差，搬家，都可以利用该系统的有效信息做出合理且最有利的决策。

## 参考资料

1. 犯罪地图的公开，单勇，<http://www.iolaw.org.cn/showNews.aspx?id=50993>
2. 治安地图，百度百科，<https://baike.baidu.com/item/治安地图>
3. CrimeReports：美国犯罪记录查询网站，网易新闻，<http://tech.163.com/09/1009/11/5L6A1NUU000915BF.html>
4. Crime Reports， <https://www.crimereports.com>
5. 众包导航应用Waze全球用户量突破2800万，新浪新闻，<http://tech.sina.com.cn/t/csj/2012-11-02/17257765419.shtml>
6. Waze官网，<https://www.waze.com/zh/about>
7. 软件项目风险管理(Project Risk Management)，<http://blog.csdn.net/jasonchen_gbd/article/details/51786837>
8. 软件风险控制计划案例, <https://wenku.baidu.com/view/31365712804d2b160b4ec069.html?qq-pf-to=pcqq.c2c>

# 可行性研究的前提

## 业务分析

用户角度：

1. 事件地图清晰直观，已于查看：
   1. 采用不同图标，不同颜色的文字标识不同的事件，并根据事件类型选择明显/暗淡色彩
   2. 在不同的缩放级别下，事件显示方法不同。如，低缩放倍率下，显示该地区的总事件数目，高缩放倍率下，直接标识所有事件于地图上
   3. 支持事件热度图
2. 完备的事件搜索功能和相应的过滤器：
   1. 用户可以根据事件名，事件类型，发生日期，事件状态等事件属性来搜索某一地区的事件
3. 简单直观的事件信息页面：
   1. 用户选中一个事件后显示事件页面，点击地图后隐藏事件信息
   2. 事件页面会在头几行列出事件的重要信息，如事件类型，发生日期，是否被相关单位处理，事件地址。重要信息之后会紧接着更为详细的信息
   3. 事件信息页面还会显示最近15天同类型事件的发生频度图
4. 详细易用的事件报告功能：
   1. 事件真伪检验
   2. 事件详细格式化
   3. 对已有事件的真伪提出疑问
5. 订阅自己住宅周围街道的事件警报，方便用户及时了解自己住宅周围的情况
6. 简单易用的Android 客户端
   1. 用户可以通过客户端报告事件
   2. 管理员可以向某事件周围用户推送事件验证请求
7. 性能
   1. 能够保证高并发访问

管理员（市场,执法机构）：

1. 添加事件类型
2. 批量修改事件类型，事件属性（是否被相关机构处理）
3. 对事件的增加（适用于执法机构），验证，删除，修改（优化）
   1. 当一个事件被认为虚假时，可以删除
   2. 执法机构（如果有），则可以手动添加历史事件记录
   3. 当某地区只有一个用户报告该事件时（此时事件验证算法无法工作），管理员可以通过验证功能来显示/隐藏该事件，或者是推送事件验证信息到临近用户，请求验证
   4. 当事件的街道信息有误，或者描述不规范时，管理员可以修改（优化）
4. 向指定Android客户端推送事件验证消息

## 项目目标

1. 系统安全：为了防止系统遭遇攻击，使用随机密钥+时间戳的形式进行鉴权
2. 通过安卓客户端，网页进行事件上报
3. 通过网页查看周边事件信息
4. 用户可以根据条件（街道名，事件类型，发生时间，事件是否被相关部门处理）查询及筛选信息
5. 用户可以标注短时间内发布的信息的可信度（投票的形式）
6. 用户可以订阅指定类型的事件警报
7. 查看指定地区某种事件在某段时间的发生情况
8. 系统可以自动优化事件（合并不同用户上报的同一事件，给予可信度低的事件进行降权）
9. 不仅仅是违法事件，还包括商家的欺诈行为和黑心商家出没地的标注
10. 采用不同图标在地图上标识不同的事件
11. 将违法事件归类显示某段日期之间的同类型事件热度图
12. 项目最终运行再Ubuntu 服务器上

## 系统特点

1. 拥有移动客户端，除了在网页上进行事件报告外，用户也可以通过移动客户端快速及时的进行事件报告
2. 提供二级管理员，该账户一般给予执法机构，用于添加记录在案的事件信息
3. 用户事件真伪验证，管理员可以针对某一个事件向周围的用户发送事件验证请求
4. 用户可以标注未经验证的事件的可信度
5. 本系统内置算法将可以自动优化事件，如合并重复事件，优化事件地理位置信息等
6. 本系统事件不仅仅局限于违法事件，同时具有中国特色的黑心商家，假冒伪劣产品出售店铺，宰客事件均可上报。

## 条件、假设与限制

### 限制

本项目还存在以下约束：

1. 系统需要能够长时间的高并发稳定运行
2. 进行系统各模块划分，模块任务分配，接口说明相关文档的撰写，需要时间：1周
3. 法律和政策方面：公安机关可能以“担心犯罪地图可能被犯罪分子利用，造成安全防范薄弱区域案件高发，进而影响市民的安全感”为由否决本项目
4. 使用第三方地图API接口，腾讯地图API
5. 硬件，软件，运行环境和开发环境方面的条件和限制：4台学生笔记本，Windows10 操作系统， Java ， JavaScript 编程语言。 MySQL数据库服务器，Ubuntu服务器，NetBeans作为IDE。
6. 可利用的信息和资源：老师提供的材料，网上搜索资料，课堂教材，图书馆藏书
7. 系统投入使用的最晚时间：本学期期末（2017年12月中旬）

### 解决方案

为了完成本文所述的软件硬件功能，项目组成员应该在前期编写相关分析文档阶段就进行地图API的学习，熟悉开发工具，方便后期编码阶段迅速上手。另外，项目组也应该及早进行编码，在编写文档的过程中就进行相关模块的开发，避免编码阶段过于赶进度。

项目组内各成员应该及时沟通，提出问题，并讨论解决，及时解决技术难题，推动项目发展。同时项目组每个人写的代码应该进行交叉检查，确保代码风格符合要求，简化后期维护成本，及时发现潜在漏洞。

充分遵循该解决方案，可以在保证项目质量的前提下做到项目如期完成。

## 评价尺度

该系统预期使用对象主要是管理员，社会普通群众，因此，评价系统应该从以下几个方面出发：

1. 首先是系统功能完备性，系统应该满足各类使用人员的基本使用需求。
2. 其次是系统使用的难易程度，要方便所有用户使用，特别是对于一个第一次使用本系统的人，他应该能够快速上手，不需要花费太多的功夫找某个功能在哪里，如何使用
3. 维护成本（时间，人员，经费）不能过高，所以系统实现中应该要有完备的容错机制和代码注释，代码实现一切从易于维护的角度入手，不写生涩难懂的代码
4. 项目的开发时间进度需要保证一直处于计划之中，在学期内完成项目并测试

# 业务与技术可行性

## 对所建系统的说明

1. 项目设计、开发、编写在windows10操作系统下进行；
2. 使用4台学生笔记本；
3. 采用C/S模式提高易用性，交互性和安全性；

## 业务可行性与评估

综合考虑软件设计的完备性，从业务逻辑方面，通过分析事件处理的业务流程，从用户和管理员不同角度对本项目的业务可行性进行评估如下：

1. 用户角度

用户通过软件地图，清晰直观的了解区域所发生的各类事件，通过不同图标、颜色、文字高亮等区分事件。通过缩放地图、点击事件图标、搜索指定事件等方式，了解到事件的具体信息。整个软件操作简单、界面简单明了，从易用、易上手的角度是可行的。

1. 软件推出网页端和移动端，用户可以通过移动端进行及时、有效的事件报告。对于订阅事件警报的用户，能够通过邮箱及时的接收指定区域的最新的事件发生情况，对于突发事件可以起到及时了解，做好应对准备。所以，从事件的及时性考虑是可行的。
2. 用户可以对事件进行按条件的详细筛选、可靠度标记，系统对于同类事件、高频事件等也给出统计分析，方便用户查看。对于用户报告的事件，系统会向事件发生地临近用户推送事件验证请求，结合系统内置的判断算法再加上附近用户的事件验证结果来确认上报事件的可靠性。执法机关、官方机构也可以通过系统提供的接口，发布记录在案的历史事件，供用户查阅。对此，在事件真实性方面是可行的。
3. 通过此系统，执法机关可以对事件频发的区域加强管理。人名群众参与到打击犯罪等违法，不公平的事件中，对于预防、阻止违法行为发生起到良好的作用，因此，从社会有用性，维护社会治安等方面来看是可行的。
4. 管理员角度
5. 将事件进行分类，给每个事件贴上对应的标签，事件信息完备。用户上报事件后，系统对用户上报的事件信息进行格式化处理，确保信息完整且格式符合要求，且及时发布并通知附近的用户进行验证反馈，管理员对于这些事件信息，结合系统内置算法做再次校验处理，确保事件的真实性。这样来保证信息的正确是可行的。
6. 管理员可以对事件进行批量修改，对于消息推送也可以进行范围性的操作。此外系统对于事件会有预处理，进行相应的输入信息格式化等。从系统的高效方面来说是可行的。

## 技术可行性与评估

考虑到软件设计在技术方面的构想方面的局限性和优越性，综合各种因素，需要对软件在设计技术方面的各个方面的具体的可行性进行具体的分析，下面将从多个角度来具体阐述Incident Reports的技术可行性：

1. 软件采用C/S模式开发，具有系统信息安全性高，系统易用性高的特点。并通过将任务合理分配到Client端和Server端，降低了系统的通讯开销，可以充分利用两端硬件环境的优势。从安全性和交互性考虑是可行的。
2. 系统对用户上报的事件信息，通过算法进行输入信息格式化，并且进行一定程度的筛选、过滤。对于同一邻近地区报告量大的事件，算法自动进行分析处理，初步判别事件真实性并优化事件信息（归并事件），并及时标记在地图上方便其他用户查阅，这样能够避免系统管理员过于繁忙导致的事件不能及时处理。算法的实现仅为软件层面，通过计算大量提交的事件报告的地理位置差和事件描述/类型来确定多个事件是否为同类型事件，最后再归并这些同类型事件，该算法易于实现，所以是可行的。
3. 系统对于各类事件、各区域事件发生数量等进行统计分析，将结果反馈给用户，用户对某地区或某类事件有相应的认识，例如想去某地区旅游，这将是一个参考标准。这点在技术上不难实现，是可行的。
4. 系统具有高并发的特性（接口层），无论是网页还是移动客户端都会通过接口层和数据库交互，在接口层我们将每一个请求单独建立一个线程处理（并设置较短的超时时间），从而实现高并发，保证系统的稳定性，由于实现高并发我们只做软件层面上的处理（建立新线程处理请求），所以这一点是易于实现的。

# 成本模型分析与估算

## 估算方法简介

用例分析技术为软件需求规格化提供了一个基本的元素，而且该元素是可验证、可度量的用例通常由评估业务领域的战略目标和场景组成，它们也用来说明软件应用的复杂度。用例用来描绘一个系统外在可见的需求情况，是代表系统中各个项目相关人员（风险承担人，Stakeholder）之间就系统的行为所达成的契约用例可以作为项目计划、进度控制、测试等环节的基础。而且用例还可以使开发团队与客户之间的交流更加顺畅。

### 用例数估算前提条件

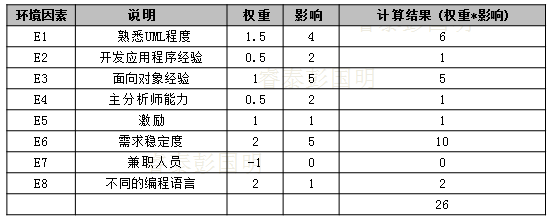
通过分析用例的角色和场景，可以得到可靠的一个应用程序的规模和工作量的估算值。用例建模是一种得到广泛使用的技术，用来获取软件应用的业务过程和需求。用例模型的建立帮助划定项目的功能边界及功能结构，它为实现自上而下的估算提供基础。用例点数是一个项目估算方法，它使用项目的用例来准确的估算项目规模和工作量。

### 用例数估算步骤

1. 角色复杂度等级划分及计数
2. 用例复杂度等级划分及计数
3. 计算未平衡用例点数
4. 使用技术复杂度因子和环境复杂度因子平衡(如图1，2)
5. 估算项目规模和开发工作量
6. 角色复杂度等级划分及计数
7. 用例复杂度等级划分及计数
8. 计算未平衡用例点数（如公式1）
9. 将UAW与UUCW相加得出未平衡用例点数（ Unadjusted Use Case Point ，UUCP）,UUCP=UAW+UUCW

图1 使用技术复杂度因子平衡

[](http://photo.blog.sina.com.cn/showpic.html#blogid=93c2f6700101384z&url=http://s2.sinaimg.cn/orignal/93c2f670gc3e814effa01)  
 TCF = 0.6 + (.01\*Total Factor). 图1中, TCF = 1.07

图2 使用环境复杂度因子平衡  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=93c2f6700101384z&url=http://s2.sinaimg.cn/orignal/93c2f670g79fd9bb09671)  
 ECF = 1.4 + (-0.03\*Total Factor). ECF = 0.62

公式1 软件规模用例点数计算公式如下：

  软件规模举例：UCP=TCF\*ECF\*UUCP

### 总结

如果没有历史数据可供参考，行业专家推荐使用15-30之间的一个数字，典型的，你可以使用20人时/UCP。

与专家估算法相比，用例点数估算可能偏高。为了得到准确结果，公式中的变量需要进行调整，特别是刚开始这样估算时。

软件功能分解（WBS）对估算的有效性有很大影响，应以3至5层之间为宜。

场景中的步数。很多的步数将使复杂性偏高，用例点数增大。较少的步数将使复杂性偏低。

包含和扩充用例增大了复杂性。将这些记为单独的用例。

用例中的角色数也影响估算。如果可能，将角色归纳为更高层次的角色。这将减少复杂性而不影响用例。

技术和环境因素值需要根据不断获得的实际数据来做调整。越多项目使用用例点数来估算，估算将越准确。

生产力因素只能从历史资产中获得。在没有得到符合公司实际的生产力基准数据前，将生产力划为3个等级：高级工程师15人时/UCP；中级工程师20人时/UCP；初级工程师30人时/UCP；

## 成本估算实施步骤

### Actor的权重计算

Actor权重参考表

|  |  |
| --- | --- |
| 接口类型 | Actor权重 |
| 类DOS型界面 | 0.8 |
| 简单的对话型界面 | 1.6 |
| 复杂的对话型界面 | 2.3 |
| 简单的GUI界面 | 2.4 |
| 复杂的GUI界面 | 3.0 |

Actor的权重合计表

|  |  |  |
| --- | --- | --- |
| Actor | Actor权重 | 理由 |
| 用户 | 3.0 | 复杂GUI界面 |
| 管理员 | 3.0 | 复杂GUI界面 |
| 二级管理员 | 2.4 | 简单的GUI界面 |
| 合计 | 8.4 |  |

### UseCase权重的计算

UseCase权重参考表

|  |  |  |
| --- | --- | --- |
| 类型 | 意义 | 权重（系数） |
| 简单型 | 3个以下事务/4个以下分析类 | 4～7 |
| 平均型 | 4至7个事务/5至10个分析类 | 8～12 |
| 复杂型 | 8个以上事务/11个以上分析类 | 13～17 |

UseCase权重合计表

|  |  |  |
| --- | --- | --- |
| UseCase名 | 权重 | 理由 |
| 查看事件地图  事件搜素 | 10  5 | 平均型  简单型 |
| 浏览事件详细信息 | 4 | 简单型 |
| 事件可信度标记  事件上报  订阅  事件类型管理  事件管理  推送事件  管理自己上传的数据  提供报案信息 | 4  7  4  8  12  4  8  4 | 简单型  简单型  简单型  平均型  平均型  简单型  简单型  简单型 |
| 合计 | 70 |  |

### UUCP计算(UUCP：Unadjusted Use Case Point)

UUCP＝∑Actor权重+∑UseCase权重＝78.4

### 技术复杂度（TCF）计算

将待开发系统所涉及的一些技术因素系数根据涉及程度赋予从0到6之间的值(0：未涉及； 3：一般性；5：本质性)。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 类目 | 权重 | 系数 | 值 | 理由 |
| T1 | 分布式系统 | 0 | 1 |  |  |
| T2 | 系统吞吐量大、处理时间短 | 1 | 1 |  |  |
| T3 | 在线用户使用效率高 | 0 | 1 |  |  |
| T4 | 内部处理复杂 | 2 | 1 |  |  |
| T5 | 代码可供后续项目复用 | 3 | 1 |  |  |
| T6 | 易安装调试 | 1 | 1 |  |  |
| T7 | 易用性高 | 1 | 1 |  |  |
| T8 | 移植性高 | 1 | 1 |  |  |
| T9 | 易扩展和修改 | 5 | 1 |  |  |
| T10 | 并行处理 | 3 | 1 |  |  |
| T11 | 需要特别考虑系统的安全性 | 3 | 1 |  |  |
| T12 | 第三方软件能够直接访问 | 0 | 1 |  |  |
| T13 | 需要对用户进行特别培训 | 0 | 1 |  |  |
| 合计 | TFactor=∑权重×系数 | 20 |  |  |  |

由上表得到TFactor的值，根据下面的算式得出该项目技术复杂度的权重

TCF(Technical Complexity Factor)＝0.6 + (0.01×TFactor) ＝0.8

### 环境复杂度（EF）权重的计算

根据项目开发的具体环境，给下列环境因素的系数赋予0-6之间的值

E1-E4   0 无经验 3 一般 5 专家级

E5     0 无热情 5 高度热情

E6     0 极端不安定 3 平均 5 不变

E7     0 无复用    5 高度复用（复用度80%以上）

E8     0 效率低下  2 一般 5 开发高度自动化

E9     0 公司内/全职开发 5 外包/兼职开发

E10    1 简单开发语言 3 平均 5 非常难的开发语言

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 类目 | 权重 | 系数 | 值 | 理由 |
| E1 | 开发人员对开发流程的熟悉度 | 1 | 1 |  |  |
| E2 | 开发人员对该类应用的开发经验 | 0.5 | 1 |  |  |
| E3 | 开发人员对开发方法的习惯程度 | 0.5 | 1 |  |  |
| E4 | 项目经理的能力 | 3 | 1 |  |  |
| E5 | 开发人员对该项目的热情度 | 3 | 1 |  |  |
| E6 | 开发式样的稳定性（很少变更） | 3 | 1 |  |  |
| E7 | 该项目的复用程度 | 2 | 1 |  |  |
| E8 | 采用开发工具的开发效率 | 1 | 1 |  |  |
| E9 | 开发人员的来源及工作方式 | -0 | 1 |  |  |
| E10 | 程序设计语言的难度 | -2 | 1 |  |  |
| 合计 | EFactor=∑权重×系数 | 12 |  |  |  |

由上表得到EFactor的值，根据下面的算式得出该项目技术复杂度的权重

EF(Environmental Factor)＝1.6 - (0.03×EFactor) ＝1.24

### UCP(Use Case Point)的计算

UCP=UUCP×TCF×EF=78.4×0.8×1.24≈77.8

### 计算项目风险系数

（风险系数 = E1-E8中2以下的值与E9-E10中4以上的值的总和）

风险系数为3 → 1UCP = 26人时

### 估算工作量

1人月 = 20人日 = 80人时

|  |  |  |
| --- | --- | --- |
| **规模** | **77.8** | **UCP** |
| **生产率** | **6.5** | **人日/UCP** |
| **工作量** | **25** | **人月** |

### 估算总时间

Time（月）= C\*工作量^P

C为进度调整系数（取3），P为组织内部开发能力调整系数（取0.4）

Time = 3\*25^0.4 = 11月

# 项目风险可行性研究

## 风险管理方法

1. 识别风险

从项目的具体情况出发，列举出可能出现的风险，弄清楚每一种可能的风险情况。

1. 定向风险分析

主观分析辅以现有模型对每个风险可能照成的影响进行分析，并结合其对项目的影响和发生概率给出风险大小的量值。最后根据该量值对风险进行排序，分离出高风险事件。

1. 规划对风险的应对

针对之前的风险分析结果，制定风险发生时的应对方案，降低风险对项目的威胁。实际操作中可通过提前开发相关原型的发放来提前应对潜在风险，降低风险。

1. 监控风险

在应对风险的时，监控应对计划的执行情况，跟踪已经识别的风险，监测残余风险，避免该风险再次对项目产生的影响。并且定期对风险进行重新评估，定期对整个项目风险进行重新认识。

## 风险预测与识别

### 定义阶段（可行性研究，需求分析）：

1. 项目开发过程中需求变更
2. 项目初期需求不明确
3. 产品规划的规模过大
4. 外部因素对需求分析的影响
5. 分析人员无类似项目经验，花费在设计的时间过长，且功能分析可能不完善

### 开发阶段（概要设计，详细设计，实现，测试）：

1. 代码编写的过程中缺少必要的编码规范，导致大量重复工作和开发失误
2. 开发人员不熟悉软件工具和开发环境（开发人员技术培训不足）导致项目进展缓慢
3. 参与开发/测试人员能力不足，导致部分功能实现过于复杂，性能得不到保障且难于后期维护
4. 实际开发过程中开发出了冗余，和系统无关的功能，导致项目开发周期被拉长
5. 编写过多原型测试，导致项目开发周期被拖长
6. 项目开发人员的流失
7. 开发进度不切实际
8. 性能需求无法达到系统规划要求
9. 在不熟悉或未经检验的软件和硬件环境中运行系统所产生的意料之外的问题，导致系统测试周期过长，无法按时完成；
10. 面临新技术（软件或硬件）的挑战；
11. 开发过程中缺乏质量跟踪，导致测试过程中系统出现过多问题，延长了测试周期
12. 开发出的成品和计划有一定的差别，达不到计划的理想运行状态

### 运行维护阶段（运行，维护）：

1. 运行过程中系统停止响应
2. 收到大量查询请求，系统瘫痪
3. 代码不规范，导致难于维护

## 风险评估与分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类别 | 潜在风险 | 风险发生概率 | 风险后果影响 | 风险 |
| 定义阶段 | 开发过程中需求变更 | 中 | 轻度 | 11 |
| 初期需求不明确 | 中 | 轻度 | 11 |
| 产品规划的规模过大 | 低 | 轻度 | 19 |
| 外部因素对需求分析的影响 | 高 | 中度 | 8 |
| 分析人员无相似经验，功能分析不完善 | 高 | 严重 | 5 |
| 开发阶段 | 缺少必要的编码规范 | 低 | 轻度 | 18 |
| 不熟悉软件工具和开发环境 | 低 | 轻度 | 18 |
| 开发人员能力不足 | 中 | 轻度 | 12 |
| 开发了冗余，和系统无关的功能 | 中 | 严重 | 8 |
| 编写过多原型测试 | 中 | 轻度 | 9 |
| 项目开发人员的流失 | 低 | 轻度 | 19 |
| 开发进度不切实际 | 中 | 严重 | 8 |
| 性能需求无法达到系统规划要求 | 中 | 严重 | 6 |
| 产生的意料之外的问题 | 中 | 轻度 | 9 |
| 面临新技术（软件或硬件）的挑战 | 低 | 轻度 | 13 |
| 开发过程中缺乏质量跟踪 | 低 | 中度 | 9 |
| 开发出的成品和计划有一定的差别 | 低 | 中度 | 9 |
| 运行维护阶段 | 运行过程中系统停止响应 | 高 | 严重 | 5 |
| 收到大量查询请求，系统瘫痪 | 中 | 严重 | 6 |
| 代码不规范，导致难于维护 | 中 | 严重 | 6 |

（1-5 是不能接受的风险；6-9 是不希望有的风险；10-17 是有控制的接受的风险； 18-20 是不经评审即可接受的风险）

## 风险及规避措施

1. 软件/设备方面
2. 统一项目设计、开发、编写（要求在同一软件工具和环境）在windows10操作系统下进行
3. 用户
4. 在项目开发前期确定客户的需求，并对自身的需求有明确的定位和对扩充条件的管理机制
5. 技术
   1. 开发人员的水平不一，针对不同困难，互相帮助互相检查
   2. 时时关注项目所需的技术更新情况
6. 人力
7. 在项目经理的带领下合理分配人力资源
8. 管理
9. 定期开展项目开发进度讨论会（了解每个组员的进度，检查解决问题）
10. 规定代码规范，方便最后的代码统一
11. 对类似项目的开发管理过程进行参考
12. 出现风险错误等适当时缓解风险，从而降低对实现项目目标的不利影响。
13. 写开发日志

**项目可能存在的风险以及规避方法：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 风险意识 | | 风险应对措施 | |
|  | 潜在风险事件 | 风险发生后果 | 应急措施 | 预防措施 |
| 定义阶段 | 项目开发过程中需求变更 | 工期延误 | 追加资源 | 加班 |
| 项目初期需求不明确 | 工期延误 | 提交讨论 | 讨论 |
| 产品规划的规模过大 | 工程延误 | 删减不必要 | 根据时间需求制定开发计划 |
| 外部因素对需求分析的影响 | 项目进度拖延 | 根据需求进行调整 | 建立范围变更程序 |
| 分析人员无类似项目经验，花费在设计的时间过长，且功能分析可能不完善 | 项目延期，影响到员工能力的发挥 | 换人 | 学习类似案例，提高分析效率 |
| 开发阶段 | 缺少必要的编码规范 | 质量问题，成员间沟通不便 | 开会修改 | 统一环境和编码规范 |
| 不熟悉软件工具和开发环境 | 代码质量问题，工期缓慢 | 培训 | 提前学习 |
| 能力不足 | 工期缓慢 | 培训 | 提前学习 |
| 开发出了冗余，和系统无关的功能 | 不能满足需求 | 修改设计 | 编码之前进行相关的审议 |
| 编写过多原型测试 | 不能满足客户需求，工期延误 | 减少没必要的测试 | 开会讨论所需原型 |
| 项目开发人员的流失 | 工期延误 | 加班补进度 | 找人找替补 |
| 开发进度不切实际 | 项目脱期 | 加班加点 | 制定工作计划 |
| 性能需求无法达到系统规划要求 | 达不到预期客户目标 | 追加功能 | 制定计划，审核要求 |
| 实际开发过程中开发出了冗余，产生的意料之外的问题 | 工期延误 | 修改项目 | 时时审核自己的代码 |
| 面临新技术（软件或硬件）的挑战 | 项目更新，耽误时间 | 加班加点 | 学习 |
| 开发过程中缺乏质量跟踪 | 质量问题 | 及时的解决 | 实现评审测试计划 |
| 开发出的成品和计划有一定的差别 | 达不到客户需求 | 及时的解决 | 制定合理开发计划 |
| 运行维护阶段 | 运行过程中系统停止响应 | 影响用户体验 | 工作人员维护 | 日常监测 |
| 收到大量查询请求，系统瘫痪 | 不能满足用户的需求，数据溢出 | 将现有数据备份 | 应用较大的数据库 |
| 代码不规范，导致难于维护 | 质量问题 | 修改设计 | 审核开发计划 |

# 可行性分析结论

通过对项目整体进行可行性分析，得出该项目无论在操作可行性、技术可行性、经济可行性及社会可行性上均满足要求，因此，开发此系统的构想是可行的，可着手进行开发。