目录

[1.风险分类 1](#_Toc69658433)

[1.1需求风险 1](#_Toc69658434)

[1.2计划编制风险 1](#_Toc69658435)

[1.3组织和管理风险 1](#_Toc69658436)

[1.41人员风险 1](#_Toc69658437)

[1.5开发环境风险 1](#_Toc69658438)

[1.6产品风险 2](#_Toc69658439)

[1.7设计和实现风险 2](#_Toc69658440)

[2.风险辨识 2](#_Toc69658441)

[3.风险管理模型 3](#_Toc69658442)

[3.1Boehm模型 3](#_Toc69658443)

[3.2CRM模型 3](#_Toc69658444)

[3.3Leavitt模型 4](#_Toc69658445)

# 1.风险分类

## 1.1需求风险

①需求已经成为项目基准,但需求还在继续变化;②需求定义欠佳,而进一步的定义会扩展项目范畴;③添加额外的需求;④产品定义含混的部分比预期需要更多的时间;⑤在做需求中客户参与不够;⑥缺少有效的需求变化管理过程。

## 1.2计划编制风险

①计划、资源和产品定义全凭客户或上层领导口头指令,并且不完全一致;②计划是优化的,是"最佳状态",但计划不现实,只能算是"期望状态";③计划基于使用特定的小组成员,而那个特定的小组成员其实指望不上;④产品规模(代码行数、功能点、与前一产品规模的百分比)比估计的要大;⑤完成目标日期提前,但没有相应地调整产品范围或可用资源;⑥涉足不熟悉的产品领域,花费在设计和实现上的时间比预期的要多。

## 1.3组织和管理风险

①仅由管理层或市场人员进行技术决策,导致计划进度缓慢,计划时间延长;②低效的项目组结构降低生产率;③管理层审查 决策的周期比预期的时间长;④预算削减,打乱项目计划;⑤管理层作出了打击项目组织积极性的决定;⑥缺乏必要的规范,导至工作失误与重复工作;⑦非技术的第三方的工作(预算批准、设备采购批准、法律方面的审查、安全保证等)时间比预期的延长。

## 1.41人员风险

①作为先决条件的任务(如培训及其他项目)不能按时完成;②开发人员和管理层之间关系不佳,导致决策缓慢,影响全局;③缺乏激励措施,士气低下,降低了生产能力;④某些人员需要更多的时间适应还不熟悉的软件工具和环境;⑤项目后期加入新的开发人员,需进行培训并逐渐与现有成员沟通,从而使现有成员的工作效率降低;⑥由于项目组成员之间发生冲突,导致沟通不畅、设计欠佳、接口出现错误和额外的重复工作;⑦不适应工作的成员没有调离项目组,影响了项目组其他成员的积极性;⑧没有找到项目急需的具有特定技能的人。

## 1.5开发环境风险

①设施未及时到位;②设施虽到位,但不配套,如没有电话、网线、办公用品等;③设施拥挤、杂乱或者破损;④开发工具未及时到位;⑤开发工具不如期望的那样有效,开发人员需要时间创建工作环境或者切换新的工具;⑥新的开发工具的学习期比预期的长,内容繁多。

客户风险

①客户对于最后交付的产品不满意,要求重新设计和重做;②客户的意见未被采纳,造成产品最终无法满足用户要求,因而必须重做;③客户对规划、原型和规格的审核 决策周期比预期的要长;④客户没有或不能参与规划、原型和规格阶段的审核,导致需求不稳定和产品生产周期的变更;⑤客户答复的时间(如回答或澄清与需求相关问题的时间)比预期长;⑥客户提供的组件质量欠佳,导致额外的测试、设计和集成工作,以及额外的客户关系管理工作。

## 1.6产品风险

①矫正质量低下的不可接受的产品,需要比预期更多的测试、设计和实现工作;②开发额外的不需要的功能(镀金),延长了计划进度;③严格要求与现有系统兼容,需要进行比预期更多的测试、设计和实现工作;④要求与其他系统或不受本项目组控制的系统相连,导致无法预料的设计、实现和测试工作;⑤在不熟悉或未经检验的软件和硬件环境中运行所产生的未预料到的问题;⑥开发一种全新的模块将比预期花费更长的时间;⑦依赖正在开发中的技术将延长计划进度。

## 1.7设计和实现风险

①设计质量低下,导致重复设计;②一些必要的功能无法使用现有的代码和库实现,开发人员必须使用新的库或者自行开发新的功能;③代码和库质量低下,导致需要进行额外的测试,修正错误,或重新制作;④过高估计了增强型工具对计划进度的节省量;⑤分别开发的模块无法有效集成,需要重新设计或制作。

过程风险

①大量的纸面工作导致进程比预期的慢;②前期的质量保证行为不真实,导致后期的重复工作;③太不正规(缺乏对软件开发策略和标准的遵循),导致沟通不足,质量欠佳,甚至需重新开发;④过于正规(教条地坚持软件开发策略和标准),导致过多耗时于无用的工作;⑤向管理层撰写进程报告占用开发人员的时间比预期的多;⑥风险管理粗心,导致未能发现重大的项目风险。

# 2.风险辨识

识别风险是系统化地识别已知的和可预测的风险，在可能时避免这些风险，且当必要时控制这些风险。根据风险内容，我们可以将风险分为：

1. 产品规模风险：与软件的总体规模相关的风险。
2. 商业影响风险：商业风险影响到软件开发的生存能力。商业风险包含的五个主要的风险是：

* 市场风险：开发了一个没有人真正需要的优秀产品或系统；
* 策略风险：开发的产品不符合公司的整体商业策略；
* 销售风险：开发了一个销售部门不知道如何去卖的产品；
* 管理风险：由于重点的转移或人员的变动而失去了高级管理层的支持的风险；
* 预算风险：没有得到预算或人力上的保证。

1. 客户特性风险：与客户的素质以及开发者和客户沟通能力相关的风险。
2. 过程定义风险：与软件过程定义相关的风险。
3. 开发环境风险：与开发工具的可用性及质量相关的风险。
4. 技术风险：技术风险是指在设计、实现、接口、验证、维护、规约的二义性、技术的不确定性、陈旧的技术等方面存在的风险。技术风险威胁到软件开发的质量及交付的时间，如果技术风险变成现实，则开发工作可能变得很困难或根本不可能。

(7)人员数目及经验带来的风险：与参与工作的软件工程师的总体技术水平及项目经验相关的风险。

# 3.风险管理模型

## 3.1Boehm模型

Boehm用公式RE=P(UO)\*L(UO)对风险进行定义，其中RE表示风险或者风险所造成的影响，P(UO)表示令人不满意的结果所发生的概率，L(UO)表示糟糕的结果会产生的破坏性的程度。在风险管理步骤上，Boehm基本沿袭了传统的项目风险管理理论，指出风险管理由风险评估和风险控制两大部分组成，风险评估又可分为识别、分析、设置优先级3个子步骤，风险控制则包括制定管理计划、解决和监督风险3步。

Boehm思想的核心是10大风险因素列表，其中包括人员短缺、不合理的进度安排和预算、不断的需求变动等。针对每个风险因素，Boehm都给出了一系列的风险管理策略。在实际操作时，以10大风险列表为依据，总结当前项目具体的风险因素，评估后进行计划和实施，在下一次定期召开的会议上再对这10大风险因素的解决情况进行总结，产生新的10大风险因素表，依此类推。

10大风险列表的思想可以将管理层的注意力有效地集中在高风险、高权重、严重影响项目成功的关键因素上，而不需要考虑众多的低优先级的细节问题。而且，这个列表是通过对美国几个大型航空或国防系统软件项目的深入调查，编辑整理而成的，因此有一定的普遍性和实际性。但是它只是基于对风险因素集合的归纳，尚未有文章论述其具体的理论基础、原始数据及其归纳方法。另外，Boehm也没有清晰明确地说明风险管理模型到底要捕获哪些软件风险的特殊方面，因为列举的风险因素会随着多个风险管理方法而变动，同时也互相影响。这就意味着风险列表需要改进和扩充，管理步骤也需要优化。

虽然其理论存在一些不足，但Boehm毕竟可以说是软件项目风险管理的开山鼻祖。在其之后，更多的组织和个人开始了对风险管理的研究，软件项目风险管理的重要性日益得到认同。

## 3.2CRM模型

SEI（Software Engineering Institution）作为世界上著名的旨在改善软件工程管理实践的组织，也对风险管理投入了大量的热情。SEI提出了持续风险管理管理模型CRM（Continuous Risk Management）。

SEI的风险管理原则是：不断地评估可能造成恶劣后果的因素；决定最迫切需要处理的风险；实现控制风险的策略；评测并确保风险策略实施的有效性。

CRM模型要求在项目生命期的所有阶段都关注风险识别和管理，它将风险管理划分为5个步骤：风险识别、分析、计划、跟踪、控制。框架显示了应用CRM的基础活动及其之间的交互关系，强调了这是一个在项目开发过程中反复持续进行的活动序列。每个风险因素一般都需要按顺序经过这些活动，但是对不同风险因素开展的不同活动可以是并发的或者交替的。

## 3.3Leavitt模型

SEI和Boehm的模型都以风险管理的过程为主体，研究每个步骤所需的参考信息及其操作。而Aalborg大学提出的思路则是以Leavitt模型为基础，着重从导致软件开发风险的不同角度出发探讨风险管理。

1964年提出的Leavitt模型将形成各种系统的组织划分为4个有趣的组成部分：任务、结构、角色和技术。这4个组成部分和软件开发的各因素很好地对应起来：角色覆盖了所有的项目参与者，例如软件用户、项目经理和设计人员等；结构表示项目组织和其他制度上的安排；技术则包括开发工具、方法、硬件软件平台；任务描述了项目的目标和预期结果。Leavitt模型的关键思路是：模型的各个组成部分是密切相关的，一个组成部分的变化会影响其他的组成部分，如果一个组成部分的状态和其他的状态不一致，就会造成比较严重的后果，并可能降低整个系统的性能。

将这个模型和软件风险的概念相对应，即一个系统开发过程中任何Leavitt组成成分的修改都会产生一些问题，甚至导致软件修改的失败。根据Leavitt模型，任何导致风险发生的因素都可以归结为模型中的组成部分，例如技术及其可行性；或者归结为组成部分之间的联系，例如程序开发人员使用某一技术的能力。因此，使用Leavitt模型从4个方面分别识别和分析软件项目的风险是极有条理性和比较全面的。在进行软件项目管理时，可以采用不同的方法对不同的方面进行风险管理。

Leavitt模型实际上是提出一个框架，可以更加广泛和系统地将软件风险的相关信息组织起来。Leavitt理论的设计方法和实现研究已经广泛应用于信息系统中，它所考虑的都是软件风险管理中十分重要的环节，而且简单、定义良好、适用于分析风险管理步骤。