

## 2.2 软件项目开发管理

- 软件项目管理概念及特征

- 基本概念

- 项目

精心定义的一组活动，使用受约束的资源(资金、人、原料、能源、空间等)来满足预定义的目标

- 项目管理

有效的组织与管理各类资源(例如人)，以使项目能够在预定的范围、质量、时间和成本等约束条件下顺利交付(deliver)

- 特征

- 软件产品不可见性
    - 项目不确定性
    - 软件过程的多变化性
    - 软件人员的高技能及其高流动性

- 软件项目管理的 4P

- 软件人员 (People)

- 角色
      - 人员选择
      - 团队组织方式

- 主程序员式组织结构




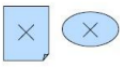

以主程序员为核心，主程序员既是项目管理者也是技术负责人，团队其他人员的职能进行专业化分工

- 优点：成员之间采取简单的交流沟通模式
          - 缺点：很难找到技术和管理才能兼备的主程序员

- 矩阵式组织结构

将技术与管理工作进行分离，技术负责人负责技术决策，管理负责人负责非技术性事务的管理决策和绩效评价

- 软件产品 (Product)

需求分析	软件设计	软件实现	软件测试	软件运行
				
<ul style="list-style-type: none"><li>• 用例模型</li><li>• 软件需求规格说明</li></ul>	<ul style="list-style-type: none"><li>• 软件体系结构描述</li><li>• 设计模型</li></ul>	<ul style="list-style-type: none"><li>• 源程序</li><li>• 目标代码</li><li>• 可执行构件</li></ul>	<ul style="list-style-type: none"><li>• 测试规程</li><li>• 测试用例</li></ul>	<ul style="list-style-type: none"><li>• 相关的运行时文件</li><li>• 用户手册</li></ul>
<div>开发管理文档</div> <div><div>计划文档<ul style="list-style-type: none"><li>- 工作分解结构</li><li>- 业务案例</li><li>- 发布规格说明</li><li>- 软件开发计划</li></ul></div><div>操作文档<ul style="list-style-type: none"><li>- 发布版本说明书</li><li>- 状态评估</li><li>- 软件变更申请</li><li>- 实施文档、环境</li></ul></div></div>				

- 软件过程（Process）
  - 选择软件过程模型
  - 基于过程框架指定初步的项目计划
  - 过程分解，指定完整计划，确定工作任务列表，任务对应产出物
- 项目（Project）
  - W5HH原则
  - 项目的基本要素
    - 结果
    - 进度表
    - 工作
    - 资源
  - 软件项目管理计划

软件项目管理计划是一个用来协调所有其他计划、以指导项目实施和控制的文件，它应该随着项目的进展和信息的补充进行定期完善。

引言	项目的目标、影响项目管理的各种约束条件
项目组织	开发团队的组织方式、人员构成与分工
风险分析	可能的风险以及发生的可能性、降低风险的策略
资源需求	项目所需的硬件与软件资源
工作分解	将项目分解成一系列的活动，指定项目里程碑和可交付的文档
项目进度	项目中各活动之间的依赖关系、完成每个里程碑预期需要的时间、在活动中的人员分配
监控和报告机制	需要提交的管理报告、提交时间以及项目监控机制

- 软件项目估算

项目估算是项目的规模、工作量、时间和成本等进行预算和估计的过程。

- 挑战
  - 项目的复杂性
  - 项目的不确定性
  - 没有可以参考的历史数据
- 估算内容
  - 规模估算
  - 工作量估算
  - 进度估算
  - 成本估算
- 估算的复杂不确定性
  - 估算的风险取决于资源、成本及进度的定量估算中的不确定性
- 估算方法
  - 分段估算
    - 从宏观估算开始，在项目执行中对各阶段的估算进行细化
    - 适用于最终该产品不可知或不确定性很大的项目
  - 专家判断

通过借鉴历史信息，专家提供项目估算所需的信息，或根据以往类似项目的经验，给出相关参数的估算上限

## ● 参数估算

通过对大量的项目历史数据进行统计分析，使用项目特性参数建立经验估算模型，估算诸如成本、预算和持续时间等活动参数

### ● 代码行技术 (LOC)

- $L = \frac{a+4m+b}{6}$ , L: 代码行数, a: 乐观值, b: 悲观值, m: 可能值
- $C = \mu \times L$ , L: 估计的代码行数,  $\mu$ : 每行代码的单位成本, c: 总成本
- $PM = \frac{L}{v}$ , PM: 总的工作量 (人月), v: 平均生产率

### ● 功能点技术方法

信息域加权因子:

信息域参数	加权因子			合计
	简单	中等	复杂	
外部输入	3	4	6	$\Sigma$
外部输出	4	5	7	$\Sigma$
外部查询	3	4	6	$\Sigma$
内部逻辑文件	7	10	15	$\Sigma$
外部逻辑文件	5	7	10	$\Sigma$
未调整功能点 UFC				$\Sigma$

系统复杂度调整值 Fi: 取值 0-5

F <sub>1</sub>	可靠的备份和恢复	F <sub>8</sub>	在线升级
F <sub>2</sub>	数据通信	F <sub>9</sub>	复杂的界面
F <sub>3</sub>	分布式处理	F <sub>10</sub>	复杂的数据处理
F <sub>4</sub>	性能	F <sub>11</sub>	代码复用性
F <sub>5</sub>	大量使用的配置	F <sub>12</sub>	安装简易性
F <sub>6</sub>	联机数据输入	F <sub>13</sub>	多重站点
F <sub>7</sub>	操作简单性	F <sub>14</sub>	易于修改

功能点计算:  $FP = UFC \times [0.65 + 0.01 \times \sum F_i]$

### ● 信息域

- 外部输入
- 外部输出
- 外部查询
- 内部逻辑文件
- 外部接口文件

- 估算公式  $FP = UFC \times [0.65 + 0.01 \times \sum F_i]$

### ● 代码行和功能点比较

- 代码行: 如果没哟开发, 无法准确的估算出来需要多少代码, 依赖于具体的开发语言
- 功能点: 开始可以估计到, 但是不容易估算, 人性化要求高, 用户体验要求高

### ● COCOMO 模型

结构性成本模型 COCOMO (COConstructive COst MOdel) 是一种利用经验模型进行成本估算的方法。

$$PM_{\text{nominal}} = A * (\text{Size})^B$$

- PMnominal: 人月工作量
- A: 工作量调整因子
- B: 规模调整因子
- Size: 规模, 单位是千行代码或功能点数

类型	A	B	说明
组织型	2.4	1.05	相对小的团队在一个高度熟悉的内部环境中开发规模较小, 接口需求较灵活的系统。
嵌入式	3.6	1.2	开发的产品在高度约束的条件下进行, 对系统改变的成本很高。
半独立型	3.0	1.12	介于上述两者中间

- 用例点估算

	基本流	扩展流	业务规则	折合标准用例
功能A	12	3	4	2.3
功能B	8	4	3	1.8
功能C	6	2	3	1.4
合计				5.5

标准用例 = (基本流 + 扩展流 + 2 × 业务规则) / 10

生产率 = 6 工作日 / 单位用例

工作量 = 6 × 5.5 = 33 工作日

- 故事点方法

- 故事点

- 理想日

- 基本做法：把一些常见“标准任务”给出一个“标准点数”，形成比较基线；估算时只要是同一类型任务，直接写故事点数而非天数

- 机器学习方法

- 人工神经网络

是采用一种学习方法导出一种预测模型，首先建立神经网络，再使用一组历史项目数据（样本数据）训练网络，训练后的网络可以用于估算新项目的工作量。

- 基于案例的推理方法

基于案例的推理方法可以用于基于类推的估算，即识别出与新项目类似的案例，再调整这些案例，使其适合新项目的参数。

- 项目进度安排

- 软件延期交付的原因

- 人员工作量关系

- 软件项目进度计划

- 工作量分配——将项目划分为多个活动、任务

40-20-40 法则，前期分析和设计、编码、测试

- 定义任务网络——确定任务项目依赖性

网络图

- 为任务安排工作时间段——时间分配

程序评估及评审技术（PERT）、关键路径方法（CPM）

- 标出任务最早开始时间与结束时间（ES）

- 标出任务最迟开始时间与结束时间（LF）

- 计算关键路径、

- 确定任务的开始/结束时间

- 绘制最终的任务进度安排（甘特图）

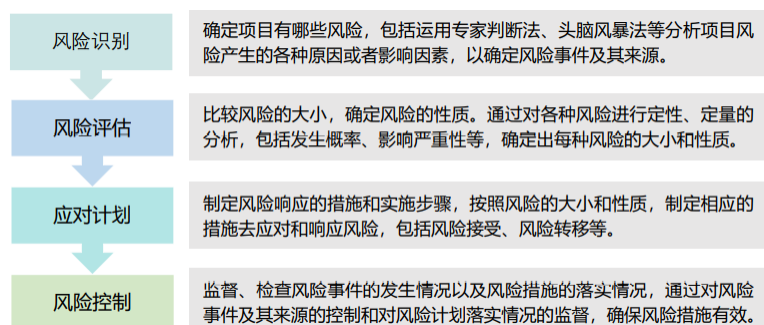
- 确认每个任务的资源需求——确认任务资源

人员资源分配图

- 确定责任——确认每个任务的负责人与参与人

- 明确结果——明确任务的输出结果
- 确定里程碑——确定任务与项目里程碑关系

## • 项目风险管理



## • 风险识别

### 风险识别

- 软件规模风险：
  - 估算准确程度？
  - 用户需求可能发生变化的频度与规模？
- 商业影响风险：
  - 交付期限？
  - 政府出台新政策？
- 客户相关风险：
  - 陌生客户？客户高层的重视程度？
  - 客户的配合程度？
- 软件过程风险：
  - 开发者不了解/不熟悉选定的过程模型？
  - 没有维护足够的文档？
- 开发环境风险：
  - 无法得到可用的工具？
  - 没有或不会使用工具？
- 开发技术风险：
  - 之前无该技术的经验？
  - 该技术难以实现某些需求？
- 开发人员风险：
  - 没有足够的经验与技能？
  - 某些人员会中途离开？

## • 风险预测：建立风险表

- 列出可能的风险以及风险类型
- 估计风险发生的可能性或概率
- 估计风险可能产生的影响或后果
- 确定风险缓解策略

## • 风险缓解

## • 风险监测

## • 风险管理及应急计划

## • 风险管理的七个原则

### 风险管理的7个原则

- 保持全面观点：考虑软件风险及软件所要解决的任务问题；
- 采用长远观点：考虑将来要发生的风险，并制定应急计划使将来发生的风险成为可管理的；
- 鼓励广泛交流：如果有人提出一个潜在的风险，要重视它；
- 结合：考虑风险时必须与软件过程相结合；
- 强调持续的过程：整个软件过程中，要持续保持警惕。随着信息量增加，要修改已识别的风险；随着知识的增加，要加入新的风险；
- 开发共享的产品：如果所有利益相关者共享相同版本的软件产品，有利于风险识别和评估；
- 鼓励协同工作：汇聚利益相关者的智慧、技能和知识；