软件工程：第十二章 软件项目管理

导学目标

- 掌握软件规模估算的两种方法
- 掌握工作量估算的两种模型
- 掌握开发时间的估计以及甘特图
- 掌握三种人员组织结构
- 掌握质量保证的定义及保证措施
- 掌握软件配置管理项以及过程

第一节 软件规模估算

1、软件项目管理的概念

通过计划、组织、控制一系列活动，合理配置使用资源，达到既定目标的活动

2、软件项目管理过程

从一组项目计划活动开始，而制定计划的基础是工作量和完成期限估算；为了估算项目的工作量和完成期限，首先需要估算软件的规模

3、软件规模的估算

常用方法是**代码行技术**和**功能点技术**

4、代码行技术

- 依据开发类似产品的经验和历史数据，估计实现一个功能所需要的源代码行数，把实现每个功能所需要的源代码行数进行累加，就可以得到实现整个软件所需要的源代码行数
- 当程序较小时，我们常用的单位是**代码行数(LOC**)
- 当程序较大时，我们常用的单位是**千行代码数(KLOC)**
- 具体方法
    1. 找 n 名有经验的工程师估计
        - a:程序的最小规模
        - b:程序的最大规模
        - m:程序的最可能的规模
    2. 求三种规模的平均值

$$\overline{a} = (a_1 + a_2 + \ldots a_n) / n$$

$$\overline{b} = (b_1 + b_2 + \ldots b_n) / n$$

$$\overline{m} = (m_1 + m_2 + \ldots m_n) / n$$

    3. 求程序的规模

$$L=(\overline{a}+4\overline{m}+\overline{b})/6$$

- 优点：
  - 代码是所有软件中都有的产品，很容易计算代码行数
- 缺点
  - 源程序规模不等于软件规模，源程序只是软件配置的一个成分
  - 用不同语言实现同一个软件的代码行数不同
  - 不适用于非过程语言

5、功能点技术

- 定义
  - 依据软件信息域特性和软件复杂性评估结果估算软件规模
- 信息域特性
  - 输入项数：用户向软件输入的项数，这些输入给软件提供面向应用的数据
  - 输出项数：软件向用户输出的项数，他们向用户提供面向应用的信息
  - 查询数：即是一种联机输入，以输出方式产生某种即时响应
  - 主文件数：每一个逻辑主文件都应计数
  - 外部接口数：机器可读的全部接口的数量，用这些接口把信息传递给另一个系统
- 估算功能点的步骤
  1. 计算**未调整功能点UFP**

信息域参数	计数		加权因数				加权计数
			简单	中间	复杂		
用户输入数		×	3	4	6	=	
用户输出数		×	4	5	7	=	
用户查询数		×	3	4	6	=	
文　件　数		×	7	10	15	=	
外部接口数		×	5	7	10	=	
总　计　数	→						

  2. 计算**复杂度因子TCF**

$$TCF=0.65+0.01\times DI$$

$DI=\sum_{i=1}^{14}Fi$　　DI:软件规模综合影响度

F_i(i=1到14)是复杂性校正值，通过回答下图问题确定。

评定每个因素的尺度是0—5：

0	1	2	3	4	5
没有影响	偶然的	适中的	普通的	重要的	极重要的

F_i：

1. 是同是否需要可靠的备份和回复？
2. 是否需要数据通信？
3. 是否有分布处理的功能？
4. 性能是否关键？
5. 系统是否运行在既存的高度实用化的操作环境中？
6. 系统是否需要联机数据项？
7. 联机数据项是否需要联机处理以建立多重窗口显示或操作？
8. 主文件是否联机更新？
9. 输入、输出、文件、查询是否复杂？
10. 内部处理过程是否复杂？
11. 程序代码是否被设计成可复用的？
12. 设计中是否包括转换和安装？
13. 系统是否被设计成可重复安装在不同机构中？
14. 应用是否被设计成便于修改和易于用户使用？

3. 计算**功能点数FP**

FP = UFP×TCF

- 功能点数所用编程语言无关，看起来功能点技术比代码行技术更合理一些，但是在判断信息域特性复杂级别和技术因素的影响程度时，存在着相当大的主观因素

第二节 工作量估算

工作量是软件规模的函数，单位是人月

支持大多数估算模型的经验数据，都是从有限个项目的样本集中总结出来的，因此，没有一个估算模型可以适用于所有类型的软件和开发环境

1、静态单变量模型

这类模型的总体结构形式如下：

$$E = A + B \times (ev)^C$$

其中，A、B 和 C 是由经验数据导出的常数，E 是以人月为单位的工作量，ev 是估算变量（KLOC 或 FP）。下面给出几个典型的静态单变量模型。

1. **面向KLOC的估算模型**

（1）Walston_Felix 模型

$$E=5.2\times(\mathrm{KLOC})^{0.91}$$

（2）Bailey_Basili 模型

$$E=5.5+0.73\times(\mathrm{KLOC})^{1.16}$$

（3）Boehm 简单模型

$$E=3.2\times(\mathrm{KLOC})^{1.05}$$

（4）Doty 模型（在 KLOC>9 时适用）

$$E=5.288\times(\mathrm{KLOC})^{1.047}$$

2. **面向FP的估算模型**

（1）Albrecht & Gaffney 模型

$$E=-13.39+0.0545\mathrm{FP}$$

（2）Maston，Barnett 和 Mellichamp 模型

$$E=585.7+15.12\mathrm{FP}$$

从上面的模型中可以看出，相同的KLOC或FP值，用不同模型估算得出不同的结果

主要原因是：这些模型多数都是仅根据若干应用领域中有限个项目的经验数据推导出来的，适用范围有限。因此，必须根据当前项目的特点选择适用的估算模型，并且根据需要适当地调整(例如，修改模型常数)估算模型

2、动态多变量模型

- 工作量是软件规模和开发时间两个变量的函数。是根据从4000多个当代软件项目中收集的生产率数据推导出来的

$$E=(\mathrm{LOC}\times B^{0.333}/P)^3\times(1/t)^4$$

- t：以月或年为单位的项目持续时间；
- B：特殊技术因子，随着需求增加缓慢增加。小程序0.16(5～10KLOC)，大程序(超70KLOC)0.39。
- P：生产率参数，反应过程管理、使用语言、系统的复杂程度等对工作量的影响实时嵌入软件2000；系统软件10000；商业系统28000等

第三节 进度计划

软件项目的进度安排是这样的一组活动：它通过把工作量分配给特定的软件工程任务并规定完成各项任务的起止时间，从而估算出项目的工作量分布于计划好的项目持续期内。进度计划将随着时间的流逝而不断演化

1、估算开发时间

- 工作量估算完，估算开发时间。如工作量为20人月项目，可能是下列几种进度表：
  - 1个人用20个月
  - 4个人用5个月
  - 20个人用1个月等
- 产生问题：进度表不符合实际，软件开发的时间和从事开发的工作人数不是简单的反比关系

2、估算开发时间模型

（1）Walston_Felix 模型

$$T = 2.5E^{0.35}$$

（2）原始的 COCOMO 模型

$$T = 2.5E^{0.38}$$

（3）COCOMO2 模型

$$T = 3.0E^{0.33+0.2\times(b-1.01)}$$

（4）Putnam 模型

$$T = 2.4E^{1/3}$$

其中：

E 是开发工作量（以人月为单位）；

T 是开发时间（以月为单位）。

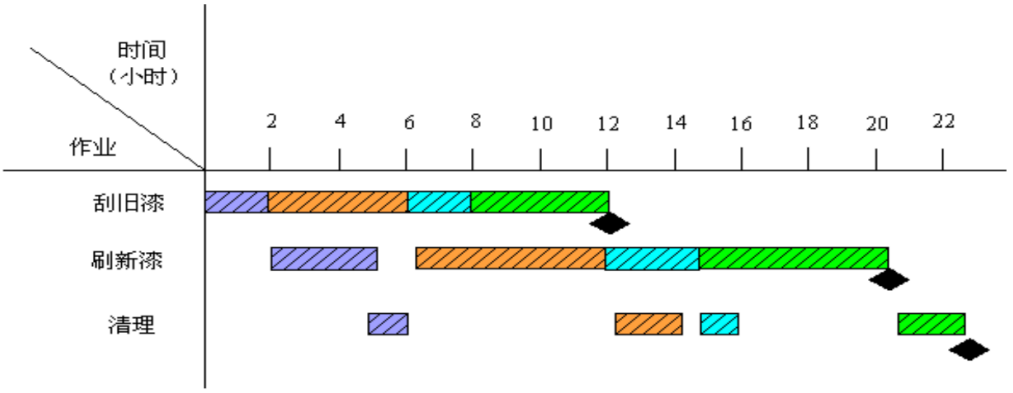
3、甘特图(Gantt 图)

- 示例
  - 矩形木板房需重新油漆。
  - 三步：刮旧漆，刷新漆，清除溅在窗上油漆。
  - 15 名工人，5把刮旧漆刮板，5 把刷漆刷子，5 把清除溅在窗上油漆小刮刀
- 图

各道工序估计需用的时间（小时）

墙壁 \ 工序	刮旧漆	刷新漆	清理
1或3	2	3	1
2或4	4	6	2

甘特图表达：（加菱形代表里程碑）

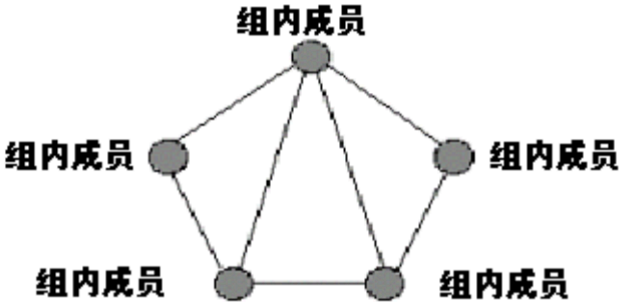


第四节 人员组织

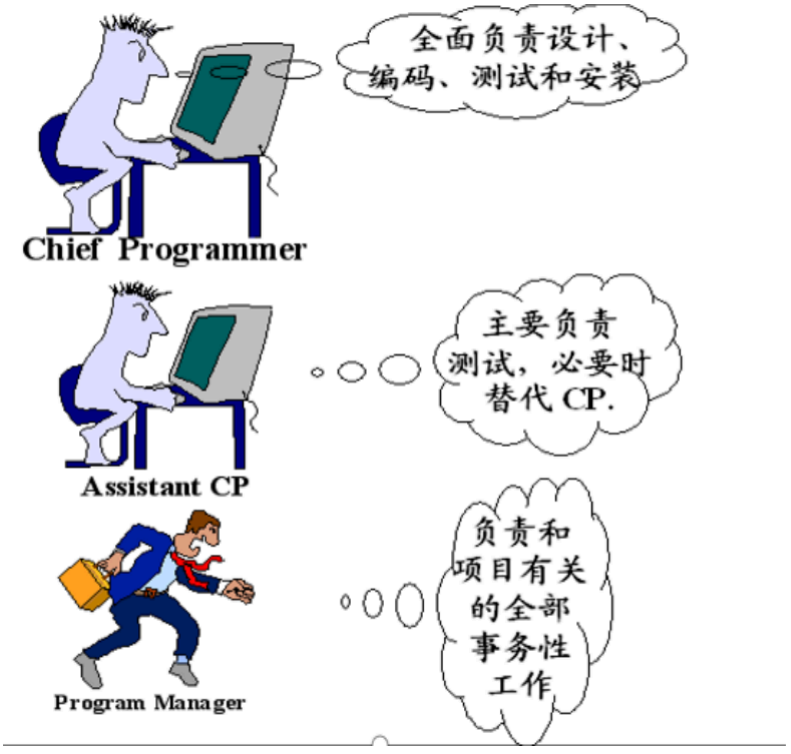
必须把多名开发人员合理的组织起来，使他们分工协作完成开发工作

3 种典型的组织方式

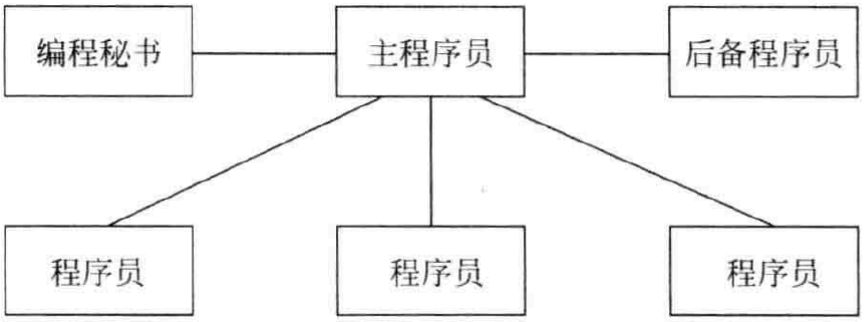
1. **民主制程序员组**
    - 组内成员完全平等，通过协商作出技术决策
    - 优点：积极性比较高、学习气氛比较浓郁
    - 缺点：没有权威指导，缺乏必要的协调
    - 适用领域：开发时间长，开发难度大的项目
    - 通信链路多，组内成员要少而精
    - 如果有n个成员，通信信道共n*(n-1)/2



2. **主程序员组**

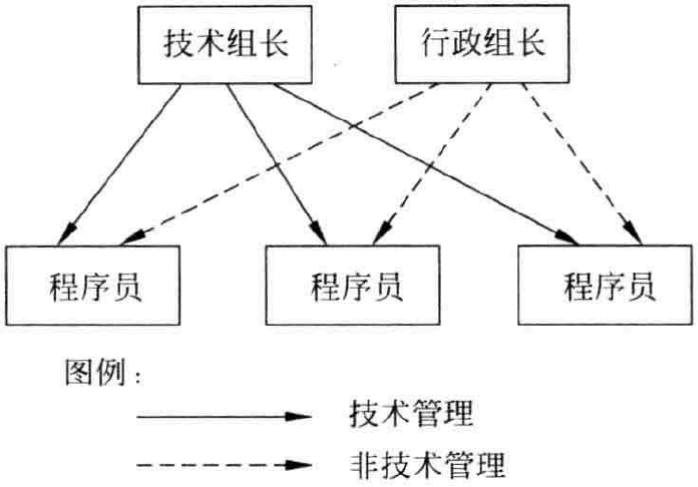


    - 组织形式

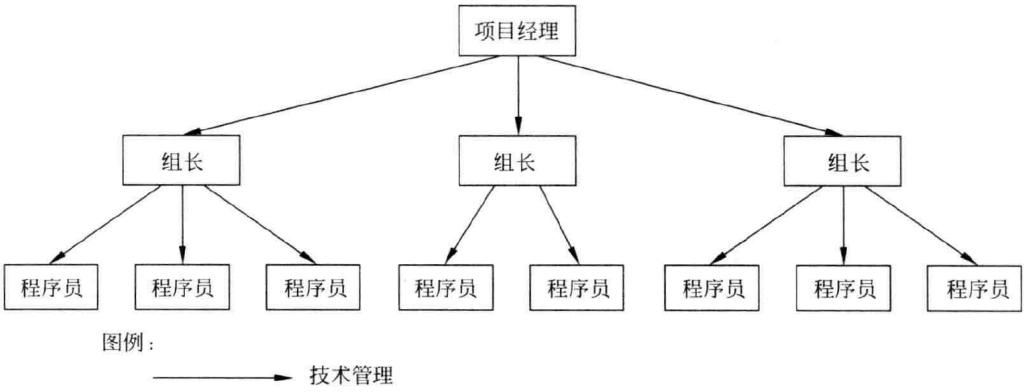


- 两个重要特征
  - 专业化：每名成员完成受过专业训练的工作
  - 层次化：主程序员有绝对权威

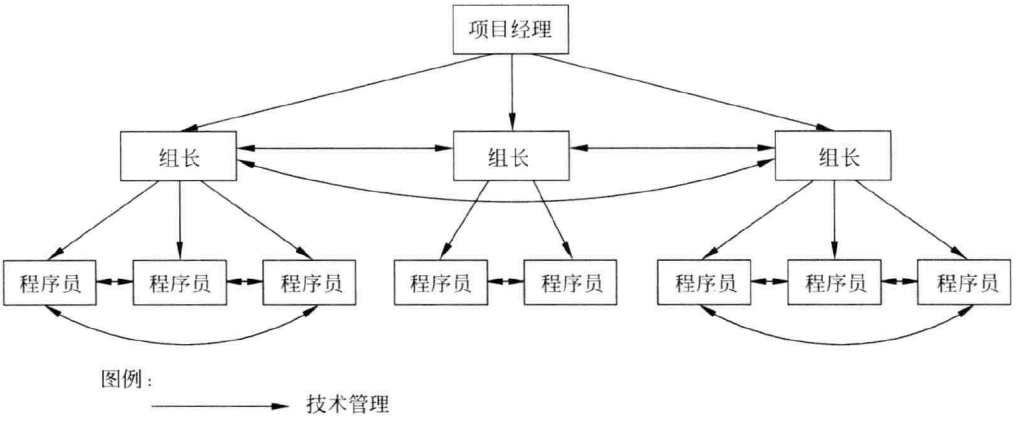
3. **现代程序员组**
   - 主程序员由两个人担任：技术负责人；行政负责人。分工明确。明确划分技术负责人和行政负责人权限
   - 现代程序员组结构



   - 软件规模较大时，程序员组分成若干个小组



   - 将民主制程序员组与主程序员组的优点结合进来，形成包含分散决策组织形式



第五节 质量保证

1、软件质量的定义

- 与软件产品满足规定的和隐含的需求能力有关的特征或特性全体

2、软件质量的三个要点

- 软件需求是度量软件质量的基础
- 按规范化标准定义开发准则，不遵守软件质量不能保证
- 不能忽略隐含需求

3、软件质量的影响因素

4、软件质量保证的措施

- 基于非执行的测试：复审或评审
- 基于执行的测试：软件测试
- 程序正确性证明

5、技术审查的必要性

- 保证编码前各阶段文档质量，及早纠正大部分缺陷
- 包括走查和审查

6、走查

- 是开发者的一次友好的会议，需要仔细规划，有明确的目的、日程、持续时间和参与人员，许多小组以星期为单位走查
- 会后将问题分发给相应人员进行解决

7、审查

- 最系统化严密的评审技术
- 审查范围比走查广泛、步骤较多
- 基本步骤
  - 综述
  - 准备
  - 审查
  - 返工
  - 跟踪
- 审查组成员：
  - 组长(同时是技术负责人)；
  - 负责开发工作的项目组代表(当前阶段和下一阶段)
  - SQA小组代表

8、程序正确性证明

- 用数学方法验证程序与说明一致。对评价小程序适用

第六节 配置管理

1、配置管理的定义

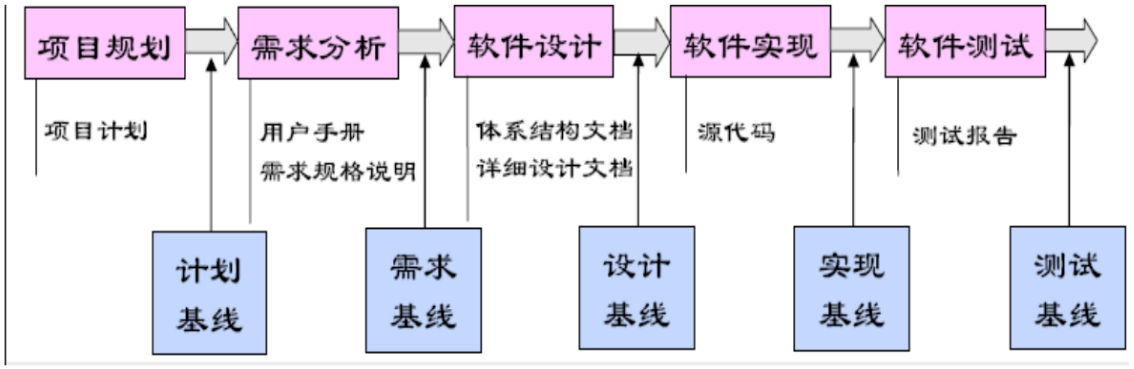
- 软件配置管理是软件的整个生命周期内管理变化的一组活动
- 软件配置管理不同于软件维护
- 软件配置的目标是使变化更正确且更容易被适应，在必须变化时减少所需花费的工作量

2、软件配置项

- 计算机程序：(源程序及目标程序)
- 文档：(包括技术文档和用户文档)
- 数据：(程序内包含的或在程序外的)

3、基线

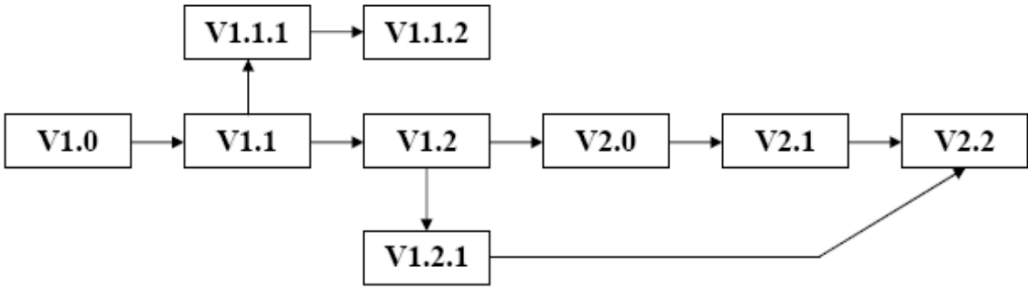
- IEEE定义：已经通过正式复审的规格说明或中间产品，可作为进一步开发基础，并且只有通过正式的变化控制才能改变它
- 简而言之，基线就是通过了正式复审的软件配置项
- 基线标志着软件开发过程的各个阶段的里程碑



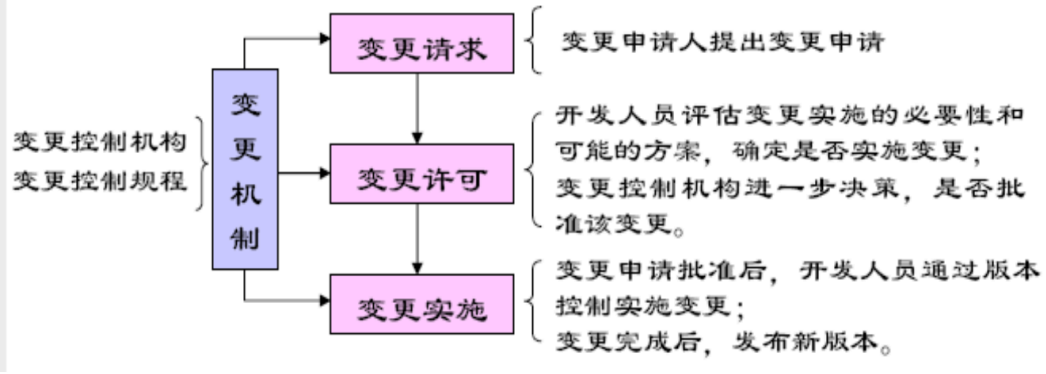
4、软件配置管理过程

- 主要有5个任务：配置标识、版本管理、变更控制、配置审计和配置报告

1. **配置标识**
   - 标识两类对象：基本对象和复合对象
   - 基本对象：软件工程师分析、设计、编码和测试时建立“文本单元”。
     - 如：需求规格说明一节，源程序清单、一组测试用例
   - 复合对象：是基本对象或其它复合对象的集合
   - 对象标识：(名字、描述、资源表、“实现”)
2. **版本控制**
   - 版本控制是对配置对象不同版本标识和跟踪过程。保证软件技术的一致性



3. **变化控制**
   - 变化控制是建立一套组织结构和控制规程，有意识地控制软件的变更过程变化控制的过程



4. **配置审计**
   - 确保所有文档内容变动不超出当初确定软件要求范围
5. **状态报告**
   - 对开发过程做系统记录，反映开发活动历史情况
   - 主要回答发生了什么事？谁做的这件事？这件事是什么时候发生的？它将影响哪些其他事物？