目录

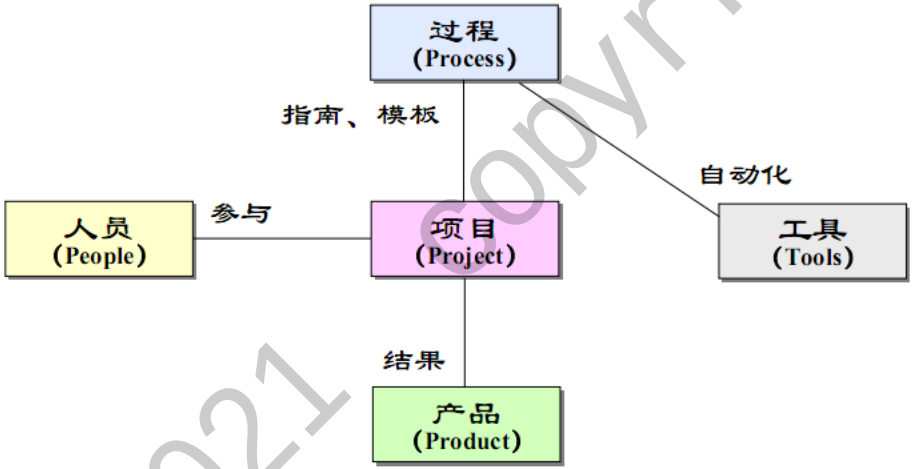
软件项目的特点和软件管理的职能

- 软件项目管理是为了使软件项目能够按照预定的成本、进度、质量顺利完成，而对成本、人员、进度、质量、风险等进行分析和管理的活动。

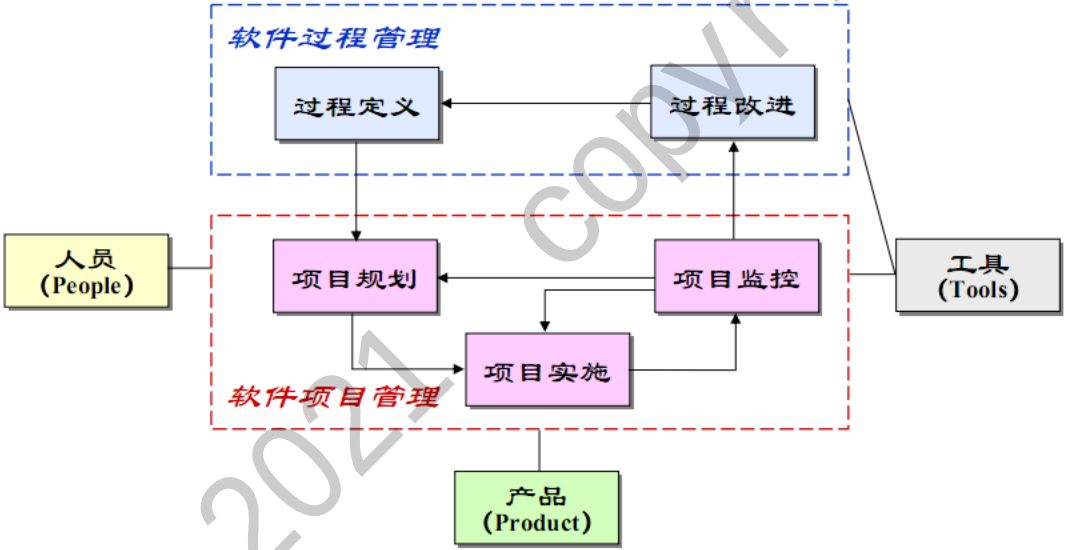
- 软件项目的特征
  - 软件产品的不可见性
  - 项目的高度不确定性
  - 软件过程的多变化性
  - 软件人员的高流动性

- 降低复杂性和控制变化是软件项目管理的关键问题。

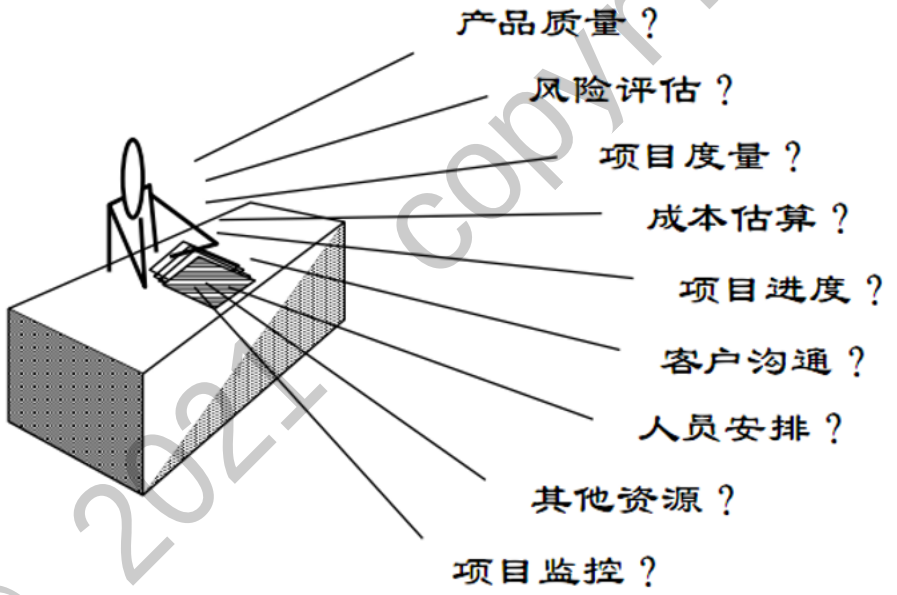
软件项目管理的4P



项目管理与过程管理的关系



软件项目管理的关注点



软件项目管理活动

- 项目启动阶段
  - 确定项目范围、组建项目团队、建立项目环境
- 项目规划阶段
  - 确定项目活动、预算项目成本、制定进度计划
- 项目实施阶段
  - 监控项目执行、管理项目风险、控制项目变更
- 项目收尾阶段
  - 客户验收项目、总结项目经验

造成软件项目失误的原因

- 造成软件项目失误的原因主要涉及到软件项目研制中的以下软件管理的许多侧面：

  计划制定、
  进度估计、
  资源使用、
  人员配备、组织机构和
  管理方法(质量,配置)等

13.1 估算软件规模

代码行技术

– 代码行技术是一种简单而直观的软件规模估算方法，它从过去开发类似产品的经验和历史数据出发，估算出所开发软件的代码行数。

- 优点
  – 简单方便，在历史数据可靠的情况下可以很快估算出比较准确的代码行数。
- 缺点
  – 需要依赖比较详细的功能分解，难以在开发初期进行估算
  – 估算结果与所用开发语言紧密相关，无法适用于非过程语言

估算软件规模 – 功能点技术

- 功能点技术
  - 功能点技术是依据软件信息域的基本特征和对软件复杂性的估计，估算出软件规模。
  - 这种方法适合于在软件开发初期进行估算，并以功能点为单位度量软件规模。

- 软件信息域的 5 个基本特征
  - 外部输入：用户进行添加或修改数据的屏幕或表格
  - 外部输出：软件为用户产生的输出屏幕或报表
  - 外部查询：软件以联机输出方式产生的独立查询
  - 内部逻辑文件：软件修改或保存的逻辑记录集合
  - 外部接口：与其他系统进行信息交换或共享的文件

估算软件规模－ 功能点技术

- 功能点计算

信息域特征	估算值		加权因子				单项总和
			简单	中等	复杂		
外部输入		×	3	4	6	=	
外部输出		×	4	5	7	=	
外部查询		×	3	4	6	=	
内部逻辑文件		×	7	10	15	=	
外部接口		×	5	7	10	=	
				未调整功能点总计 UFP			

- 计算公式 $FP = UFP \times [0.65 + 0.01 \times \sum F_i]$

估算软件规模 – 功能点技术

复杂度调整值F_i(F_1~F_{14})，其取值范围：0~5

0: 无影响，1: 偶然，2: 适中，3: 普通，4: 重要，5: 极重要

序号	F_i	技术因素
1	F_1	数据通信
2	F_2	分布式数据处理
3	F_3	性能标准
4	F_4	高负荷的硬件
5	F_5	高处理率
6	F_6	联机数据输入
7	F_7	终端用户效率
8	F_8	联机更新
9	F_9	复杂的计算
10	F_{10}	可重用性
11	F_{11}	安装方便
12	F_{12}	操作方便
13	F_{13}	可移植性
14	F_{14}	可维护性

13.2 工作量估算

- 根据代码行数或功能点数估算出工作量，工作量的单位通常是人月（pm）
- 工作量是软件规模（KLOC或FP）的函数。
- 有静态单变量模型和动态多变量模型

工作量估算 – COCOMO2模型

- COCOMO 模型
  - 结构性成本模型 COCOMO（COnstructive COst MOdel）是一种利用经验模型进行成本估算的方法
  - COCOMO2模型　　以基本 COCOMO 模型为基础，通过对影响工作量的若干因素进行估计，确定出调节因子，再对工作量估算公式进行修正。

COCOMO2模型计算公式

– 计算公式

$$E = aL^b F$$

$$F = \prod_{i=1}^{17} F_i$$

类型	a	b	说明
组织型	3.2	1.05	各类应用软件
半独立型	3.0	1.12	各类实用程序、编译程序等
嵌入型	2.8	1.2	实时处理程序、控制程序、操作系统等

其中，

E: 开发工作量（以人月为单位）

a: 模型系数

KLOC: 估计的源代码行数（以千行为单位）

b: 模型指数(COCOMO2 模型中为一个计算获得的值)

F_i(i=1~17)：成本因素。

COCOMO2模型模型指数b的取值

- COCOMO2使用的5个分级因素来计算Wi值， Wi值范围：0~5，共6个级别。

（1） 项目先例性。
（2） 开发灵活性。
（3） 风险排除度
（4） 项目组凝聚力。
（5） 过程成熟度。

$$b = 1.01 + 1.01 \times \sum_{i=1}^{5} W_i$$

成本因素Fi(i=1~17)的取值

工作量因素 F_i		非常低	低	正常	高	非常高	超高
产品因素	软件可靠性	0.75	0.88	1.00	1.15	1.39	
	数据库规模		0.94	1.00	1.09	1.19	
	产品复杂性	0.75	0.88	1.00	1.15	1.30	1.66
	可重用性		0.91	1.00	1.14	1.29	1.49
	要求文档量	0.89	0.95	1.00	1.06	1.13	
计算机因素	执行时间限制			1.00	1.11	1.31	1.67
	存储限制			1.00	1.06	1.21	1.57
	平台变动		0.87	1.00	1.15	1.30	
人的因素	分析员能力	1.50	1.22	1.00	0.83	0.67	
	程序员能力	1.37	1.16	1.00	0.87	0.74	
	应用领域经验	1.22	1.10	1.00	0.89	0.81	
	平台经验	1.24	1.10	1.00	0.92	0.84	
	语言和工具经验	1.25	1.12	1.00	0.88	0.81	
	人员连续性	1.24	1.10	1.00	0.92	0.84	
项目因素	软件工具的使用	1.24	1.12	1.00	0.86	0.72	
	多地点开发	1.25	1.10	1.00	0.92	0.84	0.78
	开发进度限制	1.29	1.10	1.00	1.00	1.00	

举例：软件 项目估算

- 某公司大约有3000名员工，准备开发一个简单的工资系统
  - 系统生成员工的工资单，列出工资的所有收入项和纳税扣除额，并在屏幕上显示工资单，工资单的功能复杂度是“复杂”；另外，系统产生7个报表，每个报表的复杂度是“简单”。
  - 系统要求用户从屏幕上输入员工的基本信息（包括员工编号、基本工资、所在等级、所属部门等）和每月的考勤情况，这两个屏幕输入的复杂度为“复杂”；另外，还有一个所得税信息的输入，其复杂度为“中等”。
  - 系统提供20个查询，每个查询的复杂度是“简单”。
  - 系统内部维护一个员工信息文件，该文件的复杂度是“复杂”。
  - 系统引用了3个数据表，包括员工基本信息、部门信息和所在等级，其中员工基本信息的复杂度是“中等”，其他两个的复杂度是“简单”。

举例：软件 项目估算（1）

- 计算未调整功能点

	低	平均	高	合计
外部输入	0×3	1×4	2×6	16
外部输出	7×4	0×5	2×7	42
外部查询	20×3	0×4	0×6	60
内部逻辑文件	0×7	0×10	1×15	15
外部接口文件	2×5	1×7	0×10	17
未调整的功能点数：				150

举例：软件 项目估算（2）

- 计算调整后的功能点

技术因素	影响值	技术因素	影响值
备份与恢复	3	联机更新	0
数据通信	0	操作方便	5
分布式处理	0	内部复杂处理	2
性能	3	可重用性	0
配置负载	2	易安装	5
联机数据输入	2	多个站点	0
终端用户效率	4	可维护性	1

$$FP = 150 \times [0.65 + 0.01 \times \Sigma Fi] = 138$$

举例：软件 项目估算（3）

- 部分开发语言的功能点转换

开发语言	LOC / FP	开发语言	LOC / FP
汇编语言	320	C	128
COBOL	91	C++	53
FORTRAN 77	105	VB 6	24
PASCAL	91	PB	16
Ada	71	Delphi 5	18
LISP	64	JAVA	48

- 系统规模
  - 假设用 VB 开发，则源程序行数 SLOC＝24×138＝3312

举例：软件 项目估算（4）

- 采用COCOMO2模型方法估算开发成本
  - 名义工作量 $E = 3.0 \times 3.312^{1.12} = 11.47$（人月）
  - 实际工作量 $E = 11.47 \times \prod_{i=1}^{17} F_i = 11.47$（为简化计算 F_i 均取 1）
  - 假设每个开发人员平均每月的人工成本为 6000 元，那么总人工成本 = 11.47 × 0.6 = 6.88（万元）

- 问题：如何考虑开发人员的数量？

Brooks规律：向一个已经延期的项目增加人力，只会使得它更加延期。软件项目的开发时间最多可以减少到正常开发时间的75%。

13.3 进度计划

13.3.1 估算开发时间

估算出完成给定项目所需的总工作量之后，接下来需要回答的问题就是：**用多长时间才能完成该项目的开发工作？**对于一个估计工作量为20人月的项目，可能想出下列几种进度表：

1个人用20个月完成该项目；

4个人用5个月完成该项目；

20个人用1个月完成该项目。

但是，这些进度表并不现实，实际上软件开发时间与从事开发工作的人数之间并不是简单的反比关系。

13.3.1 估算开发时间

- 各种模型估算开发时间的方程很相似，例如：
  - **Walston_Felix模型** $T=2.5E^{0.35}$
  - **原始的COCOMO模型** $T=2.5E^{0.38}$
  - **COCOMO2模型** $T=3.0E^{0.33+0.2\times(b-1.01)}$
  - **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）。

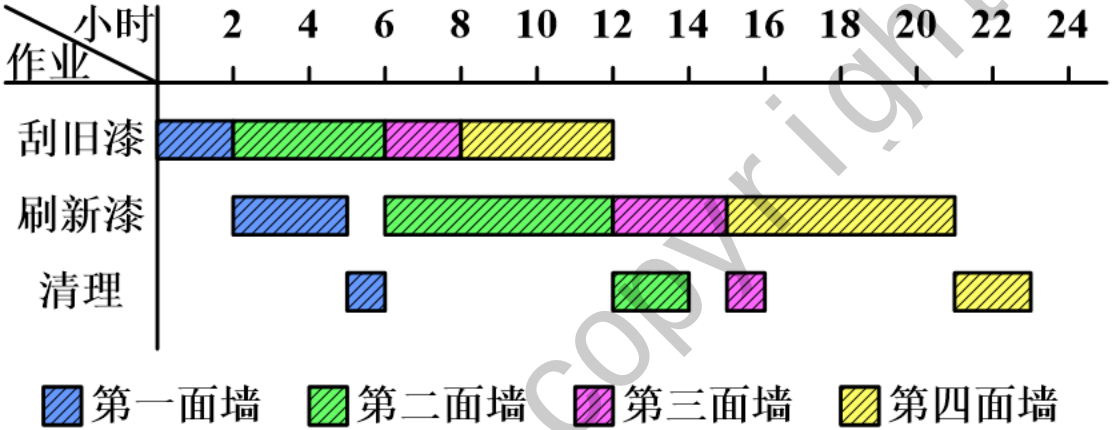
存在一个最佳的项目组规模P_{opt}，这个规模的项目组其总生产率最高。项目组的最佳规模是5.5人，即P_{opt}=5.5。

13.3.2 Gantt图

- Gantt(甘特)图是历史悠久、应用广泛的制定进度计划的工具。
- 例子：旧木板房刷漆工程(15名工人，工具各5把)

表13.5 各道工序估计需用的时间(小时)

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



Gantt图也有3个主要缺点：

(1) 不能显式地描绘各项作业彼此间的依赖关系；

(2) 进度计划的关键部分不明确，难于判定哪些部分应当是主攻和主控的对象；

(3) 计划中有潜力的部分及潜力的大小不明确，往往造成潜力的浪费。

13.4 人员组织

一、组织原则

在建立组织时应注意到以下的原则：

（1）尽早落实责任：要尽早指定专人负责软件开发，使他有权进行管理，并对任务的完成负责。

（2）减少接口：开发过程中，人员之间的联系是必不可少的，但应注意，组织的工作效率是和完成任务中存在的人际联系数目成反比的。

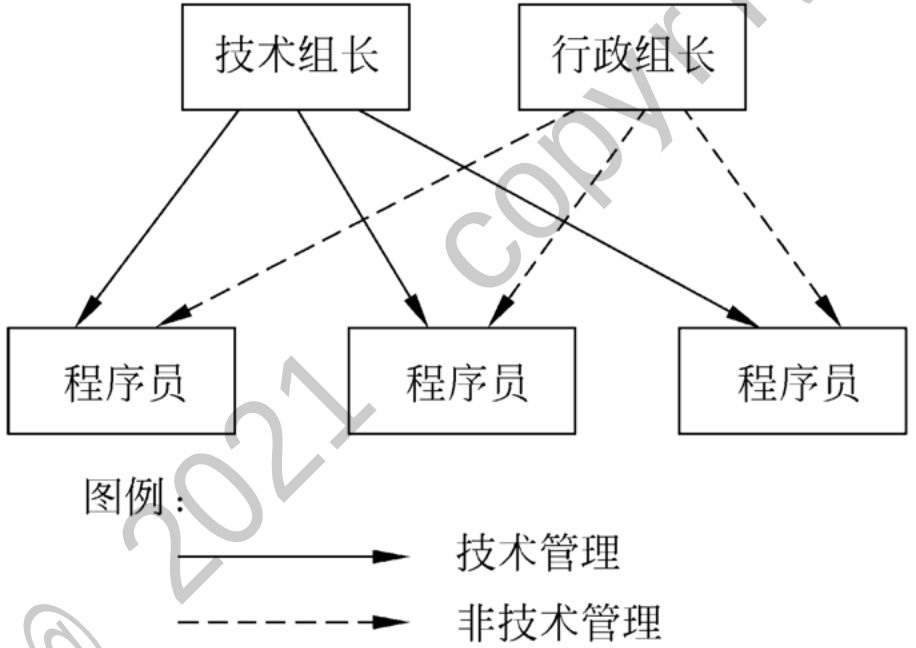
（3）责权均衡：软件经理人员所负的责任不应比委任给他的权力还大。

人员组织的形式

一个有高度凝聚力的小组由一批团结得非常紧密的人组成，他们的整体力量大于个体力量的总和。

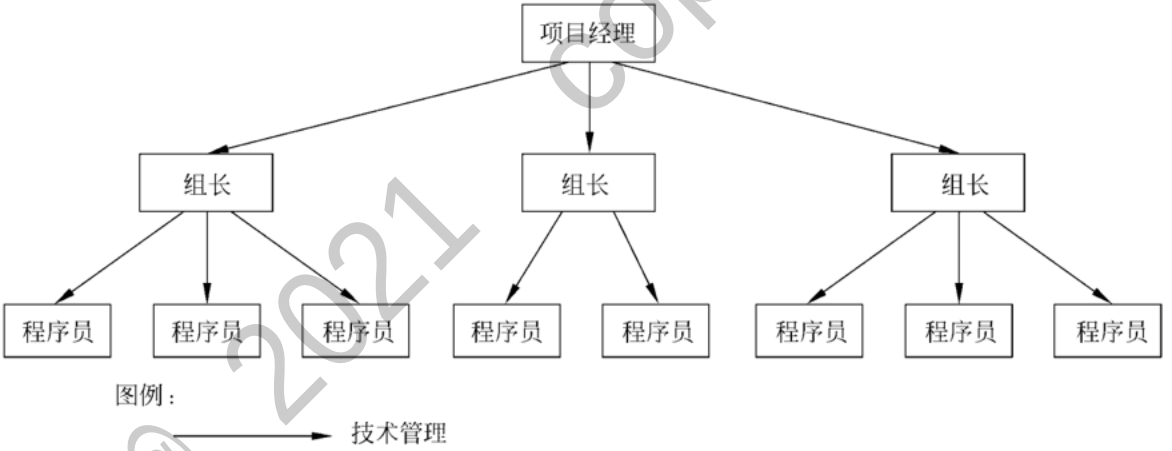
- 民主制程序员组
- 主程序员组
- 现代程序员组

现代程序员组



大型项目的技术管理组织结构

由于程序员组成员人数不宜过多，当软件项目规模较大时，应该把程序员分成若干个小组。该图描绘的是技术管理组织结构，非技术管理组织结构与此类似。



更多的项目角色分工

- 需求分析师
- 软件架构师
- 软件设计师（担任详细设计任务）
- 程序员
- 人机交互师
- 软件测试

- 项目经理
- 软件配置管理
- 质量保证人员
- 培训和支持人员
- 文档编写人员

软件项目人员配备

配备人员的原则

（1）重质量：软件项目开发是技术性很强的工作，任用少量有实践经验、有开发能力的人员去完成关键性任务，常常要比使用较多的经验不足的人员更有效。

（2）重培训：花力气培养所需的技术人员和管理人员，是有效地解决人员问题的好办法。

（3）双阶梯提升：人员的提升应分别按技术职务和管理职务进行，不能混在一起。

软件项目人员配备

对项目经理人员的要求

（1）能把用户提出的非技术性要求加以整理提炼，以技术说明书的形式转告给分析员和测试员。

（2）能说服用户放弃一些不切实际的要求，以便保证合理的要求得以满足。

（3）能够把表面上似乎无关的要求集中在一起，归结为“需要什么”、“要解决什么问题”，这是一种综合问题的能力。

（4）要懂得心理学，能说服上级领导和用户，既要让他们理解什么是不切实际的要求，又要让他们毫不勉强、乐于接受。

软件项目人员配备

评价软件人员的条件

（1）牢固掌握计算机软件的基本知识和技能。

（2）善于分析、综合问题，具有严密的逻辑思维能力。

（3）工作踏实、细致，遵循标准和规范，具有严格的科学作风。

（4）工作中表现出耐心、毅力和责任心。

（5）善于听取别人意见，善于与周围人员团结协作，建立良好的人际关系。

（6）具有良好的书面和口头表达能力。

IT公司在招聘时关注的内容

- 有没有与项目需要的技术技能/经验
- 本身的技术知识面和动手能力
- 个人学习能力和潜在的素质
- 表达能力，逻辑思维能力
- 是否是一个易于合作的人

13.5 质量保证

- 软件质量就是“软件与明确地和隐含地定义的需求相一致的程度”。

  - 功能和性能需求
  - 文档中明确描述的开发标准
  - 任何专业开发的软件产品都应该具有的隐含特征

软件质量因素与产品活动的关系

可理解性（我能理解它吗？）
可维修性（我能修复它吗？）
灵活性（我能改变它吗？）
可测试性（我能测试它吗？）

可移植性（我能在另一台机器上使用它吗？）
可再用性（我能再用它的某些部分吗？）
互运行性（我能把它和另一个系统结合吗？）

正确性（它按我的需要工作吗？）
健壮性（对意外环境它能适当地响应吗？）
效率（完成预定功能时它需要的计算机资源多吗？）
完整性（它是安全的吗？）
可用性（我能使用它吗？）
风险（能按预定计划完成它吗？）

软件质量保证措施

- （software quality assurance，SQA）

- 基于非执行的测试（也称为复审或评审）
- 基于执行的测试（即以前讲过的软件测试）
- 程序正确性证明

参加软件质量保证工作的人员

可以划分成下述两类：

- 软件工程师：
  - 技术方法和度量
  - 正式的技术复审（*包括走查（walkthrough）和审查（inspection）*）
  - 计划周密的软件测试

- SQA小组：计划，监督，记录，分析和报告。简而言之，SQA小组的作用是，通过确保软件过程的质量来保证软件产品的质量。

  确保软件过程的质量来保证软件产品的质量。

13.6 软件配置管理

软件配置管理，简称SCM（Software Configuration Management），是贯穿于整个软件工程中的保护性活动。

软件配置管理的主要目标是使软件的变更和修改可以更容易被适应，并减少当变更必须发生时所需花费的工作量。

软件工程项目中的变更和修改总是不可避免的，因此SCM活动被设计用于标记变更、控制变更、确保变更正确地实现、向其他有关的人报告变更等。

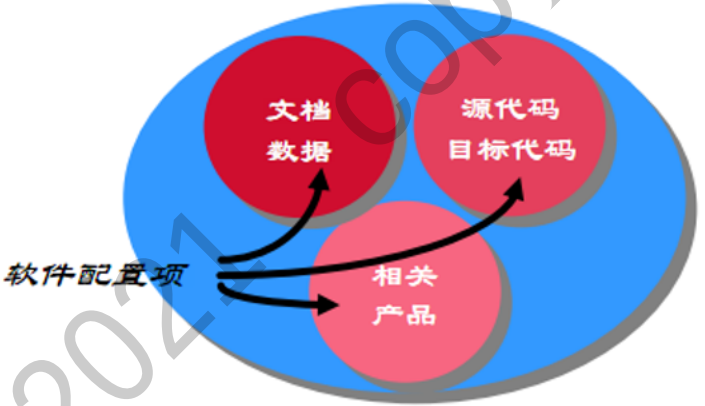
软件配置管理-管理变化

配置管理的内容

- ①标识变化；
- ②控制变化；
- ③确保适当地实现了变化；
- ④向需要知道这类信息的人报告变化。

软件配置管理-软件配置项

- 软件配置项是为了配置管理而作为单独实体处理的一个工作产品或软件。



软件配置管理和配置管理工具

软件配置项

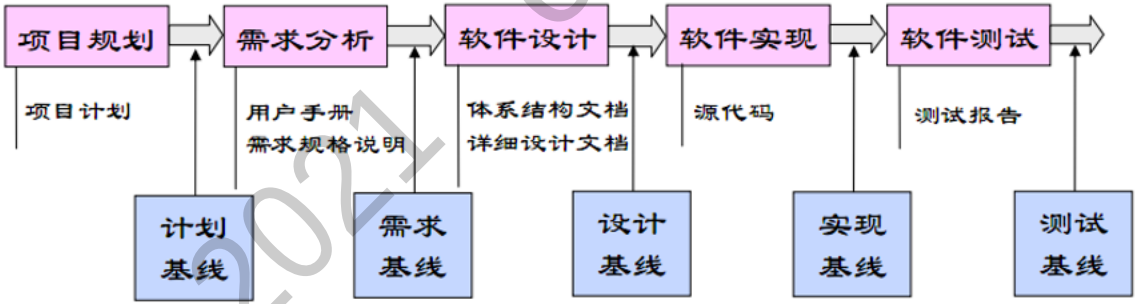
SCI是软件配置管理的对象。主要的SCI有：

- 系统规格说明书
- 软件需求规格说明书
- 用户手册初稿
- 详细设计规格说明书
- 测试计划
- 操作手册
- 软件问题报告
- 维护请求
- 软件工程标准
- 软件项目开发计划
- 可供使用的原型
- 总体设计规格说明书
- 源程序清单
- 测试报告
- 用户手册正式稿
- 可直接运行的目标码程序
- 工程变更通知
- 项目开发总结

软件配置管理和配置管理工具

- 基线是已经通过了正式复审的规格说明或中间产品，它可以作为进一步开发的基础，并且只有通过正式的变化控制过程才能改变。

- 基线标志着软件开发过程的各个里程碑。



软件配置管理和配置管理工具

- 基线标志软件生存期中各个开发阶段末尾的特定点，又称里程碑。
- 基线的作用是使各阶段工作的划分更加明确化，使本来连续的工作在这些点上断开，以便于检验和肯定阶段成果，例如明确规定不允许跨越里程碑修改另一阶段的文档。

软件配置管理工具

- *Rational ClearCase*
  - 版本控制、工作空间管理
  - 支持并行开发
  - 统一变更管理
  - 与 Microsoft 和 IBM 的开发工具相集成
- *Microsoft SourceSafe*
  - 版本控制
  - 与 Microsoft 的开发工具相集成
- *CVS*
  - 并发版本控制
  - 开放源码的软件开发

13.7 能力成熟度模型 CMM(Capability Maturity Model)

- 美国卡内基梅隆大学软件工程研究所在美国国防部资助下于20世纪80年代末建立的能力成熟度模型。
- 用于评价软件机构的软件过程能力成熟度的模型。
- 最初，建立此模型的目的主要是，为大型软件项目的招投标活动提供一种全面而客观的评审依据，发展到后来，此模型又同时被应用于许多软件机构内部的过程改进活动中。

CMMI的5个级别

最适化阶段（程度５）	最好的实践方式的确立和继续的过程改善 ·缺陷预防 ·技术变更管理 ·过程变更管理
管理阶段（程度４）	过程、成果物测定值被收集、被分析、定量的过程控制 ·定量的过程管理 ·软件质量管理
定义阶段（程度３）	组织过程的标准被确立，应用于管理和工程活动中 ·组织过程焦点 ·组织过程定义 ·培训计划 ·综合的软件管理 ·软件产品工程 ·group间的协调 ·同级评审（内部Review）
反复可能阶段（程度２）	可确立基本的项目管理、成本、进度获得了管理 ·软件配置管理 ·软件质量保证 ·软件项目的追踪和监视 ·软件子合同管理 ·软件项目计划 ·软件需求管理
初级阶段（程度１）	项目的成功依从于个人的努力（混沌）

软件管理的内容

- 制定计划
- 控制进度
- 保证质量
- 管理变化

END.