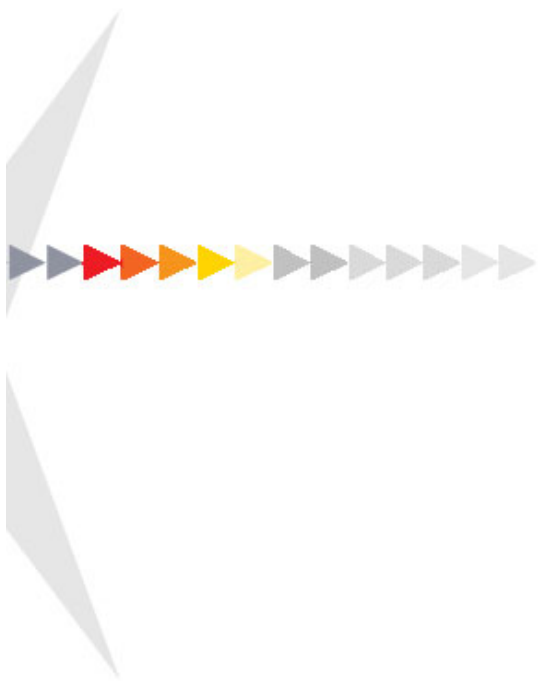


软件工程

大连理工大学软件学院



第12章 软件项目级管理

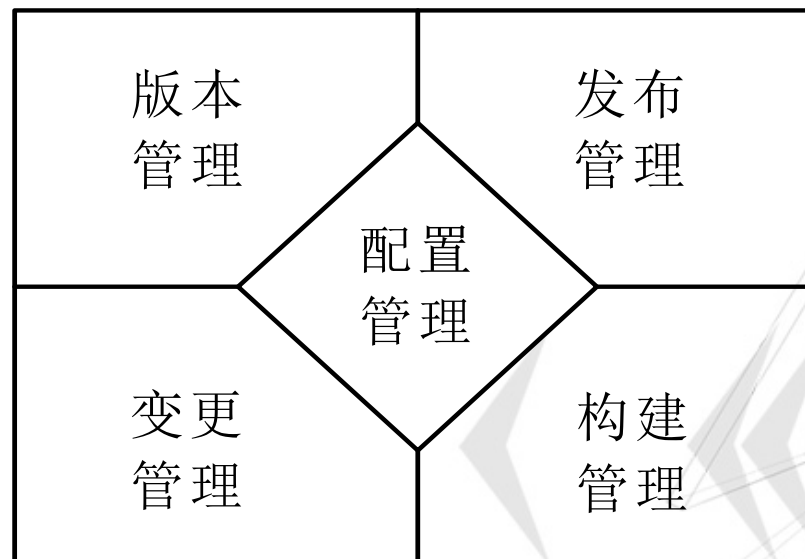


- 项目开发，除了技术，软件开发的管理技术和方法也很重要。软件的管理可以分为两个层面：首先是项目级管理，然后是组织级管理。
- 本章主要是阐述项目级管理的方面和相关技术方法，主要包括软件配置管理、软件项目管理、软件质量保证、风险管理、人员沟通管理等方面，并在每个方面中详细描述了相关的内容、技术和方法。
- 项目管理中的人员组织和管理。

软件配置管理



- 软件配置管理（**Software Configuration Management, SCM**）是一种标识、组织和控制修改的技术，贯穿于整个软件生命周期。

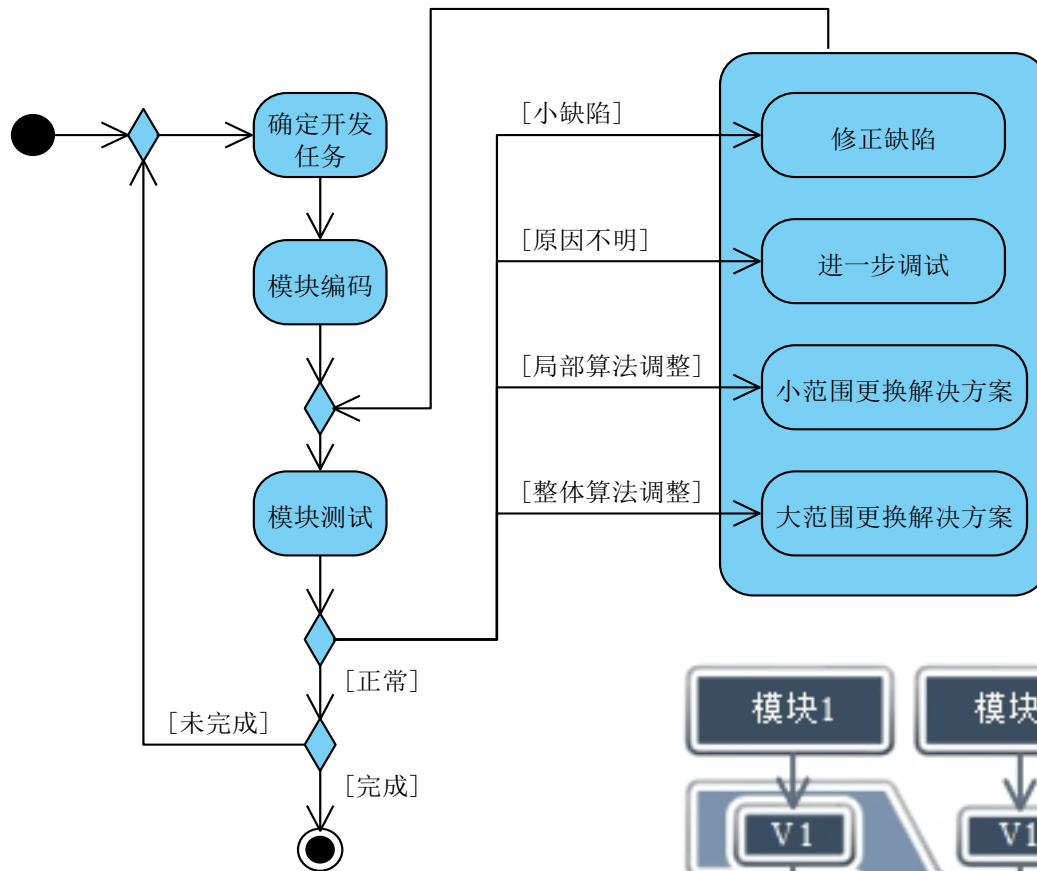


版本管理

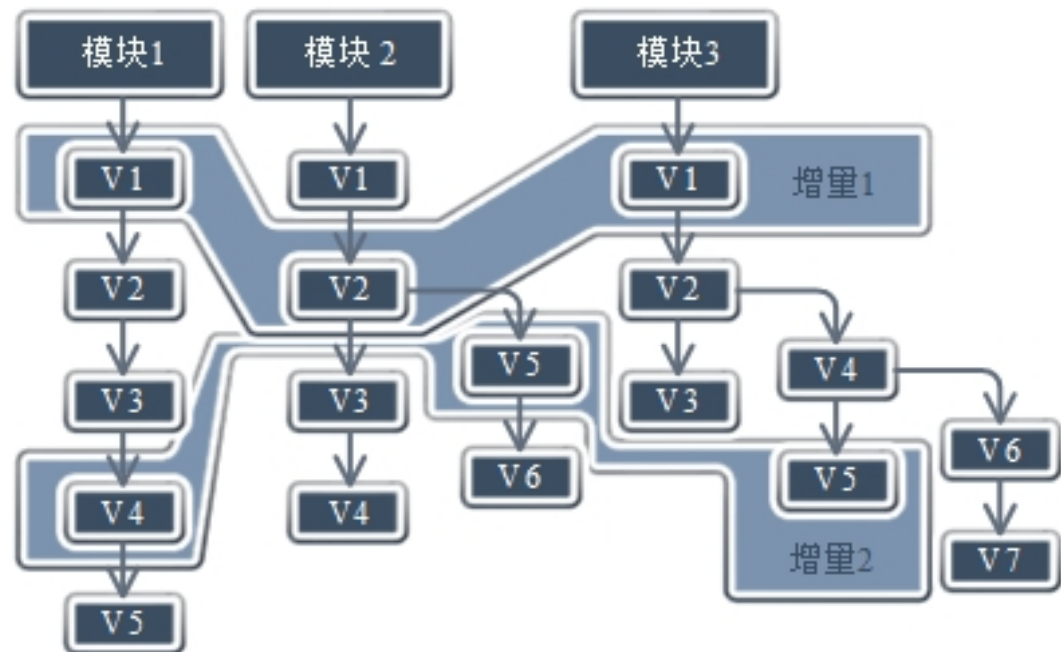


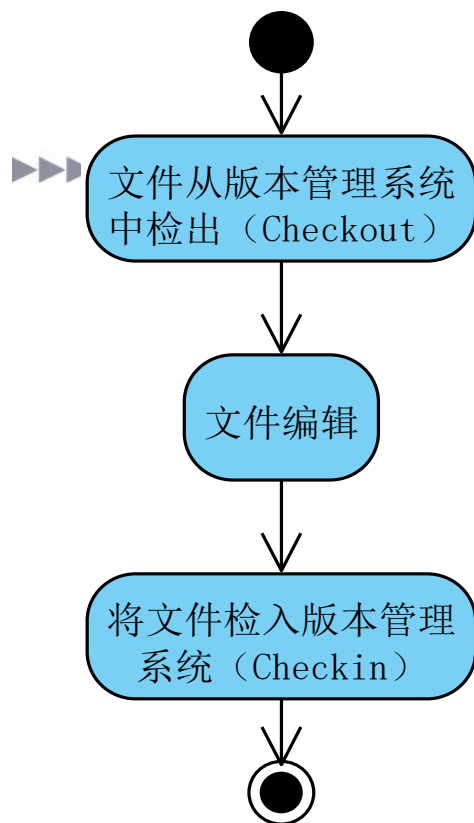
- 版本管理主要包括两个方面的工作：
 - 一方面要规范化不同开发人员之间的合作方式，必须能够保证一个人的工作不会被其它人意外的覆盖；
 - 另一方面是要确保每个人工作的对象是当前需要的版本而且能够为后续开发提供基础。
- 累进式的开发过程可能随时有一些新想法和实现，但随后又被抛弃掉，这就需要有方便回到先前工作状态的机制。

累进式的开发过程



版本树





- 版本管理系统的核心工作是对项目软件或者项目文档的管理，把存储所有项目内容的数据库称为版本仓库（**repository**），版本仓库可以理解为一个存储着所有开发历史的数据库，与通常意义的数据库不同，它一般是在现有文件系统上的高效实现。
- 所有纳入版本仓库进行管理的各种软件资产统称为软件配置项，包括各种文档、数据以及代码等。
- 配置项在其生命周期的特定时间点上通过正式评审而进入正式受控的一种状态，称之为基线（**baseline**）。
- 作为配置管理的基础，基线可理解为软件配置项的一个稳定版本，基线为后续开发活动提供了信息的稳定性和一致性。



Codeline (A)



Codeline (B)



Codeline (C)



Libraries and external components



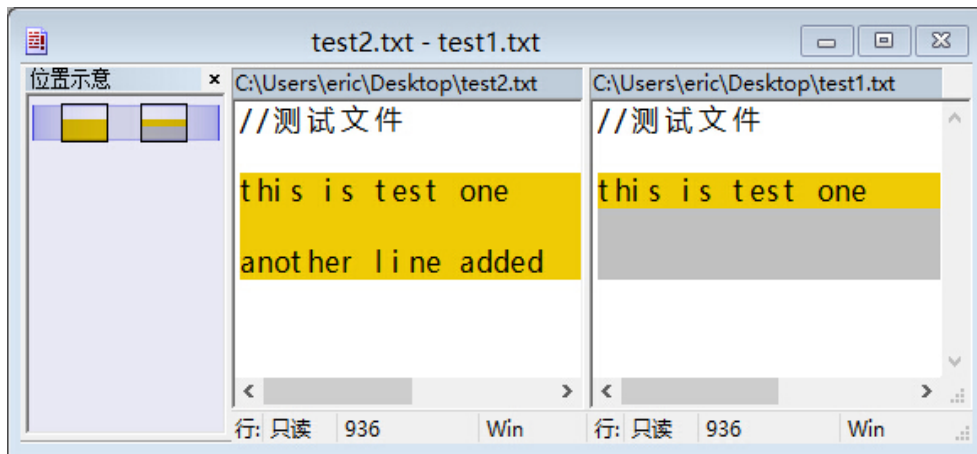
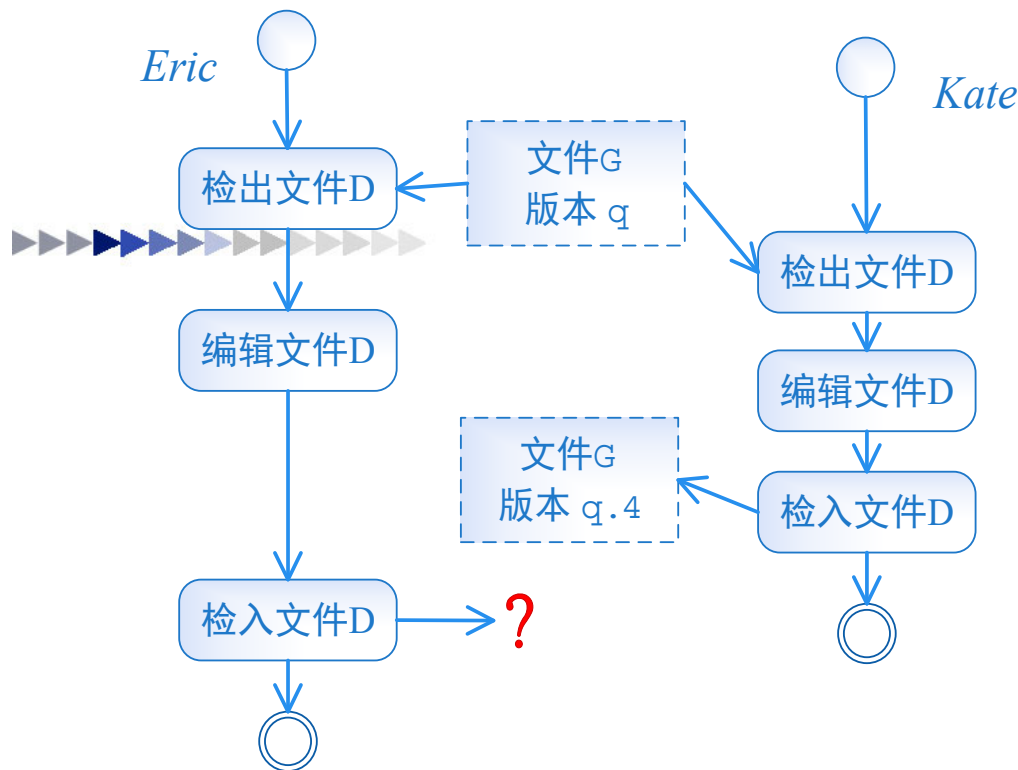
Baseline - V1



Baseline - V2



Mainline



- 版本系统对于冲突问题的解决通常存在两种方法。
- 第一种方法被称为是悲观的方法，原理是第一个检出文件的人将会拥有对该文件的排它锁。好处在于能够保证一个文件只由一个人同时进行编辑并且不会导致任何的冲突发生。
- 第二种是乐观的方法，开发人员可同时对文件进行编辑，但涉及如何合并修改和冲突的解决。
- 版本系统与其它工具联合使用会发挥出更大的价值。

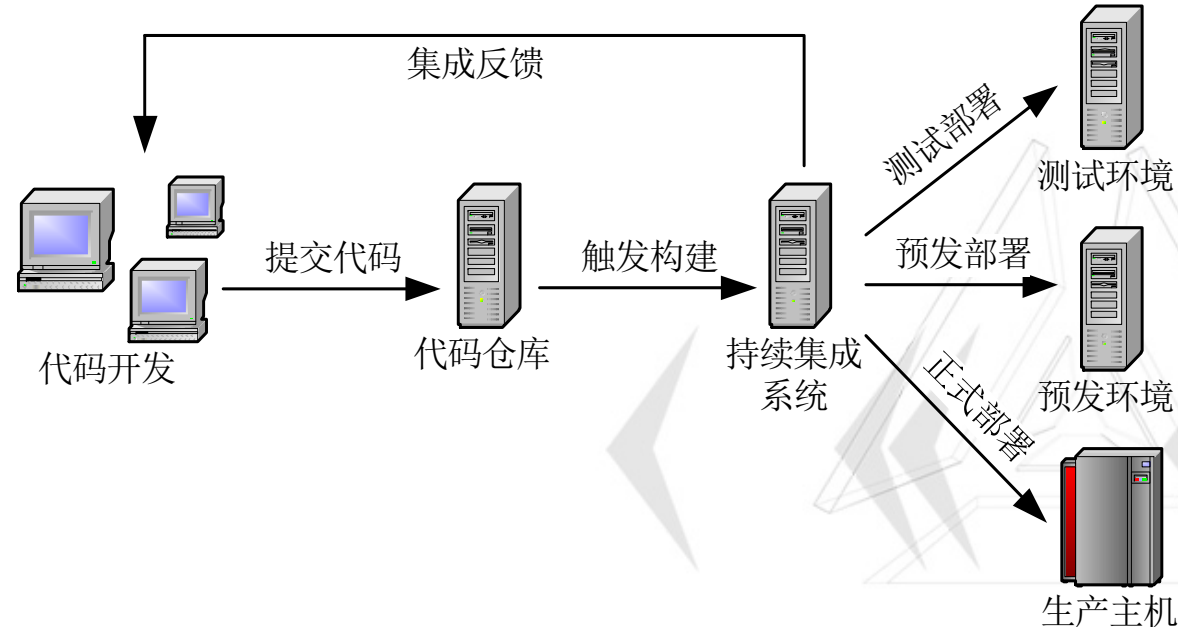
构建管理



- 构建（**Build**）管理系统的主要任务是描述最终软件产品的结构和生成过程。
 - 对于小型的开发项目，构建系统的作用是调用编译器，然后根据编程语言的不同，使用不同的链接器生成可执行文件并执行。
 - 对于大型的项目，在构建的构成中还要确保只有那些被修改的部分，才有必要进行重新编译和链接。除了编译这项主要的工作外，围绕文件的处理还有其他重要的任务，如目标软件在生成后应指定在系统中安装的具体位置等

构建管理

- 借助命令行等工具进行手工执行，如Shell或Dos等
- 开发较为大型系统时，需要引入专门的Build管理工具，如C或C++常用的Make工具和Java中的Ant工具。
- 更大型的项目可引入持续集成来进行更为复杂的构建管理



构建管理



- 在构建管理的应用领域，**Ant**和**Make**工具的主要作用：
 - 提供边界条件的管理，如系统配置以及其它相关变量；
 - 命令链的执行管理，其描述了从某些对象出发构建新对象的过程及其结果位置等。
- 举例：**Ant**的一个示例。
- 整个命令链构成了一种树状的层次结构

发布管理



- 发布管理（**Release**）的主要作用是协调在合适的时间对合适的用户交付合适产品的保证。
- 软件资源、软件开发过程以及开发人员的分散化，导致软件发布管理的复杂化。
- 软件开发不是一蹴而就的过程。
- 发布管理是对项目管理的一个有效补充。

变更管理



- 软件生存周期内全部的软件配置是软件产品的真正代表，必须使其保持精确。
- 软件过程中某一阶段的变更，均要引起软件配置的变更，这种变更必须严格加以控制和管理，保持修改信息，并把精确、清晰的信息传递到软件过程的下一步骤。
- 软件变更管理包括建立控制点和建立报告与审查制度。
- 变更管理还包括对用户的确认以及使其随时掌握变更的进度以及细节，如责任人等内容。

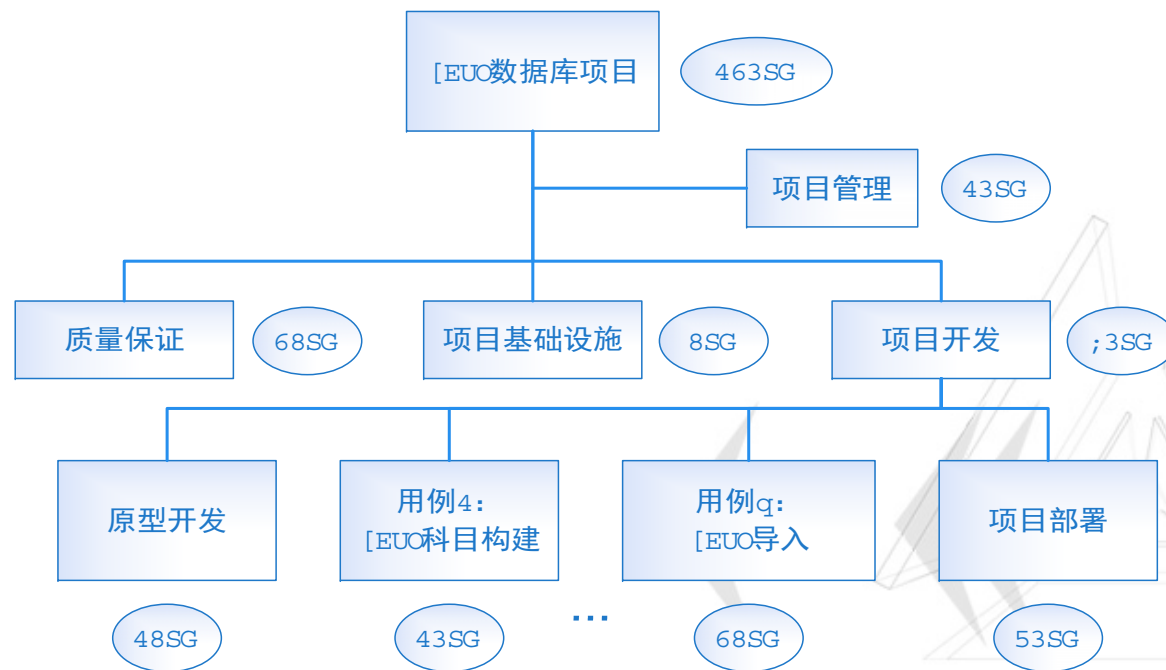
项目计划



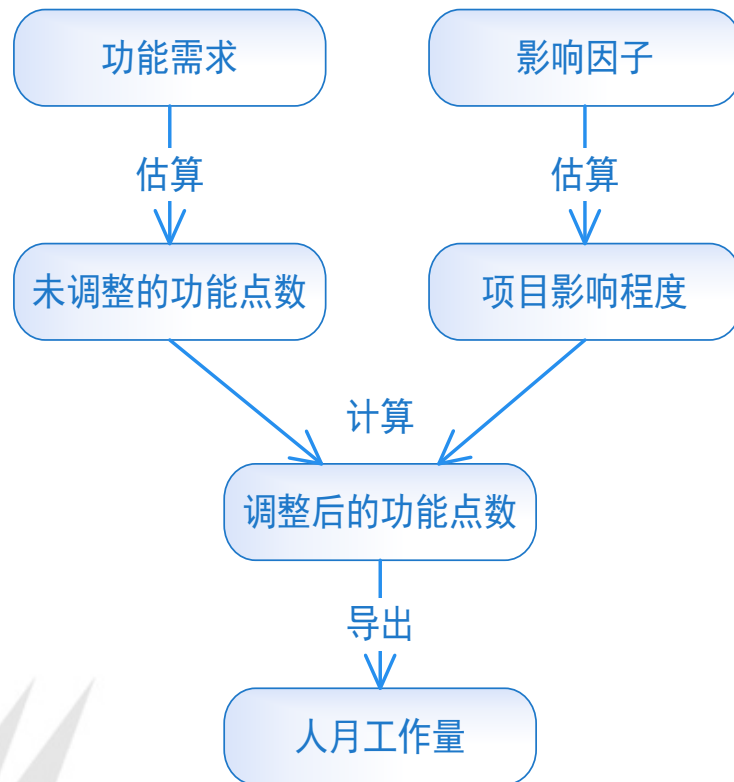
- 除了开发的方法、技术以及开发环境，**项目管理是项目成功与否的又一重要因素。**
- 与技术实现的关注点不同，项目管理主要关注**组织和管理**层面的内容。
- 项目计划实际上是一个对**项目规模、工作量、成本、进度等方面的估算，同时将人员、时间、计算机资源等各类资源统筹安排**，对项目的成功起到非常重要的作用。
 - 准确且系统地评估出项目的实际成本几乎是不可能的，能够理解项目成本估算的方法并建立起估算过程的概念，并使得估算趋于准确。
 - 项目经理在项目中处于一个核心的领导地位，他负责项目在整体上的计划、进度控制等工作，

• 工作分解WBS

- 在项目计划的开始，第一步是要决定哪些任务需要完成。这个工作可以
▶▶▶▶▶通过一种叫做工作分解结构（**Work Breakdown Structure, WBS**）的机制进行展开，其中将任务按照层次的结构由上到下逐步进行分解，
- 每项工作任务同时也给出了对应的工作量，使用单位“人天（PD）”表示。
- 对每项工作包应存在两个评估值——期望的工作量和为潜在问题预留的缓冲量。



软件规模估算



- 项目成本的估算对于项目的成功起到非常重要的作用。
- 软件规模的估算可以基于分解的方法，项目可以按照**WBS**的方式分解为子项目。
- 评估方式采用“类比”的方法，即参照以往相似项目的实际工作量导出一个估算的结果。
- 一种系统化的评估方法是功能点分析，是一项可以进行认证的评估技术。

- 功能点分析需要准确的理解当前所有的用户功能，并尽可能的将每个功能归到以下的5种任务类型之中：

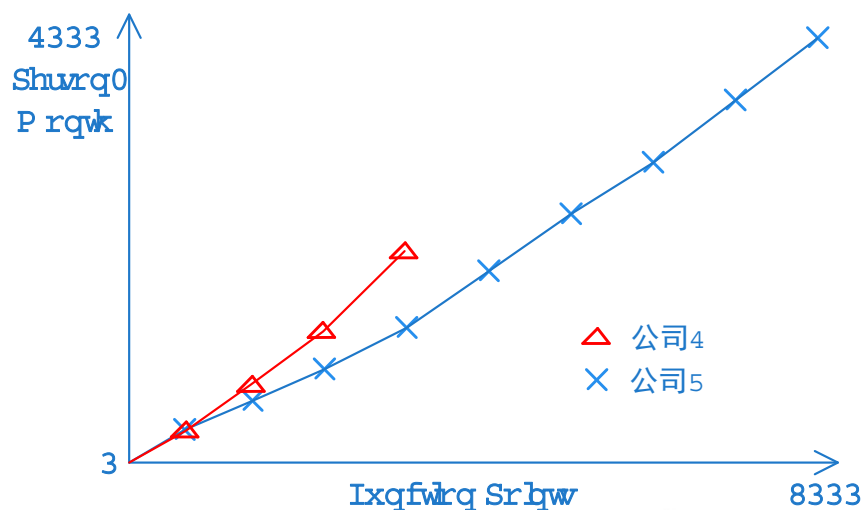
- 内部逻辑文件（**ILF**，内部数据）：在待开发系统内部处理的数据，如开发的类本身。
- 外部结构文件（**EIF**，引用数据）：从开发系统的外部引入并进行处理的数据。
- 外部输入（**EI**，输入）：从开发系统外部的输入，并由此对数据展开处理，如数据以某种格式约定（输入掩码）从系统外部的输入。
- 外部输出（**EO**，输出）：在待开发系统中实现业务计算结果外部的输出，比如数据以某种形式的输出格式（输出掩码）或对其它系统的错误消息输出。
- 外部查询（**EQ**，查询）：从外部系统发出的对数据信息的查询，对数据的查询格式、报告以及分析，不包括其它需要的附加计算。

- 然后对功能的复杂度进行考虑，简单的可以分为三个级别：容易、中等和复杂，分别赋予不同的权值。

未调整功能点数 (UFP)		113	备注
影响因子	与其它系统的交互 (0-5)	0	无交互
	分布式数据的处理 (0-5)	3	客户/服务器
	事务的高处理率 (0-5)	1	较低的连续处理要求
	处理逻辑		
	计算复杂性 (0-10)	2	简单计算
	控制复杂性 (0-5)	2	中等要求
	出错处理 (0-10)	3	数据一致性检查
	业务逻辑 (0-5)	2	标准要求
	可重用性 (0-5)	1	较低要求
	可移植性 (0-5)	1	较低要求
	可维护性 (0-5)	5	较高要求
综合影响合计 (DI)		20	
影响因子 (TCF) = $DI/100 + 0.7$		0.9	
调整后功能点数 (FP) = 未调整功能点数 (UFP) * TCF		101.7	

开发成本估算

- 将带有权重的功能点评估分值转换为对工作量的评估，一种方法是通过经验曲线进行对应。



- 另外一种方法是模型CoCoMo（Constructive Cost Model）。

子模型	作用
Application Composition model	针对使用集成CASE工具进行快速应用开发的项目工作量评估及计划调度。
▶▶▶ Early Design and Post Architecture models	针对基础软件设施、重要应用以及嵌入软件项目开发的工作量评估及计划调度。

- **CoCoMo**模型是模型的集合，根据项目的级别或种类选取其中不同的模型使用，这里主要针对项目开始的早期设计模型进行介绍。
- **CoCoMo**模型最大的用处在于提供工作量估算的公式（**PM, person months**），并以此作为项目计划调度和优化的基础，公式的基本形式为：

$$PM = A * Size^E * \prod_{i=1}^n EM_i$$

- **Size**是指不含注释的程序长度，又称为**KDSI (Kilo-Thousands of Delivered Source Instructions)**，因此其基本的单位是千代码行。
- 公式中表示每个影响因子的变量 **EM_i** 。

线性影响因子

影响因子	特低	很低	低	正常	高	很高	特高
可靠性及复杂性	0.73	0.81	0.98	1	1.30	1.74	2.38
可重用性			0.95	1	1.07	1.15	1.24
目标平台特殊性			0.87	1	1.29	1.81	2.61
人员能力	2.12	1.62	1.26	1	0.83	0.63	0.50
人员经验	1.59	1.33	1.12	1	0.87	0.71	0.62
开发环境	1.43	1.30	1.10	1	0.87	0.73	0.62
开发周期		1.43	1.14	1	1	1	

指数参数E的取值

影响因子	很低	低	正常	高	很高	特高
有过类似的开发先例	6.20	4.96	3.72	2.48	1.24	0
开发的灵活程度	5.07	4.05	3.04	2.03	1.01	0
架构/风险方案	7.07	5.65	4.24	2.83	1.41	0
团队凝聚力	5.48	4.38	3.29	2.19	1.10	0
软件开发过程成熟度	7.80	7.80	4.68	3.12	1.56	0

$$E = B + 0.01 \times \sum_{j=1}^5 SF_j$$

正则化因子 **B=0.91**



- 总体上：
 - CoCoMo模型综合了对项目有较大影响的一系列边界条件
 - 模型还提供了根据实际情况对影响因子进行调整的机会。
 - CoCoMo模型实际的表现与评估者本身的经验也有着直接的关系。

简化的分析方法

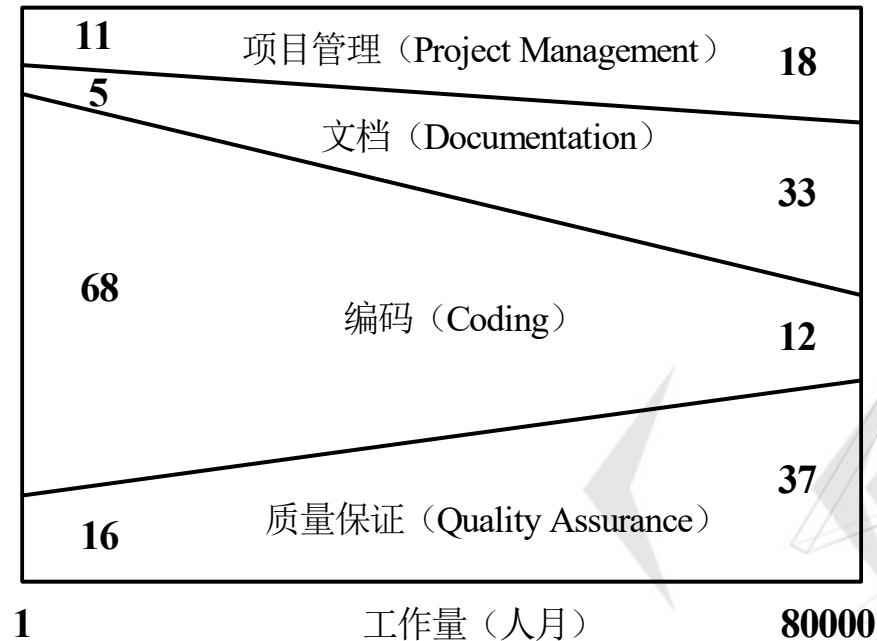
- 功能点分析和CoCoMo方法进行成本估算的关键是评估者的经验。
- 由开发人员主导进行评估，而不仅是参与评估过程。可以在开发团队内部和客户一起进行成本的评估。
- 评估的开发者应该做过新项目的需求分析工作，并且有过完整的与新项目相似类型和条件约束的项目开发经验。
- 新项目在初始的需求分析中被分解为若干任务，它们可以使用用例来进行表示。然后每个任务根据表11.6给定的四类影响因子A、B、C和D分别在规定的取值范围内进行估算。这里假定各因素间的关系是独立的，这有助于理解通过公式 $A*B*C*D$ 计算总体工作量的合理性。
- 评估过程中，
 - 首先针对那些相对中等规模任务的B值，至少要有3位评估人员参与
 - 所有评估人员能够理解这些任务并能够就它们的B值等于4（中间值）的情况达成共识。
 - 然后以此为基础，每个评估人员独立对所有余下的任务进一步评估，并在最后召集一次评估会议，就大家所有的评估值进行讨论，发现的问题和风险应进行及时记录。

- (1)对每个新的任务new_i计算其 $A*B*C*D$ 的乘积;
- (2)将各个任务的new_i值相加, 计算新项目的工作量分数:
 $AP_new=new_1+...+new_n$;
- (3)对每个已完成任务old_i计算其 $A*B*C*D$ 的乘积;
- (4)将各个任务的old_i值相加, 计算已完成项目的工作量分数:
 $AP_old=old_1+...+old_m$;
- (5)假定Au_old为以“人天”为单位的已完成项目的实际工作量, 计算工作量比例值: $Value_AP=Au_old/AP_old$;
- (6)使用比例值Value_AP计算新项目的成本估算值为 $AP_new*Value_AP$, 其中每个单一任务的工作量: $new_i*Value_AP$ 。

因素		描述
A 通过重用进行构建的可能性	1	没有通过重用进行构建的可能性, 所有内容都需要重新开发。
	1.3	具有一定的通过重用构建的可能性, 但需要识别出重用的内容和位置并进行适应性的调整。
	1.6	具有较高的通过重用构建的可能性, 需要较少的调整, 已有的文档对可重用的部分具有详细的说明和扩展方法。
	1.9	大部分可通过重用进行构建, 设计上也符合要求不需太多的调整。
	3	全部可基于现有的成果进行构建, 开发的主要工作是测试。
B 任务规模(没有注释的代码行数)	1-8	对任务的规模进行估计, 这里要求任何任务的规模不能超过最小任务规模的 8 倍, 每个任务的给定值是相对值。如果完全可通过重用构建, 则 B=0。
C 接口数目	1	不依赖或较少依赖其它任务。
	2	中等依赖其它任务。
	3	重度依赖其它子任务。
D 在项目中的关键程度	1	低度, 当有问题发生不会导致整个项目的失败。
	1.5	中度, 当有问题发生功能性和客户满意度将会受到影响, 但系统仍可用。
	2	高度, 当有问题发生将导致整个项目的失败。

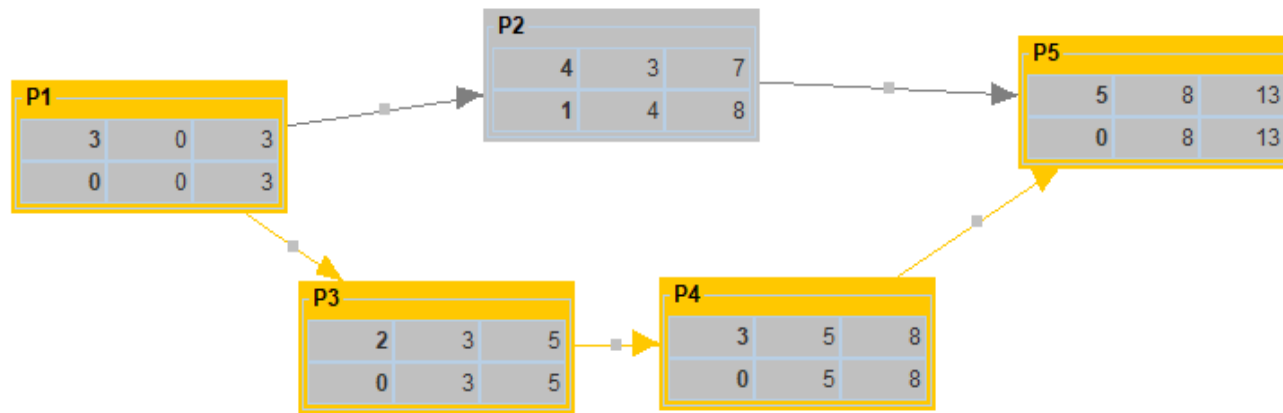
管理方面的成本

- 项目管理或者质量保证等单独活动的成本，则需要在总成本的基础上根据每个部分所占的比例进行估算。
 - 随着项目规模的增加，实际的开发工作量所占比例逐渐下降，
 - 其中的管理成本、文档化的成本以及质量保证的成本却在增加




任务安排与工程网络图

- 为了更好的完善计划，除了需要合理的估算每个工作包的工作量外，还需要识别出彼此之间的依赖关系，即理清工作任务的优先级和顺序性。

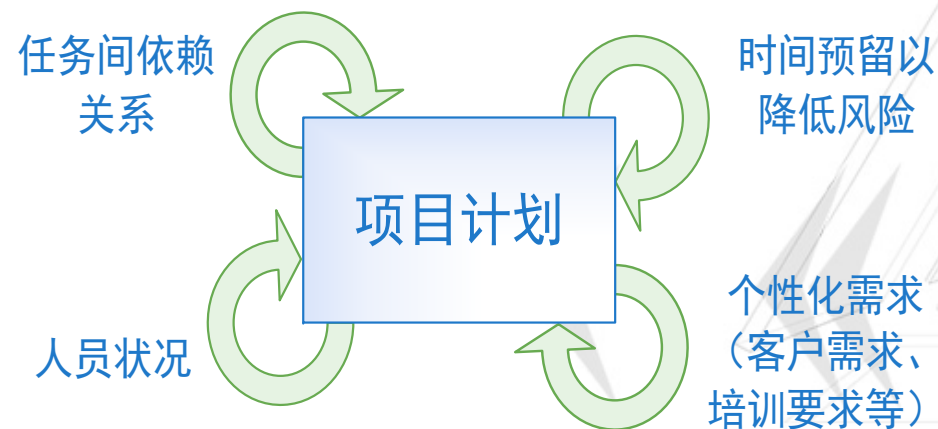


- 工作任务之间的依赖关系可以通过工程网络图进行展现，其中没有给出最小和最大工作量，而是关注每个工作包持续的时间状况。

- 
- 持续时间分析的好处是了解和掌握工作被分解到工作包后的并行情况。
 - 确定最小持续时间的路径又被称为是关键路径，每个处于关键路径上的工作包如果遇到计划外的延期则意味着整个项目的拖延。
 - 复杂项目计划中很重要的一个工作就是在关键路径上尽可能少的安排工作包，这样也能够计划在计划中为工作包产生更多的时间缓冲。
 - 每个项目计划以及所有项目员工的总体分配方案是彼此依赖的复杂系统，不仅要使得员工的工作负担尽量均衡也要使得项目尽可能快的进展下去，这对项目的计划提出了更高的要求。
















项目组织与甘特图

- 为了能够进行人员的安排，还要确定每个工作包对应的角色。
- 员工的素质和能力又会影响项目计划的调整，如进行必要的培训和指导。
- 项目计划要根据不同的条件和实际项目进行中的情况对计划进行适时的动态调整。

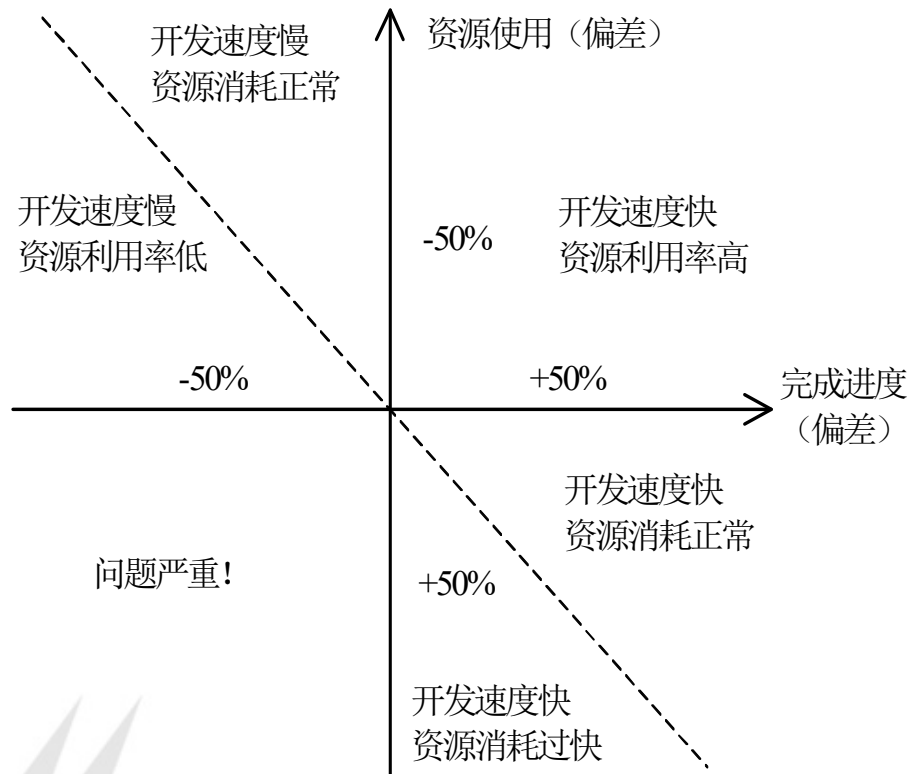




- 项目计划常使用甘特图（**Gantt chart**）图对计划进行描述，包括工作包、依赖、责任人、完成比例等。
- 实心菱形标识出里程碑的位置，表示在此位置可以对现有进度进行评审，并可根据需要对计划进行较大调整甚至终止项目。
- 内部里程碑，开发单位内部进行的进度评审；
- 外部里程碑，客户在此了解当前项目的进展并对产品进行部分的验收。

ID	任务名称	开始时间	完成	持续时间	2014年												2015年		
					03月	04月	05月	06月	07月	08月	09月	10月	11月	12月	01月	02月	03月		
1	需求确认	2014/3/17	2014/3/31	11天															
2	文献查阅、实地调研	2014/4/1	2014/7/30	87天	 														
3	数据库设计与数据整理	2014/4/15	2014/6/16	45天															
4	模块划分与概要设计	2014/6/17	2014/7/18	24天															
5	各模块详细设计与开发	2014/7/21	2014/11/10	81天	  														
6	性能评估	2014/7/21	2014/8/19	22天															
7	运行期实时风险评估	2014/8/22	2014/9/19	21天															
8	大坝三维模型	2014/9/22	2014/10/22	23天															
9	大坝时间序列模拟	2014/10/23	2014/11/10	13天	 														
10	系统集成与测试	2014/11/11	2014/12/19	29天	 														

项目计划跟踪

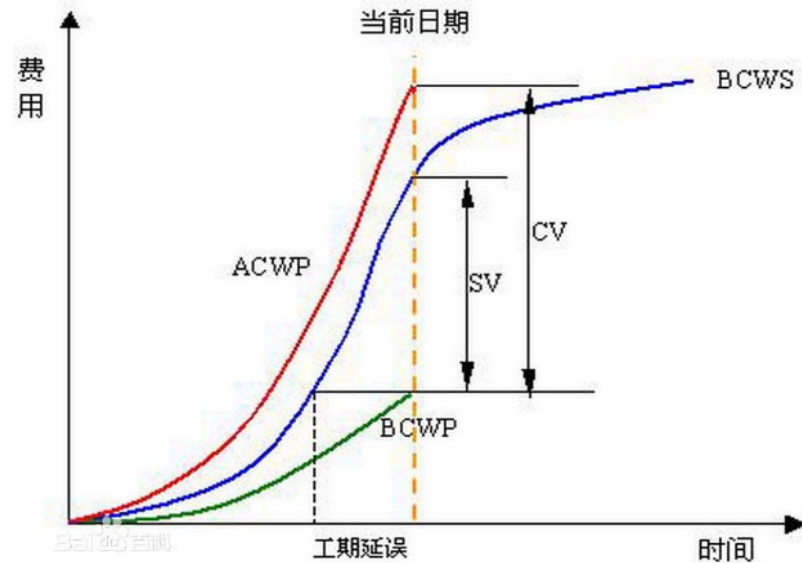
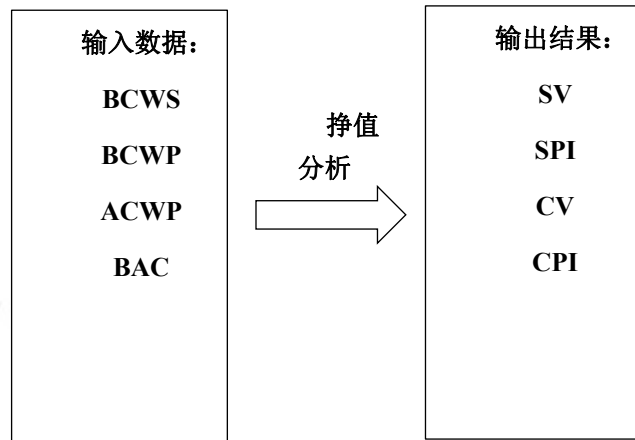


- 项目经理的一个重要的任务是随时掌握项目当前的进行状态，识别出潜在的风险或者延期的征兆并快速进行应对。
- 目标是使得每个工作包对应的曲线与图中的虚线尽量接近。
- 项目经理还要及时安排空闲的资源并进行优化。

- 项目历史的完善对一个学习型和持续改进型企业是重要的里程碑和宝贵的资源。

挣值分析模型

- 挣值分析（**Earned Value**）是对项目实施的进度、成本状态进行绩效评估的有效方法，是计算实际花在一个项目上的工作量成本，以及预计该项目所需成本和完成该项目的日期的一种方法。
- 挣值的概念，即到目前为止项目实际完成的价值



输入参数定义



- **BCWS (Budgeted Cost of Work Scheduled)** 计划完成工作的预算成本:是到目前为止的总预算成本。它表示“到目前为止原来计划成本是多少”或者说“到该日期为止本应该完成的工作是多少”，它是根据项目计划计算出来的。
- **ACWP (Actual Cost of Work Performed)** 已完成工作的实际成本:是到目前为止所完成工作的实际成本，它说明了“到该日期为止实际花了多少钱”，可以由项目组统计。
- **BCWP (Budgeted Cost of Work Performed)** 已完成工作的预算成本，又称挣值:是到目前为止已经完成的工作的原来预算成本，它表示了“到该日期为止完成了多少工作?”
- **BAC (Budgeted At Completion)** 工作完成的预算成本:是项目计划中的成本估算结果，是项目完成的预计总成本

挣值分析

- 如果项目一切正常的话，**ACWP** , **BCWP** , **BCWS**应该重合或接近重合。
- 有了这些采集的数据信息，可以使用挣值分析模型来分析输出结果，结果指标的计算为：
 - **进度偏差 SV (Schedule Variance) = BCWP-BCWS**，若此值为零，表示按照进度进行；如果为负值，表示项目进度落后；如果为正值，表示进度超前。
 - **成本偏差 CV (Cost Variance) = BCWP-ACWP**，若此值为零，表示按照预算成本进行；如果为负值，表示项目超出预算成本；如果为正值，表示低于预算成本。
 - **进度执行指标 SPI (Schedule Performance Index) = BCWP/BCWS**，指项目挣值与计划值之比。当SPI>1时，表示进度超前；当SPI=1时，表示实际进度与计划进度相同；当SPI<1时，表示进度延误。
 - **成本执行指标 CPI (Cost Performance Index) = BCWP/ACWP**，项目挣值与实际费用之比。当CPI>1时，表示低于预算，即实际费用低于预算费用；当CPI=1时，表示实际费用与预算费用吻合；当CPI<1时，表示超出预算，即实际费用高于预算费用。

参数取值原则



- 对于BCWS、BCWP、ACWP这些基本参数值，可以按照一定的时间段来统计计算，比如每周、每个月等。
- 挣值BCWP的计算是不容易的，需要考虑的原则有50/50规则和0/100规则，或者其他的经验加权法等。
 - **50/50规则**是当一项工作任务已经开始，但是没有完成时，我们就假定已经实现50%的价值，当这个工作任务全部完成的时候才实现全部的价值。
 - **0/100规则**是当一项工作任务开始，只要没有完成时，不产生任何价值，即是0，直到完成时才实现全部的价值

挣值分析举例

- 有4个工作任务，T1全部完成，T2完成一半，T3刚开始做，T4还没有开始做，截止到目前的各个指标如表

工作任务	截止到目前的 BCWS（元）	实际的 ACWP（元）	任务总预算（元）
T1	2000	1900	2000
T2	900	750	1600
T3	200	300	1000
T4	0	0	1200

挣值分析举例

- 根据**50/50**和**0/100**规则可以分别计算出**BCWP**

工作任务	截止到目前的 BCWS (元)	实际的 ACWP (元)	BCWP (元) (50/50 规则)	BCWP (元) (0/100 规则)
T1	2000	1900	2000	2000
T2	900	750	800	0
T3	200	300	500	0
T4	0	0	0	0
合计	3100	2950	3300	2000

- 按照**50/50**规则计算出的 $SPI=3300/3100=1.06$ ， $CPI=3300/2950=1.12$ ，可以看出二者都比1大，表示项目进度超前，成本花销低于预算，是一个比较好的项目执行。注意，如果比1大的过多，说明项目计划的预算不合理，过高地估计了一些任务计划预算，需要调整。
- 按照**0/100**规则计算出的 $SPI=2000/3100=0.68$ ， $CPI=2000/2950=0.69$ ，可以看出二者都比1小，表示项目进度滞后，成本花销高于预算，项目执行需要调整。

项目偏差控制

- 项目的执行过程可以是按照预算内完成、低于预算完成或者超预算完成。
 - 设定一个**允许偏差范围**，对超出允许偏差范围的挣值分析结果，需要采取措施纠正偏差。
- 对**进度偏差SV**、**进度执行指标SPI**可以分析进度问题，
 - 如果滞后严重，可以分析开发人员资源使用情况，对正在做和未开始做的任务计划做出必要的调整，降低将来的偏差。
- 对**成本偏差CV**、**成本执行指标CPI**可以分析成本问题，
 - 如果成本超出当前预算，可以分析工作任务复杂性及其预算的合理性，对正在做和未开始做的任务计划做出必要的调整，降低将来的偏差。
- 同时对项目的**质量、风险、人员**等方面进行监控，只有它们的指标在计划的控制范围之内，项目的进度和成本控制才有意义。

软件质量保证



- 时间、成本与质量在项目管理中常常相提并论。在这三个方面找到均可以满意的模式，并遵守这种模式，持续地进行管理工作是质量管理的最终目标。
- 质量管理的学派和观点：
 - 戴明理论的核心是“目标不变、持续改善和知识积累”，预防胜于检验。
 - 朱兰理论的核心思想是适用性，适用性是通过遵守技术规范，使项目符合或者超过项目相关人及客户的期望。
 - 克鲁斯比理论的核心思想是质量定义符合预先的要求，质量源于预防，质量的执行标准是零缺陷，质量是用非一致成本来衡量的。
 - 田口玄一核心思想是应用统计技术进行质量管理，通过损失函数来决定产生未满足目标产品的成本。

质量管理



- 全面质量管理(TQM)是指通过全体员工的参与、改进流程、产品、服务和公司文化，达到在百分之百时间内生产百分之百的合格产品，以便满足顾客需求。
- 软件项目的**质量管理**指的是保证项目满足其目标要求所需要的过程，质量管理的关键是**预防重于检查**，事前计划好质量，而不是事后检查。

质量管理过程



- 质量计划
 - 软件质量计划过程是确定项目应达到的质量标准，以及决定如何满足质量标准的计划安排和方法。
- 质量保证
 - 质量保证是为了提供信用，证明项目将会达到有关质量标准，而开展的有计划、有组织的工作活动。它是贯穿整个项目生命周期的系统性活动，经常性地针对整个项目质量计划的执行情况，进行评估、检查与改进等工作，向管理者、顾客或其他方提供信任，确保项目质量与计划保持一致。
- 质量控制
 - 质量控制是确定项目结果与质量标准是否相符，同时确定消除不符的原因和方法，控制产品的质量，及时纠正缺陷的过程。质量控制是对阶段性的成果进行检测、验证，为质量保证提供参考依据。
 - 软件质量控制主要就是发现和消除软件产品的缺陷。消除缺陷是通过“评审”和“测试”这类质量控制活动来实现的。质量控制方法有技术评审、走查、测试、返工、控制图、趋势分析、抽样统计、缺陷追踪等

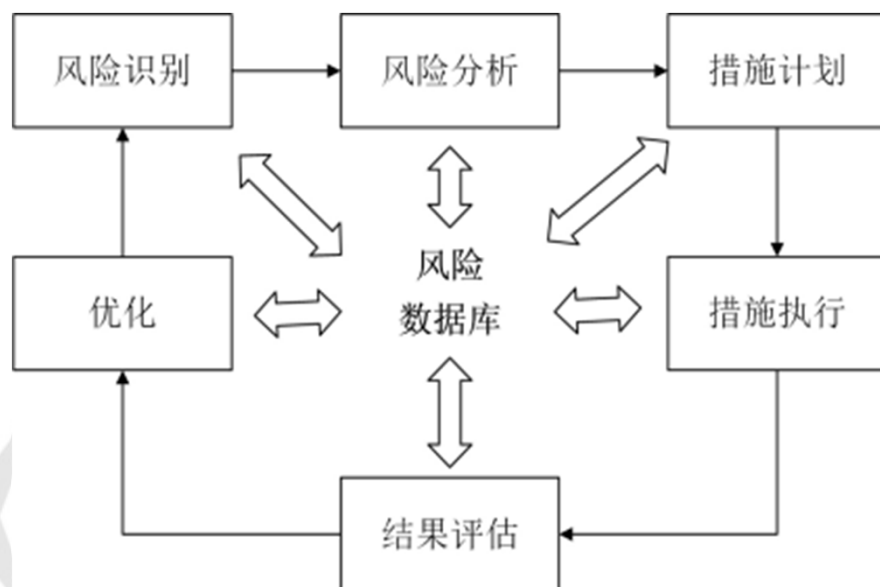
软件质量保证的内容



- 软件质量保证(SQA)包括：
 - (1) SQA过程；
 - (2)具体的质量保证和质量控制任务(包括技术评审和多层次测试策略)；
 - (3)有效的软件工程实践(方法和工具)；
 - (4)对所有软件工作产品及其变更的控制；
 - (5)保证符合软件开发标准的规程；
 - (6)测量和报告机制。
- 软件质量保证要素有：
 - **标准：**IEEE、ISO及其他标准化组织制定的软件工程标准和相关文件。要确保遵循所采用的标准，并保证所有的工作产品符合标准。
 - **评审和审核：**技术评审是由软件工程师执行的质量控制活动，目的是发现错误。审核是一种由SQA人员执行的评审，意图是确保软件工程技术遵循质量准则，

风险管理

- 软件风险是指软件开发过程中及软件产品本身可能造成的伤害或损失。
- 风险关注未来的事情，



- 风险识别：识别出项目开发过程中潜在的风险。
- 风险分析：对每个风险，需要确定两个影响因素：一是该风险出现的概率有多大，二是一旦出现风险，其破坏程度如何。可以将这两项值的乘积作为对该风险的优先级进行管理，评估可能的后果，并考虑相应的应对措施。
- 措施计划：对每个威胁的风险制定最小化其危害的应对计划，。
- 措施执行：按照计划执行应对措施。
- 结果评估：措施执行后在规定的的时间点，依据每项措施的评估指数验证该措施成功。
- 优化：对某项风险及其应对措施的优化。
- 风险数据库：组织级上建立的经验数据库

风险分析的结果数据样例



风险名称	类别	发生概率	影响程度	风险排序
用户变更需求	产品规模	80%	5	1
规模估算不准确	产品规模	70%	5	2
开发人员变动	人员数目及其经验	60%	4	3
客户对计划有异议	商业影响	50%	4	4
产品交付期限提前	商业影响	50%	3	5
软件负载超出计划	产品规模	30%	4	6
技术水平达不到	技术情况	20%	2	7
缺少对开发工具的学习	开发环境	40%	1	8
成员缺乏开发经验	人员数目及其经验	10%	3	9

项目人员



- 过程层面：包括企业中所有的过程定义及其描述。
- 能力：包括所有工作人员能够掌握和应用的所有技术和构件。
- 社交层面：包括员工间所有的交互情况、交互特点以及交互的质量。
- 工作环境：包括所有组织级的能够影响项目成败的制度和措施，包括分配的项目计算机到足够的办公用品。
- 项目的整体视图指出了一个领域的问题是不会在另外的领域中被解决的。
- 整体视图也表明不能通过过程模型使用清晰过程定义来弥补能力的缺失，因为只有先具有能力（**Know how**）才会使过程变得可用。

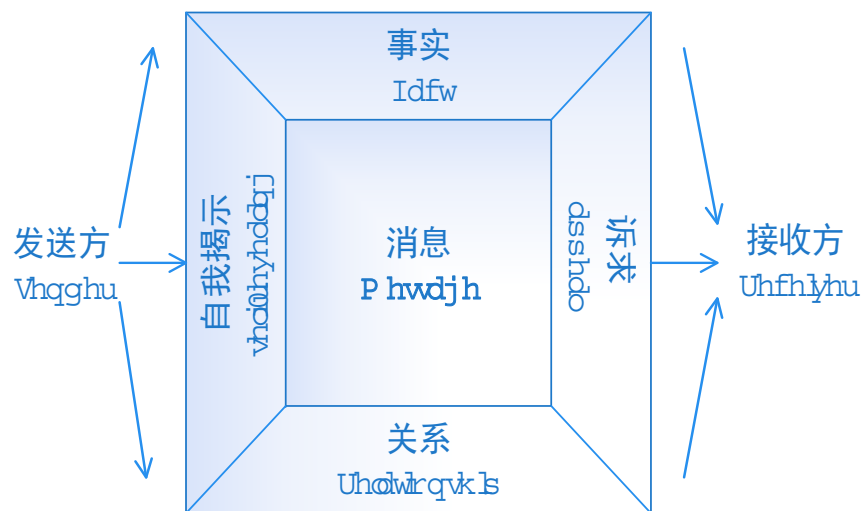
- 项目团队的组建过程：

- 组建（**forming**）：项目成员之间相互熟悉，彼此了解。团队对工作专注不够，效率低下。
- 风暴（**storming**）：团队花费大量的时间相互融合，包括职责的确定和工作的分配。团队不稳定，效率低下。
- 规范（**norming**）：团队形成规范，每个人开始关注怎样工作能最好的达到目标。团队工作效率得到提升。
- 行动（**performing**）：项目处于集中工作的稳定状态，每个成员知道自己的职责并了解与其它成员如何配合。团队工作的效率极高。

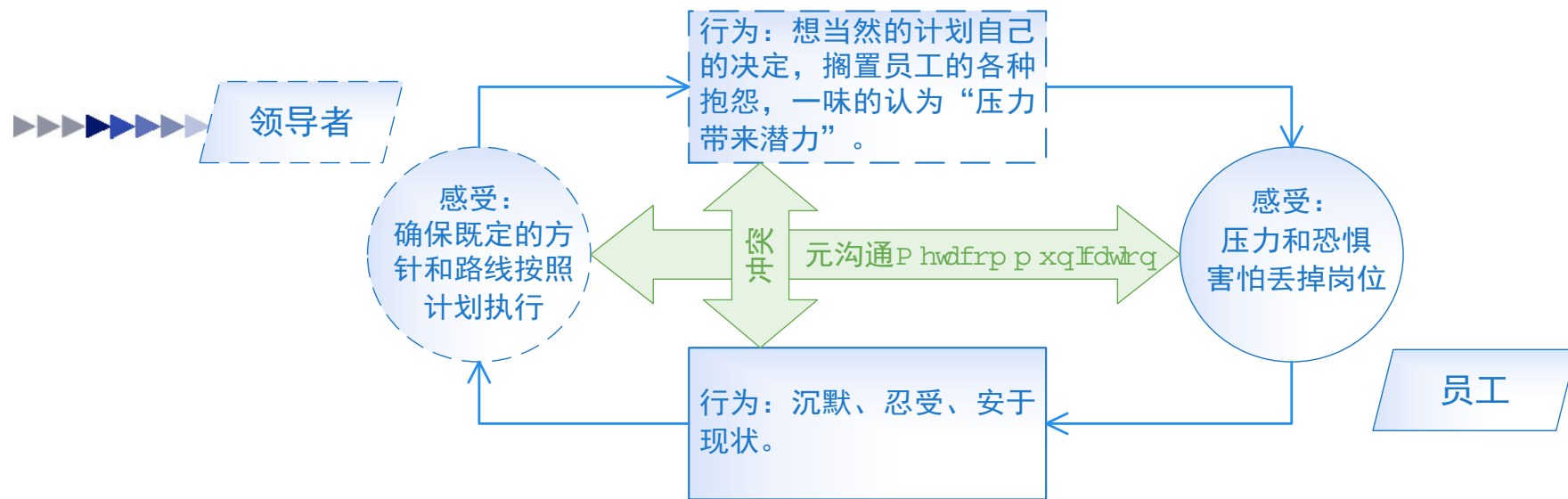
- 项目人员在项目中的角色确定：可将人员类型按照特点划分，不同类型的人员适合不同的角色，项目得益于若干不同人员类型的组合使用。

- 根据每种人员类型的特点，不是说每个人员只能安排一种角色与之对应，而是要发现哪些角色对某种人员的类型更偏重，哪些次之。

人员沟通



- 四方模型，又称为通信方（**communication square**）或四耳朵模型（**four-ears**）。
- 每条消息是由四个方面构成，强调的重点各有不同：事实（**fact**）、自我揭示（**self-revealing**）、关系（**relationship**）和需求（**appeal**）。



- 危机管理也是一项重要的沟通技巧，其中一部分内容就是要能够识别出恶性的循环沟通。
- 对这种循环模型的了解能够指导我们应多从通讯的“元层次”即感受而不是“行动层”进行改善，多关怀员工的实际感受强于生硬的命令。

作业



- 习题3、4

