第13章 软件项目管理

- 软件项目管理概述
- 项目估算
- 风险管理
- 进度管理
- 需求管理
- 配置管理

13.1 软件项目管理概述

管理目标

- **通常认为，项目成功的标志，也是项目管理人员争取的目标，应该包括以下几个方面。**

（1）达到项目预期的软件产品功能和性能要求。也就是软件产品达到了用户已认可的需求规格说明的要求。

（2）时限要求。项目应在合同规定的期限内完成。

（3）项目开销限制在预算之内。

管理涉及的范围

软件项目管理涉及的几个主要方面是人员、产品、过程和项目，即所谓4P（People、Product、Process、Project）。

（1）人员管理

美国卡内基·梅隆大学软件工程研究所的Bill Curtis在1994年发表了“人员管理能力成熟度模型”（people capability maturity model，P-CMM）。该模型力图通过吸引、培养、激励、部署和骋用高水平的人才来提升软件组织的软件开发能力。

管理涉及的范围

人员管理涉及：

① 共利益者。 包括：

- **项目的高级管理者——负责项目商务问题的决策；**
- **项目经理——负责项目的计划与实施以及开发人员的组织与管理；**
- **开发人员——项目开发的实施者；**
- **客户——提出需求并代表用户与开发人员交往的人员；**
- **最终用户——直接使用项目成果（产品）的人员。**

② 团队负责人。在小项目的情况下，项目经理就是团队负责人。而大型项目也许会有若干个设计、编程团队或是若干个测试团队。团队负责人除去负有团队日常工作的安排、组织和管理之外，还应特别注意发挥团队成员的潜能。

管理涉及的范围

③ **团队集体**。团队内部有分工是必要的，但必须很好地配合，做到步调一致，为此必须强调以下3点。

- 个人的责任心，这是团队完成工作的基本条件。
- 互相信任、尊重以及互相支持。
- 充分的交流与沟通。

管理涉及的范围

（2）产品管理

项目经理必须在项目开始时就明确项目的以下三个目标：

- **产品的工作环境。**
- **产品的功能和性能。**
- **产品工作处理的是什么数据，经它处理后得到什么数据。**

只有明确了项目的这些基本要求才能着手项目管理的各项工作，如项目估算、风险分析、项目计划的制定等。

管理涉及的范围

（3）过程管理

过程在软件工程项目中是重要的因素，它决定着项目中开展哪些活动以及对活动的要求和开展活动的顺序。

（4）项目管理

项目管理的任务是如何利用已有的资源，组织实施既定的项目，提交给用户适用的产品。

管理涉及的范围

项目管理要开展的主要工作可分为3类。

① 计划及计划管理。包括项目策划及计划制定、项目估算、风险分析及风险管理、进度管理、计划跟踪与监督。

② 资源管理。包括人员管理（人员安排、使用）、成本管理、信息管理。

③ 成果要求管理。包括需求管理、配置管理、质量管理。

13.2 项目估算

通常在项目的目标确定和软件基本功能确定之后，就应该着手项目计划的制定工作。**项目估算是制订计划的基础和依据**。

- **项目策划与项目估算**

项目策划是项目开展初期阶段的重要工作，其主要目标是得到项目计划，或者说计划（plan）是策划（planning）的结果。

13.2 项目估算

- 项目策划中需要开展的活动

(1) 确认并分析项目的特征。

(2) 选择项目将遵循的生存期模型，确定各阶段的任务。

(3) 确定应得到的阶段性工作产品以及最终的产品。

(4) 开展项目估算，包括估算产品规模、工作量、成本以及所需的关键计算机资源。

(5) 制订项目进度计划。

(6) 对项目风险进行分析。

(7) 制订项目计划。

13.2 项目估算

在项目估算中，要解决的问题是项目实施的几个主要属性，即将要开发产品的规模（size）、项目所需的工作量（effort）以及项目的成本（cost）。

13.2 项目估算

（1）**规模**。项目的规模指的是得到最终软件产品的大小。一般以编程阶段完成以后得到程序的代码行表示，如以1**千行代码**为单位，记为KLOC。

当然，在项目的开始只是对代码行的估计值。另一表示方法是**功能点**，记为FP，它是根据软件需求中的功能估算的。

13.2 项目估算

（2）**工作量**。项目的工作量按项目将要投入的人工来考虑，以一个人工作一个月为单位，记为“人月”。

（3）**成本**。软件项目的成本通常只考虑投入的人工成本，如某项目投入的总人工费用为12万元。

13.2 项目估算

- 成本计算

一个软件组织在完成多个项目以后积累了一些数据，进行成本分析后便可得到自己的生产率数值和人工价格。

- **生产率是平均每个人月完成的源程序行数，可记为KLOC/人月或FP/人月。**
- **人工价则为每人月的价值。**

有了这两个数值，如果在估出项目规模以后就可以很容易得到项目的工作量和成本，即

工作量=规模/生产率

成本=工作量×人工价

项目估算的功能点方法

- **功能点方法**（function point）简称FP方法，该方法克服了项目开始时无法得知源程序行数的实际困难，从软件产品的**功能度**（functionality）出发估算出软件产品的规模。

项目估算的功能点方法

1．功能度

功能点方法是以项目的需求规格说明中已经得到确认的软件功能为依据，着重分析要开发系统的**功能度**，并且认为，软件的大小与软件的功能度相关，而与软件功能如何描述无关，也与功能需求如何设计和实现无关。

项目估算的功能点方法

- **1．功能度**

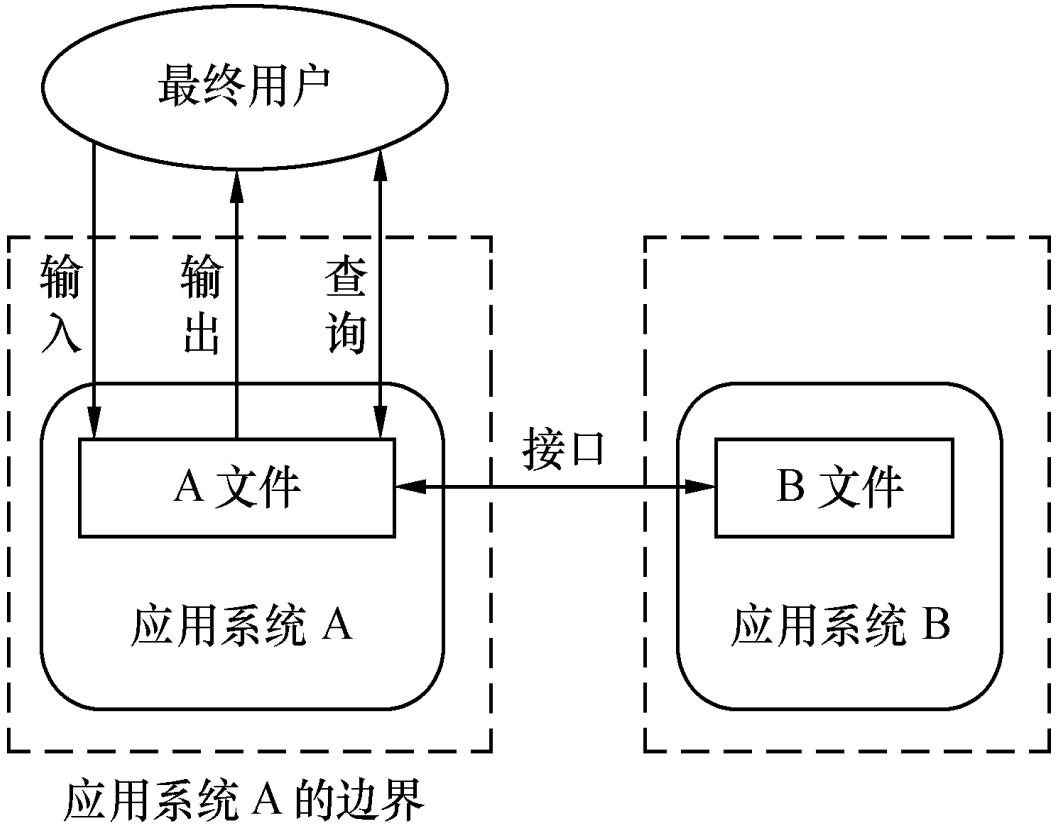
  为具体说明功能点方法，区分各种不同的功能，需要建立应用系统边界的概念。

- **应用系统边界把目标应用系统与用户和与其相关的应用系统分割开来。**
- **内部功能仅限于应用系统的边界之内，而外部功能则是跨边界的。**

项目估算的功能点方法

- 系统边界

图中系统A有4项功能都是跨越边界的，称其为外部功能。



项目估算的功能点方法

- 五种类型的功能：

（1）外部输入。外部输入处理那些进入应用系统边界的数据或是控制信息。经特定的逻辑处理后，形成内部逻辑文件。

（2）外部输出。外部输出处理离开应用系统边界的数据或控制信息。

（3）内部逻辑文件。是用户可识别的逻辑相关数据或控制信息组，它可在应用系统边界之内使用。内部逻辑文件代表应用系统可支持的数据存储需求。

项目估算的功能点方法

- 五种类型的功能：

（4）**外部接口文件**。外部接口文件是用户可识别的逻辑相关数据或控制信息构成的集合，该控制信息为应用系统所使用，却被另一应用系统所支持。外部接口文件代表应用系统外部支持的数据存储需求。

（5）**外部查询**。外部查询是唯一的输入/输出组合，它为实现即时输出引起所需数据的检索，代表了应用系统查询处理的需求。

项目估算的功能点方法

2．功能复杂性

软件项目每类功能的复杂程度可能各不相同，为表明功能复杂性的差别，将其分为简单的、中等的和复杂的3个等级。同时为表示其差异程度，分别给予不同的影响参数。下表列出了功能复杂性的影响参数值。

复杂性 / 功能度	简　单	中　等	复　杂
外部输入	3	4	6
外部输出	4	5	7
内部逻辑	7	10	15
外部接口	5	7	10
外部查询	3	4	6

项目估算的功能点方法

3．未调节功能点

只要能够从规格说明中得到了以上5种功能度的各级复杂性功能点的个数C，不难计算出未调节功能点的值。

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} \omega_{ij} C_{ij}$$

其中：i 代表功能度类型号；i=1，2，…，5；

j 代表复杂性的等级；j＝1，2，3；

ω_{ij}是第i类功能度和第j级复杂性的影响参数，即上表中第i行，第j列的参数值；

C_{ij}是第i类功能度和第j级复杂度功能点的个数。

项目估算的功能点方法

4．调节因子

任何软件都会有其自身特性，在考虑其各种自身特性时，从以下两个方面分解功能点计算的调节因子。

（1）影响因子。经过对各类软件的分析，综合出以下14种类型的影响因子：

- **数据通信**
- **分布数据处理**
- **性能目标**
- **系统配置要求**
- **事务率**
- **联机数据录入**
- **最终用户效率**
- **联机更新**
- **复杂的处理逻辑**
- **可复用性**
- **易安装性**
- **易操作性**
- **多工作场所**
- **设施变更**

项目估算的功能点方法

（2）**影响级**。上述影响因子对软件功能度的影响有多大必须加以区分，于是将影响因子的影响程度分为6级，即

0级 无影响　　1级 微小影响

2级 轻度影响　　3级 中度影响

4级 显著影响　　5级 重大影响

综合考虑14类影响因子的影响度N，应是将14种影响叠加起来，其值为0～70（14×5）。由此得到复杂度调节因子（complexity adjustment factor，CAF）

$$CAF=0.65+0.01N$$

其值应在0.65～1.35，其中基本调节常数是0.65，可见最大的调节量为35%。

项目估算的功能点方法

5．交付功能点

经过调节因子调节后的功能点值被称为**交付功能点**（delivered function point，DFP）

$$DFP=CAF\times UFP$$

6．交付功能点与软件规模

一些研究成果表明，上述计算出的功能点的值可以代表软件的规模，也可作为估算成本的依据。软件的规模可用**交付的源代码行数**（delivered lines of code，DLOC）来表示。

项目估算的功能点方法

功能点与DLOC的对应关系如下表所示。例如，

1 DFP相当于105 DLOC（COBOL程序）

1 DFP相当于128 DLOC（C程序）

语　言	DLOC	语　言	DLOC
Ada	71	Fortran	58
Algol	105	Ifam	25
Apl	32	Jovial	105
Assembcy	320	Microcode	107
Atlas	32	Pascal	91
Basic	64	PLI	80
C	128	Pride	64
CMS-2	178	SPLI	291
Cobol	105	High-Order	105
Compass	492	Machine	320
Coral-66	64	4^{th}-Generation	15
Flod	32	Interpretive	64

项目估算的功能点方法

7．功能点方法的优点

（1）DFP只与由规格说明得到的信息相关，而交付代码的行数若不通过功能点计算是不能直接从规格说明中得到的。

（2）DFP与实现软件的语言无关。

项目估算的功能点方法

8．功能点方法的不足之处

（1）针对需求规格说明进行分析时，主观因素难以完全排除，这包括：

- 对于规格说明，每人可能有不同的解释；
- 对于功能度的复杂性估计也可能因人而异；
- CAF计算时会有主观因素。

（2）非数据处理问题，如实时软件、系统软件、科学计算软件等功能点的上述计算方法并不适用。

（3）DFP的计算目前尚不能借助工具自动完成。

软件开发成本估算

1．专家判定——Delphi方法

专家判定技术就是由多位专家进行成本估算，取得多个估算值。有多种方法把这些估算值合成一个估算值，Read公司提出了Delphi技术，作为统一专家意见的方法。可得到极为准确的估算值。

软件开发成本估算

标准Delphi技术的步骤：

① 组织者发给每位专家一份软件系统的规格说明书（略去名称和单位）和一张记录估算值的表格，请他们进行估算。

② 专家详细研究软件规格说明书的内容，然后组织者召集小组会议，在会上，专家们与组织者一起对估算问题进行讨论。

软件开发成本估算

- **标准Delphi技术的步骤：**

③ 各位专家对该软件提出3个软件规模的估算值，即

a_i——该软件可能的最小规模（最少源代码行数）；

m_i——该软件最可能的规模（最可能的源代码行数）；

b_i——该软件可能的最大规模（最多源代码行数）。

无记名地填写表格，并说明做此估算的理由。

软件开发成本估算

标准Delphi技术的步骤：

④ 组织者对各位专家在表中填写的估算值进行综合和分类，做以下事情。

- 计算各位专家（序号为i，i=1，2，…，n）的估算期望值E_i和估算值的期望中值E。

$$E_i = \frac{a_i + 4m_i + b_i}{6}; \qquad E = \frac{1}{n}\sum_{i=1}^{n} E_i$$

（n 为专家人数）

- 对专家的估算结果进行分析。

软件开发成本估算

标准Delphi技术的步骤：

⑤ 组织者召集会议，请专家们对其估算值有很大差异之处进行讨论。专家对此估算值另做一次估算。

⑥ 在综合专家估算结果的基础上，组织专家再次无记名地填写表格。

软件开发成本估算

从步骤④到步骤⑥适当重复几次，最终可获得一个得到多数专家共识的软件规模（源代码行数）。

最后，通过与历史资料进行类比，根据过去完成项目的规模和成本等信息，推算出该软件每行源代码所需成本。然后再乘以该软件源代码行数的估算值，得到该软件的成本估算值。

软件开发成本估算

2．COCOMO模型

软件工程专家Barry Boehm在其著作《软件工程经济学》中提出了软件估算模型层次结构，称为构造式成本模型COCOMO（COnstructive Cost MOdel），也许这是在软件界影响最为广泛、最为著名的估算模型。

软件开发成本估算

(1) 3种类型的软件

COCOMO是针对Boehm划分的3种类型软件进行估算的。

① 固有型（organic mode）项目。规模较小，较为简单的项目，开发人员对项目有较好的理解和较为丰富的工作经验。

② 嵌入型（embedded mode）项目。这类项目的开发工作紧密地与系统中的硬件、软件和运行限制联系在一起，如飞机的飞行控制软件。

③ 半独立性（semi-detached mode）项目。项目的性质介于上述两个类型之间，其规模与复杂性均属中等，如事务处理系统，数据库管理系统等。

软件开发成本估算

（2）COCOMO的3级模型

① 基本COCOMO模型（basic model）。

- **该模型为静态、单变量，以估算出的源代码行数计算。**
- **开发工作量**

$$E = a_b(\mathrm{KLOC})^{b_b}$$ **（人月）**

其中，KLOC为交付的千行代码数。a_b、b_b为模型系数。

软件开发成本估算

- **开发周期**

$$D = c_b d_b E^{d_b} \quad \text{(月)}$$

- **系数**

基本COCOMO模型系数如下表所示。

项 目 类 型	a_b	b_b	c_b	d_b
固有型	2.4	1.05	2.5	0.38
嵌入型	3.6	1.20	2.5	0.32
半独立型	3.0	1.12	2.5	0.35

软件开发成本估算

② 中级COCOMO模型（intermediate）

- 该模型除考虑源代码行数外，还考虑调节因子（EAF），用其体现产品、软件、人员和项目等因素。
- 开发工作量

$$E = a_i (\mathrm{KLOC})^{b_i} \times \mathrm{EAF}$$

- 系数

项 目 类 型	a_i	b_i
固有型	3.2	1.05
嵌入型	2.8	1.20
半独立型	3.0	1.12

软件开发成本估算

- **调节因子EAF（effort adjustment factor）。包含了4类15种属性，其值为0.7～1.6。**

属性		非常低	低	正常	高	非常高	超高
产品属性	软件可靠性	0.75	0.88	1.00	1.15	1.40	
	数据库规模		0.94	1.00	1.08	1.16	
	产品复杂性	0.70	0.85	1.00	1.15	1.30	1.65
硬件属性	执行时间限制			1.00	1.11	1.30	1.66
	存储限制			1.00	1.06	1.21	1.56
	模拟机易变性		0.87	1.00	1.15	1.30	
	环境变更		0.87	1.00	1.07	1.15	
人员属性	分析员能力		1.46	1.00	0.86		
	应用领域经验	1.29	1.13	1.00	0.91	0.71	
	程序员能力	1.42	1.17	1.00	0.86	0.82	
	虚拟机[1]使用经验	1.21	1.10	1.00	0.90	0.70	
	程序语言使用经验	1.41	1.07	1.00	0.95		
项目属性	现代程序技术	1.24	1.10	1.00	0.91	0.82	
	软件工具使用	1.24	1.10	1.00	0.91	0.83	
	开发进度限制	1.23	1.08	1.00	1.04	1.10	

(1) 虚拟机是指为完成某一软件任务所使用软、硬件的结合体。

软件开发成本估算

③ 高级COCOMO模型（advanced）

- 高级COCOMO模型除保留中级模型的因素外，还涉及软件工程过程不同开发阶段的影响，以及系统层、子系统层和模块层的差别。
- 软件可靠性在子系统层各开发阶段有不同的调节因子。

开发阶段 / 可靠性级别	需求和产品设计	详 细 设 计	编程及单元测试	集成及测试	综　　合
非常低	0.80	0.80	0.80	0.60	0.75
低	0.90	0.90	0.90	0.80	0.88
正常	1.00	1.00	1.00	1.00	1.00
高	1.10	1.10	1.10	1.30	1.15
非常高	1.30	1.30	1.30	1.70	1.40

13.3 风险管理

- 什么是软件风险
- 风险管理的任务
- 风险评估
- 风险控制

13.3.1 软件风险

- 软件工程过程中可能出现的那些影响软件目标实现或是可能造成重大损失的事件称为软件风险。
- 在软件开发项目的最初阶段，确认需求对整个开发工作是至关重要的。软件开发项目经常遇到的一个严重问题就是用户的需求一变再变。可以说这是一个典型的软件风险。
- 软件工程项目所需的主要资源是合格的人员，有不少软件项目可能出现合格人员短缺的现象，这对于达到项目的目标自然构成威胁。

13.3.1 软件风险

- **风险的特点**
  - **可能发生的事件。风险是可能发生的事件，其发生的可能性用风险概率来描述。**
  - **会给项目带来损失的事件。**
  - **可能对其加以干预，以期减少损失。针对每一种风险，我们应弄清可能减少造成损失或避免损失的程度。对风险加以控制，采取一些有效的措施来降低风险或是消除风险。**

13.3.1 软件风险

- **风险分类**
  - **依据危害性分类**

从危害到软件项目本身讲，软件风险可分为3类：

1）成本风险。成本风险是项目预算和开销不够准确造成的。

2）绩效风险。绩效风险是系统不能提供全部或某些预期效益，或是不能实现预期的软件需求。

3）进度风险。进度风险关系到项目进度或项目达到指定里程碑的不确定性。

13.3.1 软件风险

- **风险分类**
  - **依据范围分类**

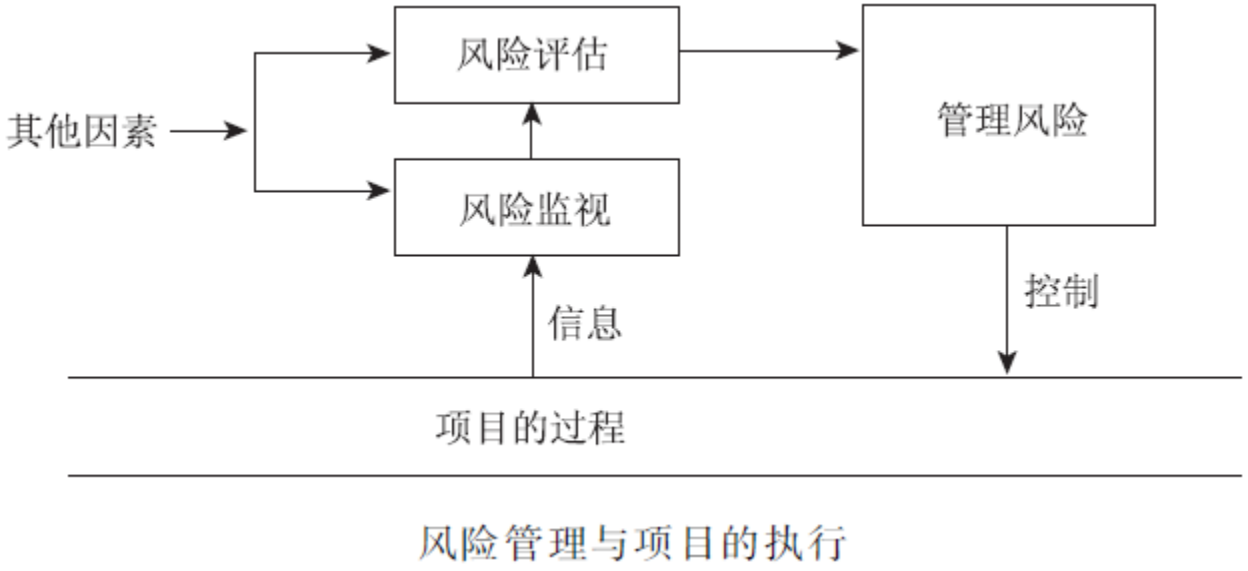
1）项目风险。这种风险涉及预算、成本、进度、人员的招聘和组织、资源的获取，以及顾客和需求等方面的问题。
2）技术风险。技术风险威胁着开发产品的质量和交付产品的时间。技术风险会涉及设计方案、实现、接口、验证以及维护等方面的问题。
3）商业风险。商业风险的发生会威胁开发软件的生命力，它会危及软件项目和产品出路。如市场风险、策略风险、管理风险、预算风险。

13.3.2 风险管理的任务

- **风险管理的目标**
  - 识别风险。识别风险是要找出可能的风险，对其进行分析、评估，并进一步对这些风险排序，以突出最为险恶的风险。
  - 采取措施，把风险造成的影响降低到最小。
- **风险管理的策略**
  - 回避风险。例如改变项目的某些功能或性能需求使风险不可能发生。
  - 转移风险。把风险转移到其他系统，或是借助购买保险将经济损失转移，从而化险为夷。
  - 承受风险，接受风险，但将风险损失控制在项目资源可承受的范围之内。

13.3.2 风险管理的任务

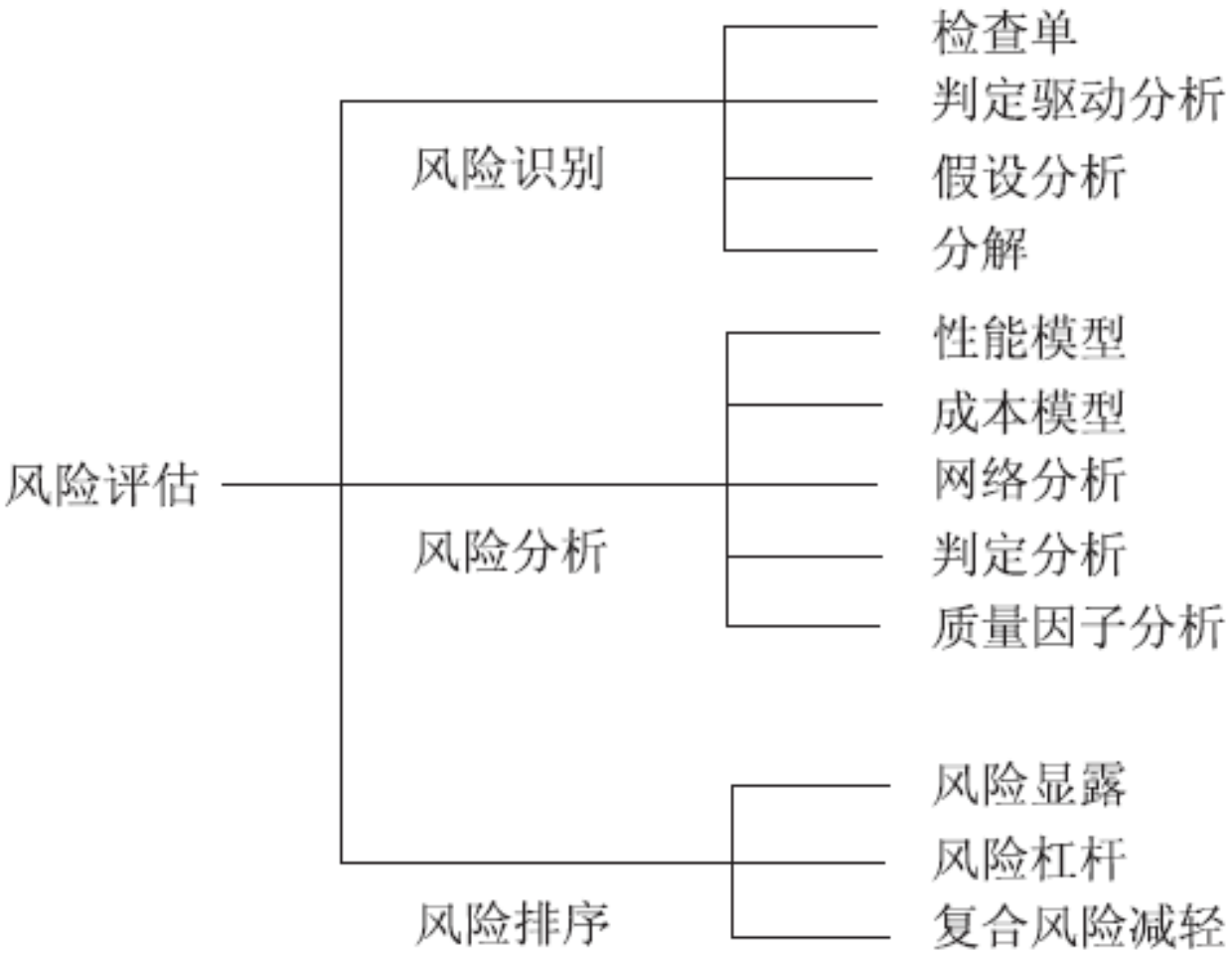
- **风险管理活动**
  - **为达到风险管理的目标，必须使风险管理活动围绕着风险评估和风险控制开展。实施风险管理可以将其融入开发过程，如软件开发的螺旋模型本身就包含有风险管理。**



风险管理与项目的执行

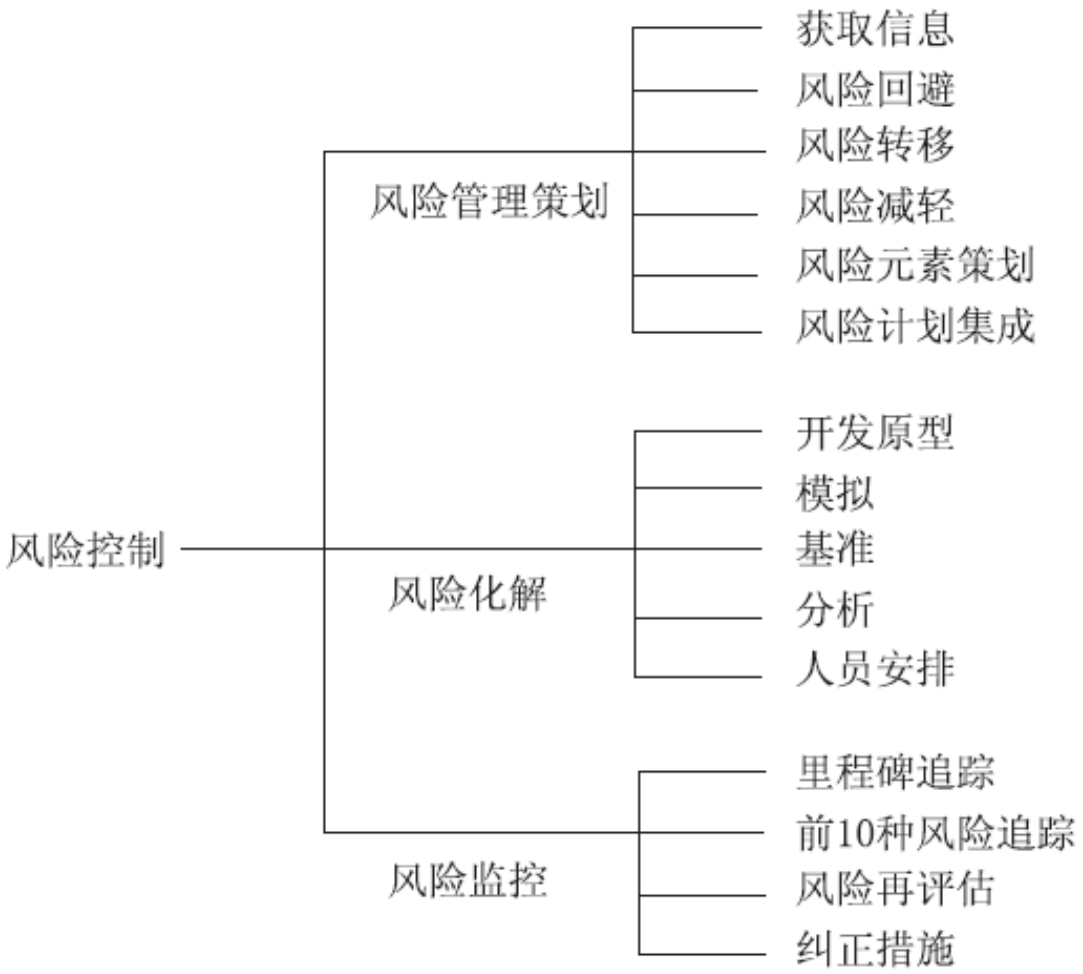
13.3.2 风险管理的任务

- **风险管理活动**



13.3.2 风险管理的任务

- **风险管理活动**



13.3.3 风险评估

- **风险评估的目标是认识可能的风险，它应该是风险控制的前提。**
- **风险评估通常包括：风险识别、风险分析和风险排序3个方面的内容**

13.3.3 风险评估

- **风险识别**

  - 风险识别是风险评估的第一步。就某个特定的软件工程项目来说，从项目的具体情况出发，列举出可能出现的风险，真正弄清每一可能风险的情况是风险识别的主要任务。
  - 检查单(checklist)是识别风险的有力工具。采用检查单来识别风险是将检查单中所列举的各种风险，对照即将开发的软件项目，逐一加以甄别，判定检查单中哪些风险在该项目中可能发生。
  - 在进行风险识别时采用访谈、调查还是会议的方式，或是对计划、过程和工作产品进行评审的方式，均应根据软件项目的具体情况决定。

13.3.3 风险评估

- **风险分析**
  - **风险分析的任务是分析每个风险可能造成的影响，给出风险大小的量值。**
  - **进行分析可以借助一些已有的模型，但也并非所有已列出的风险都可借助模型进行分析，因此常常采用主观分析。**
  - **风险分析方法包括：COCOMO成本模型，判定分析，网络分析，质量因子分析，性能分析等。**

13.3.3 风险评估

- **风险排序**
  - **识别并分析风险将使我们初步弄清可能妨碍达到项目目标的危险事件，然而各种风险的后果会有很大的差别。**
  - **我们必须对其加以区别，以便把管理者的目光集中到最高风险的事件上。这里所说的风险高低是以风险显露造成损失的大小来衡量，损失大的风险自然应该给予更多的重视。**

13.3.3 风险评估

- **风险概率：**风险概率指的是风险事件出现的可能性，显然可能性大的事件其风险概率值较高。
- 可以把各种概率值的风险划分为3类：低概率风险、中概率风险和高概率风险。

3 类风险的概率

概　率	取 值 范 围
低	0.0～0.3
中	0.3～0.7
高	0.7～1.0

13.3.3 风险评估

- **风险影响：**不同的风险在发生时会造成多大的影响各不相同，影响的大小需要加以度量，例如可以用损失的金额数来衡量。但为了简单和直观。可以把风险影响分为4个等级，并按1~10来赋值。

风险影响级别

影　响　级	取 值 范 围
低	0～3
中	3～7
高	7～9
甚高	9～10

13.3.3 风险评估

- **风险排序的步骤**

1）对已识别和分析了的风险估计概率的类别，判断其属于高概率风险，还是中概率风险，或是低概率风险。如有必要，给出其概率值的大小。

2）评估每个风险对项目的影响级，如为低级、中级、高级或甚高级。如有必要给出其影响级的加权值。

3）风险排序应根据该项目各有关风险的概率和风险影响。显然，具有高概率及高影响级的风险应该排在具有中概率和高影响级风险的前面。

4）针对排序列在前几位的风险采取缓解措施和跟踪措施。

13.3.3 风险评估

- **风险显露**

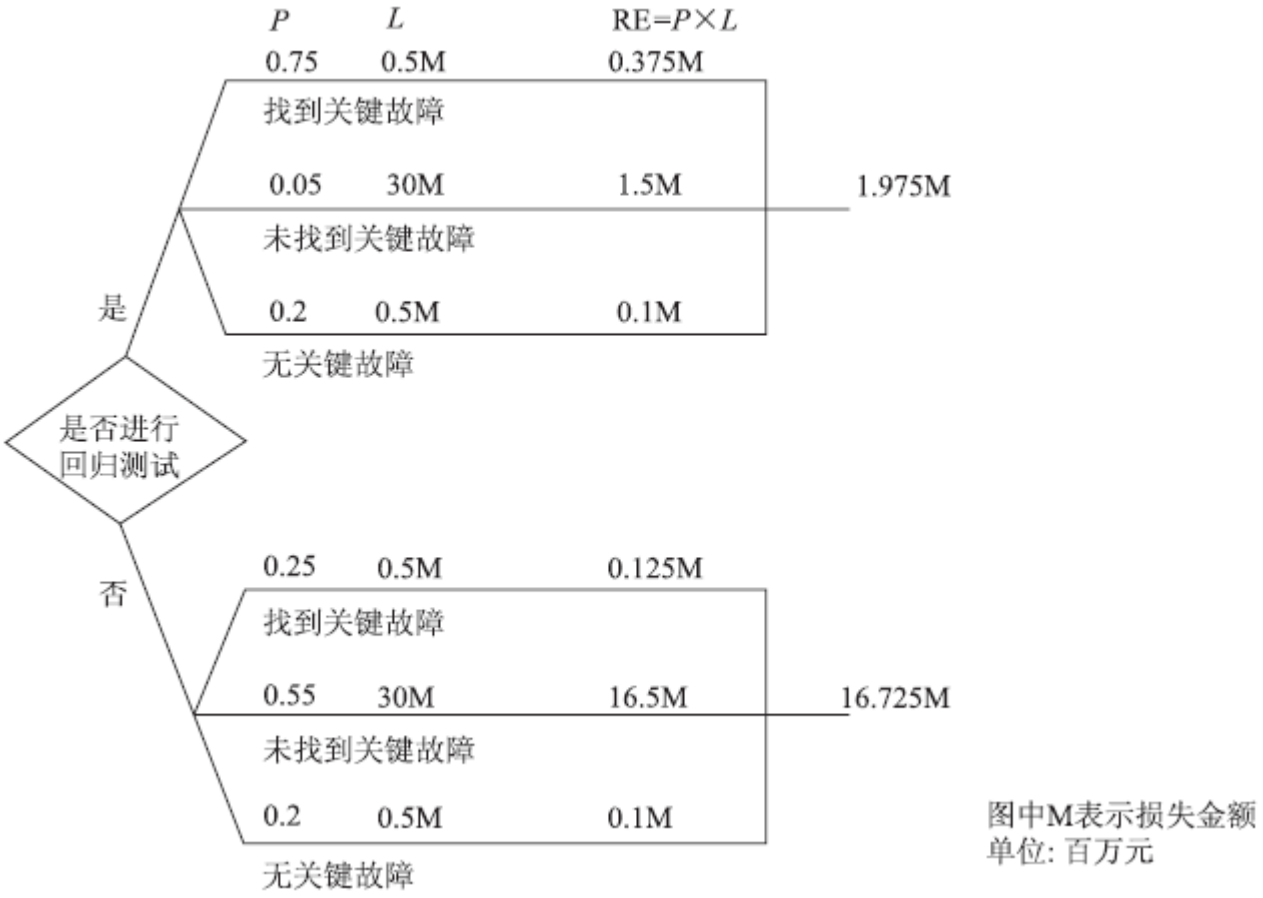
✓ 风险对项目威胁的大小可用风险显露来表示，它既和风险概率有关，也和风险发生造成损失的大小有关。

$$RE(R)=P(R)\times L(R)$$

其中，RE(R)是风险R的发生可能给项目造成的损失；P(R)是风险R发生的概率；L(R)是风险R如果发生会造成的损失。

13.3.3 风险评估

- **风险显露**
  - ✓ 计算实例：以回归测试为例。



13.3.4 风险控制

- **风险管理策划**

➢ **风险管理策划是要针对每个已经过识别和分析认为应该受控的风险制定风险管理计划。**

➢ **按Boehm的意见，风险管理计划主要包括以下5个方面：**

1）该项风险为什么重要，为什么一定要管理；
2）风险管理应该能够提供什么以及什么时候提供；
3）实施这些风险管理活动的责任人是谁；
4）风险怎样能够得到减轻，该采取什么措施；
5）需要什么资源。

13.3.4 风险控制

- **风险化解**
  - **风险化解是要实际消除风险或减轻风险。实施风险管理计划从根本上讲就是将风险化解。**
  - **为了帮助选择风险减轻的方法，必须考虑减轻风险的成本。我们把风险显露的损失差与风险减轻成本的比称为风险杠杆(risk leverage)。即**

$$\text{风险杠杆} = \frac{\text{风险减轻前的风险显露} - \text{风险减轻后的风险显露}}{\text{风险减轻成本}}$$

13.3.4 风险控制

- **风险化解**

➢ **印度Infosys公司根据许多软件项目的实践总结出10大风险及其相应的化解措施。 这10大风险包括：**

- ✓ 受过技术培训的人员不足
- ✓ 过多的需求变更
- ✓ 需求不明确
- ✓ 人员流失
- ✓ 外部因素对项目决策的影响
- ✓ 性能需求达不到要求
- ✓ 进度计划不切实际
- ✓ 面临新技术（软件或硬件）的挑战
- ➢ 商业知识不足
- ➢ 连接故障或性能迟钝

13.3.4 风险控制

- **风险监控**
  - **随时监控的必要性：**由于风险是一些概率事件，它经常依赖于外部因素。在外部因素改变以后，风险构成的威胁可能和以前的评估有很大的差别。显然，对风险的理解也要随时间改变，进而所采取的风险化解措施可能影响着对风险的认识。
  - **跟踪监控：**上述的风险动态特性表明，不应把项目的风险看成是静止不动的。必须定期对风险进行重新评估。

13.4 进度管理

- 进度控制问题
- 甘特图
- 时标网状图
- **PERT图**

进度控制问题

1．值得重视的现象

软件项目能否按计划的时间完成，及时提交产品是项目管理的一个重要课题。我们都希望按计划及时完成，但项目未能按预期的进度提交产品，延误工期的现象经常会出现。我们必须重视这一现象，分析其原因，并有针对性地
采取措施。

2．制订项目进度安排的条件

制订项目进度安排计划是为了实施，自然希望越准确，越符合实际越好，但是怎样才能做到这一点，需要在这以前做些工作，创造良好的条件，使得进度安排的确定是有根据的。

进度控制问题

这些条件包括以下7条：

（1）项目分解。无论多么大、多么复杂的项目都必须首先将其划分成能够管理的若干活动和若干任务，并且往往这种分解是多个层次的。

（2）确定各部分之间的相互关系。划分后的活动和任务按项目本身的要求，必定存在着一定的相互依赖关系，如谁先谁后，或是两者应该并行互不依赖等。

（3）时间分配。为每项活动和任务分配需要的时间，如需要多少人天的工作量。

进度控制问题

（4）**确认投入的工作量**。应确认按项目要求的人力投入工作量在实际工作中能够予以满足，而不致出现某些工作阶段人力投入不足的现象。

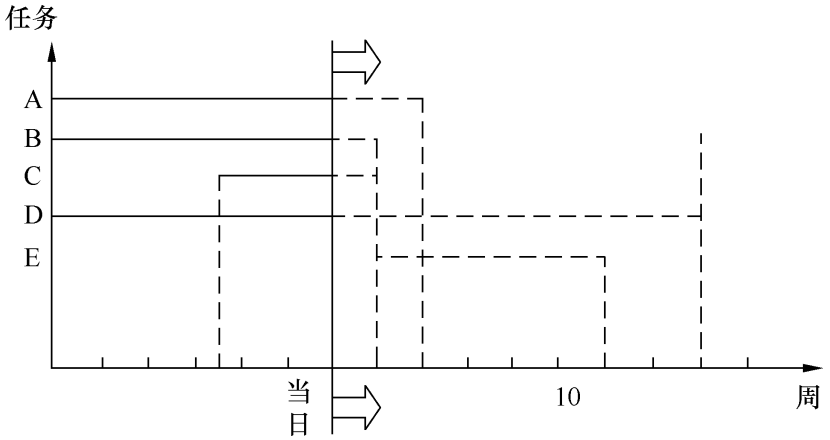
（5）**确定人员的责任**。

（6）**规定工作成果**。任何分配的任务都应给出符合要求的工作成果，它应该是整个项目的一个组成部分。

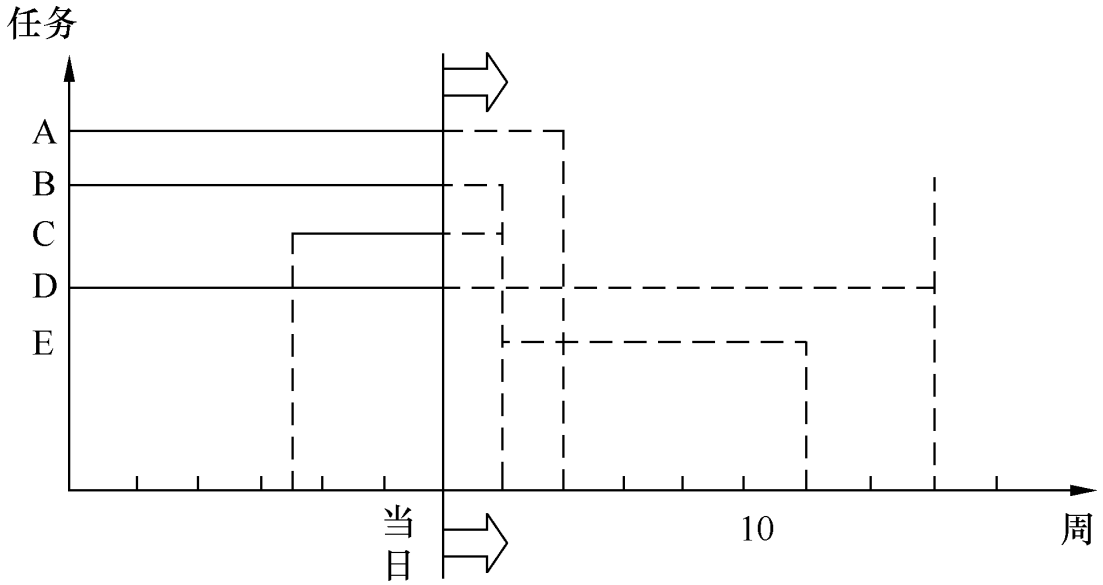
（7）**规定里程碑**。任何一项工作完成后需经过一定形式的检验，如经过评审或审核（批准）得到认可，被认为确已完成，表示一个里程碑已经完成。**里程碑**也被称为**基线**。

甘特图

- **甘特图**（Gantt chart）是表示工作进度计划以及工作实际进度情况最为简明的图示方法。
- 甘特图中横坐标表示时间，以水平线段表示子任务的工作阶段，可以为其命名。
- 线段的起点和终点分别对应着该项子任务的开工时间和完成时间，线段的长度表示完成它所需的时间，有实线和虚线之分，一开始做出各项子任务的计划时间，应该都以虚线表示。

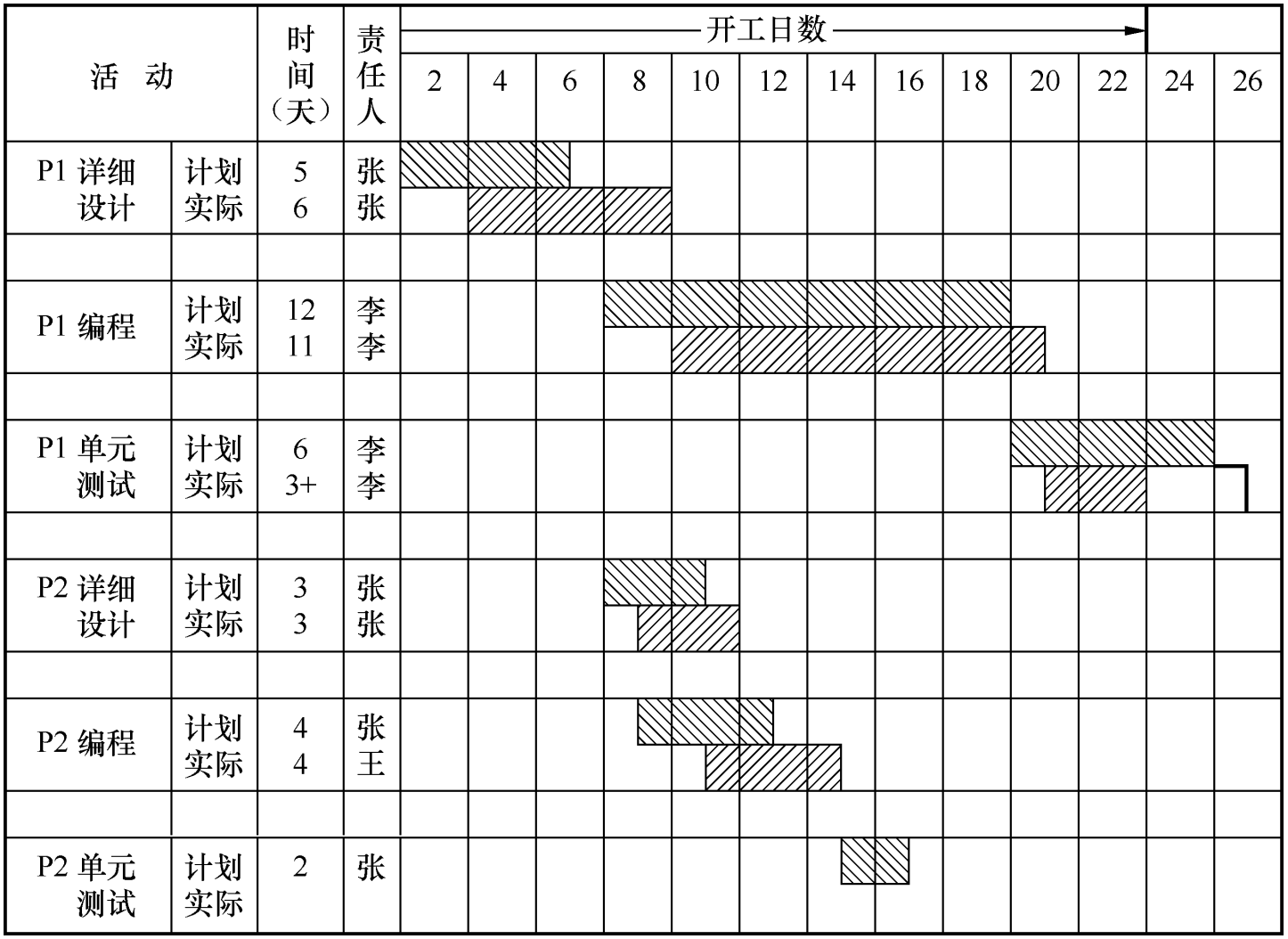


甘特图



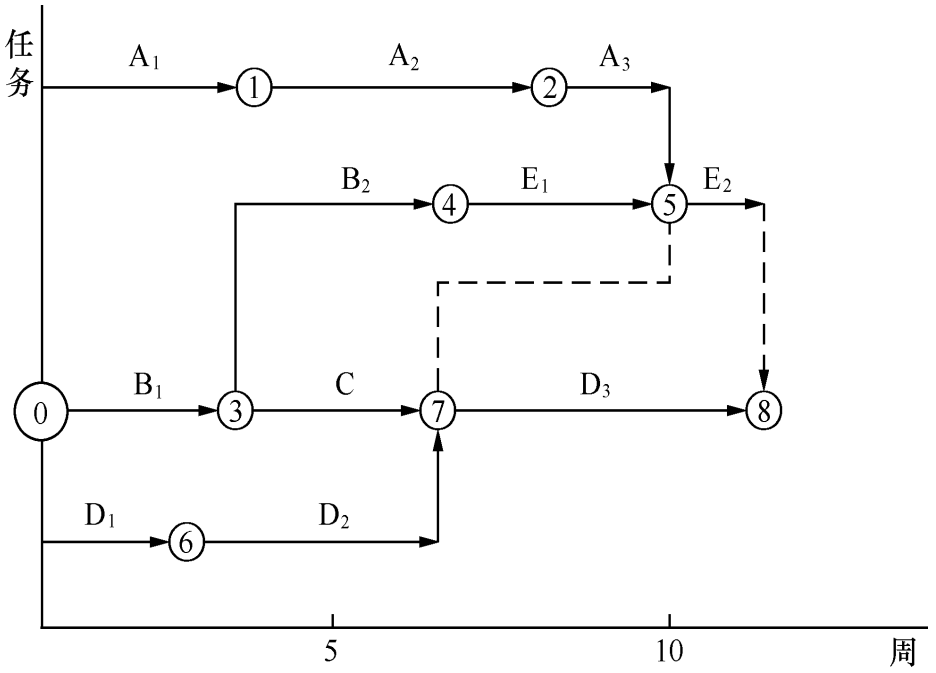
甘特图可以清楚地表示各项子任务在时间对比上的关系，但无法表达多个子任务之间更为复杂的衔接关系。

甘特图


活 动
时间（天）
责任人
开工日数
2
4
6
8
10
12
14
16
18
20
22
24
26
P1 详细设计
计划
实际
5
6
张
张
P1 编程
计划
实际
12
11
李
李
P1 单元测试
计划
实际
6
3+
李
李
P2 详细设计
计划
实际
3
3
张
张
P2 编程
计划
实际
4
4
张
王
P2 单元测试
计划
实际
2
张

时标网状图

为克服甘特图的缺点，将甘特图做了一些修改，形成了时标网状图（time Scalar network）。图中的任务以有向线段表示，其指向点表示任务间的衔接点，并且都给予编号，可以显示出各子任务间的依赖关系。它显示出比甘特图具有优越性。



PERT图

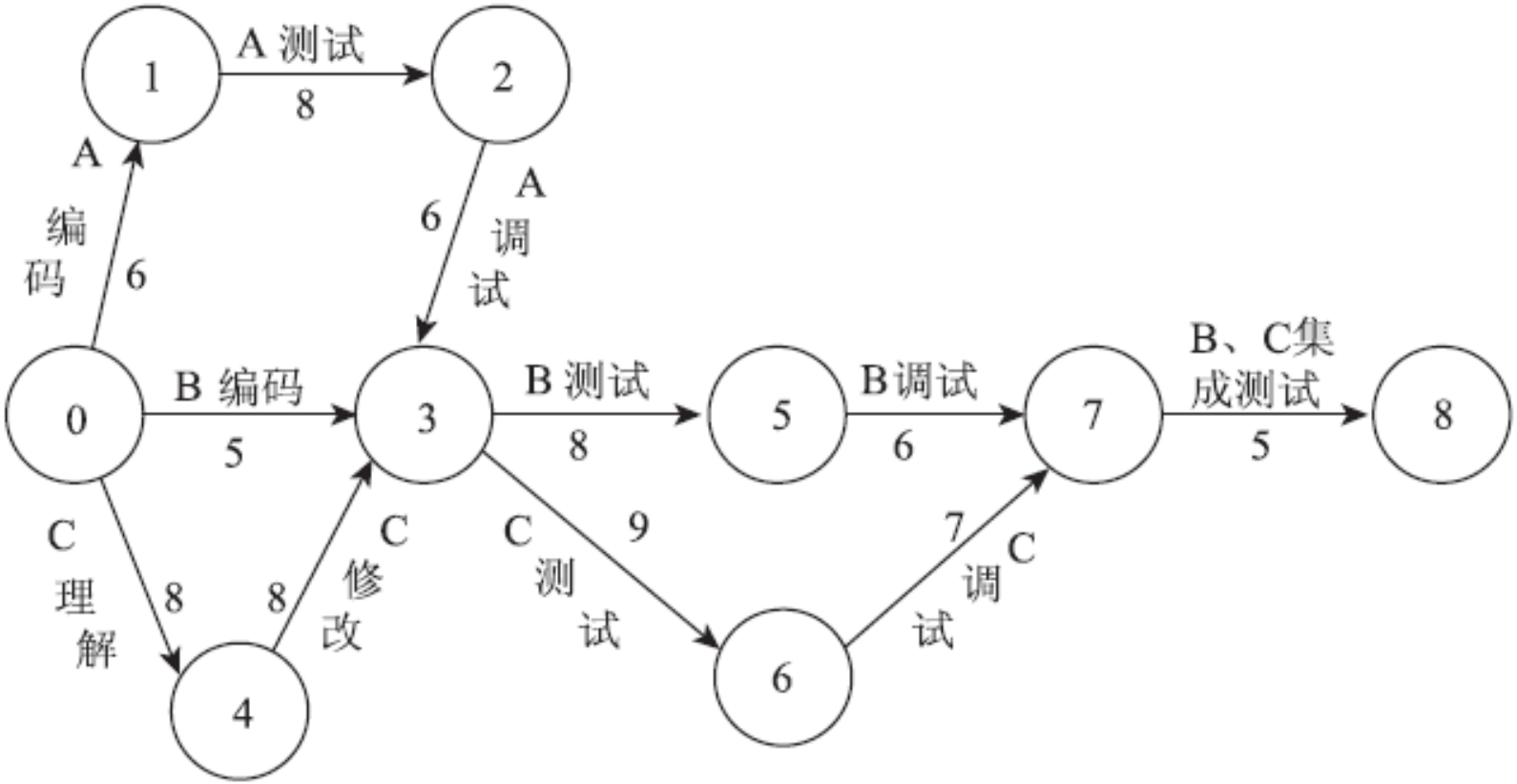
计划评审技术（program evaluation and review techique，PERT）也称网络图方法，或简称**PERT图方法**，它的另一名称是**关键路径法**（critical path method，CPM）。

PERT图

- PERT图中以有向的箭头作为边表示子任务，它是有名称（即子任务名）、有长度（即完成此项子任务所需的时间）的向量；
- 以有编号的圆圈作为结点，它应该是子任务向量的始发点或指向点；
- 由若干条边和若干个结点构成了网状图，于是我们可以沿相互衔接的子任务形成的路径，进行路径长度的计算、比较和分析，从而实现项目工期的控制。

实例

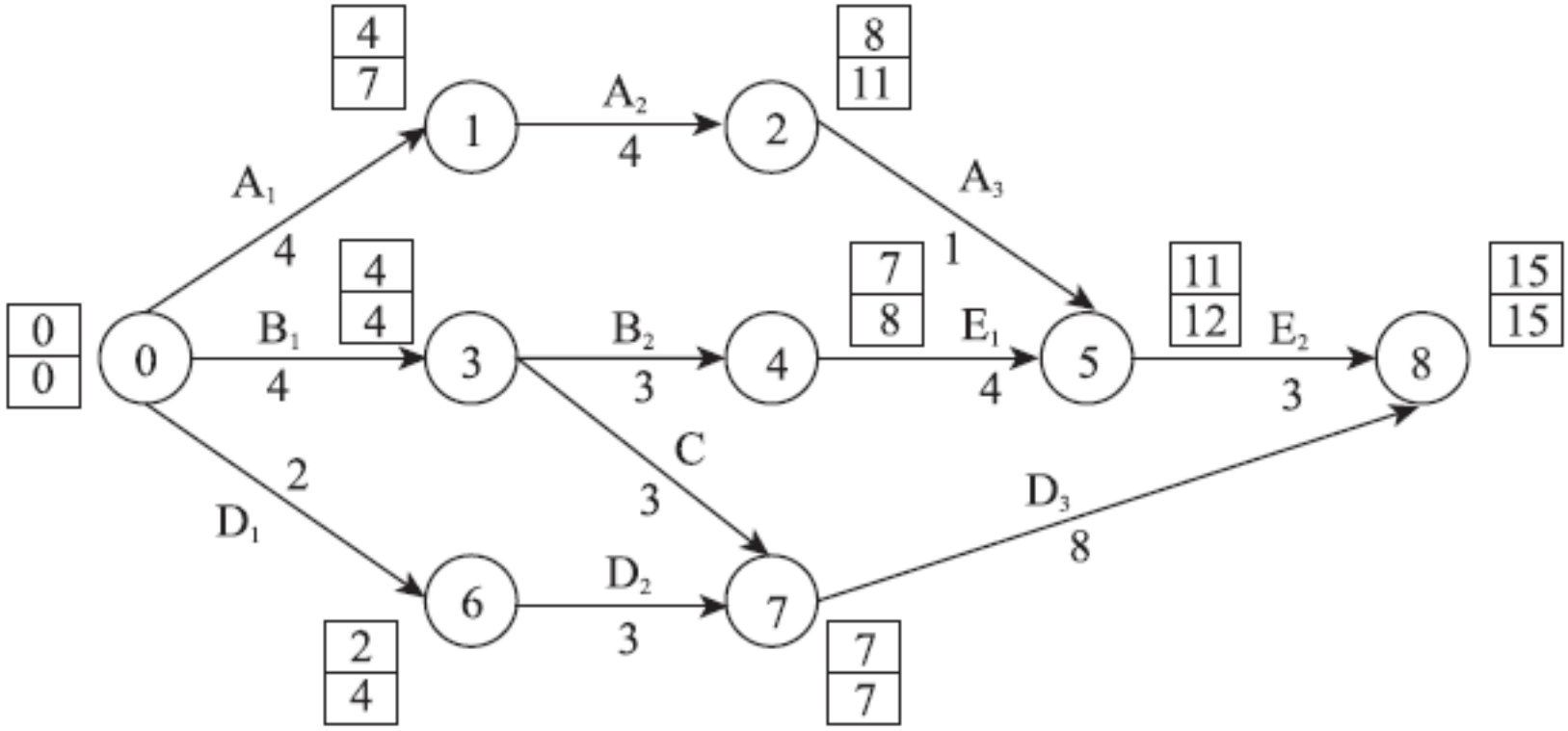
- **PERT图**



开发模块 A、B、C 的网状图

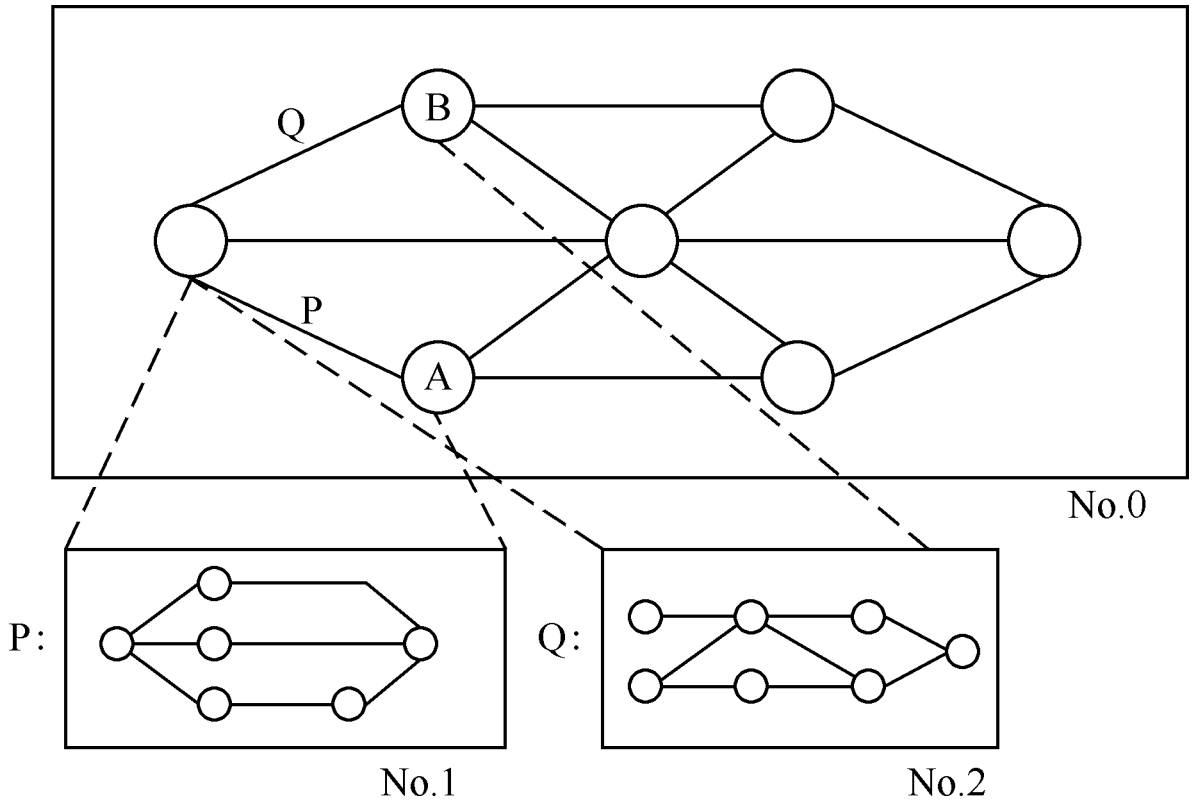
实例

- **PERT图**


0
0
0
4
7
1
8
11
2
4
4
3
7
8
4
11
12
5
15
15
8
A1
4
A2
4
A3
1
B1
4
B2
3
E1
4
E2
3
C
3
D1
2
D2
3
D3
8
6
2
4
7
7
7

实例

- 分层PERT图



13.5 需求管理

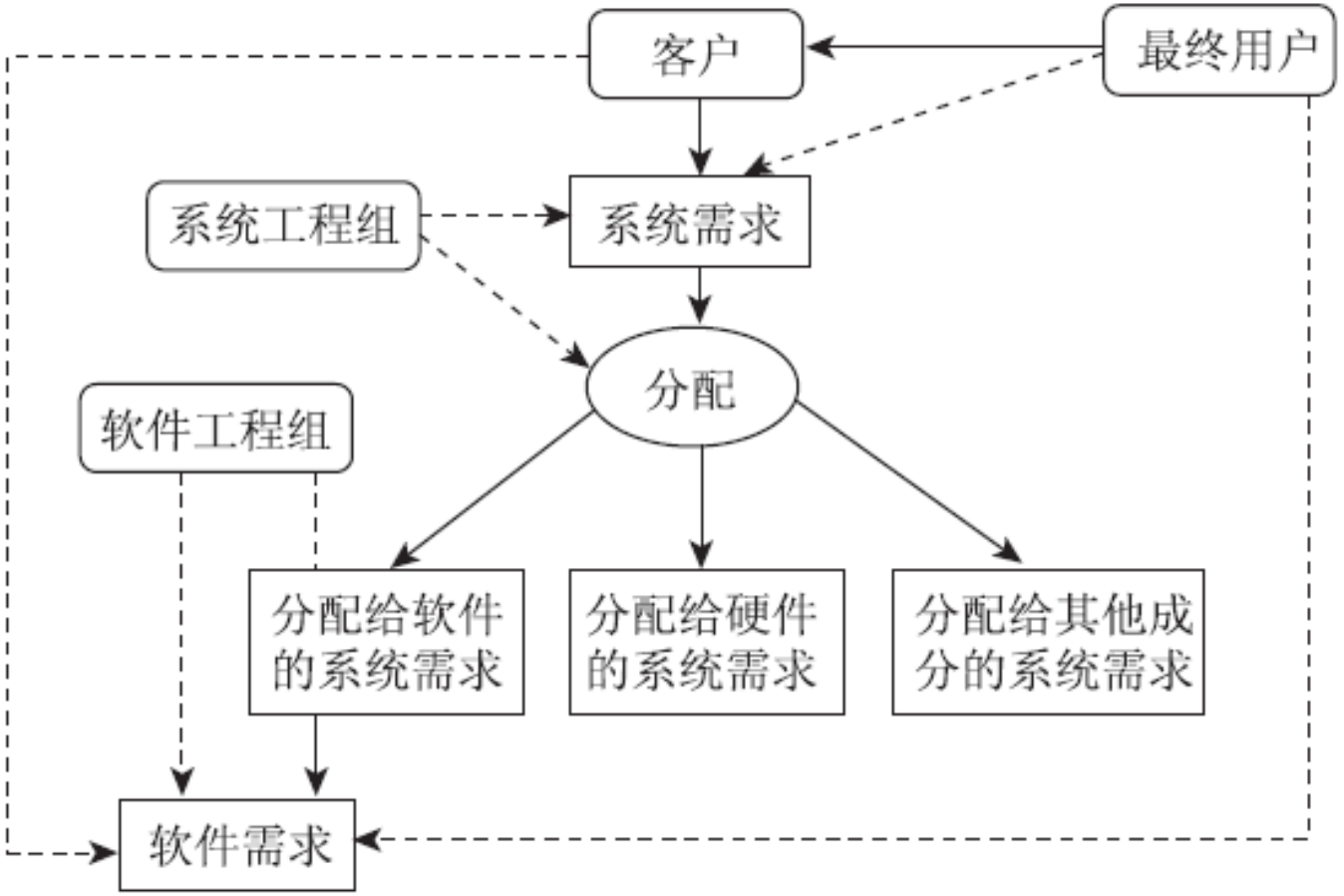
- **系统需求与软件需求**
- **需求工程**
- **需求变更**
- **需求变更控制**
- **可追溯性管理**

13.5.1 系统需求与软件需求

- **系统和系统需求分配**
- ✓ **系统：**通常考虑的系统是指基于计算机的系统或计算机控制的系统，它是在计算机控制下完成特定功能的系统。例如，航空管制系统、飞行器惯性导航系统、生产控制系统等。
- ✓ **系统需求分配：**系统工程组面对用户，负有开发系统的责任。系统工程组在从用户那里取得系统需求以后，应将系统需求进行分解，也就是把已确定的系统需求分配给系统的各个组成部分。

13.5.1 系统需求与软件需求

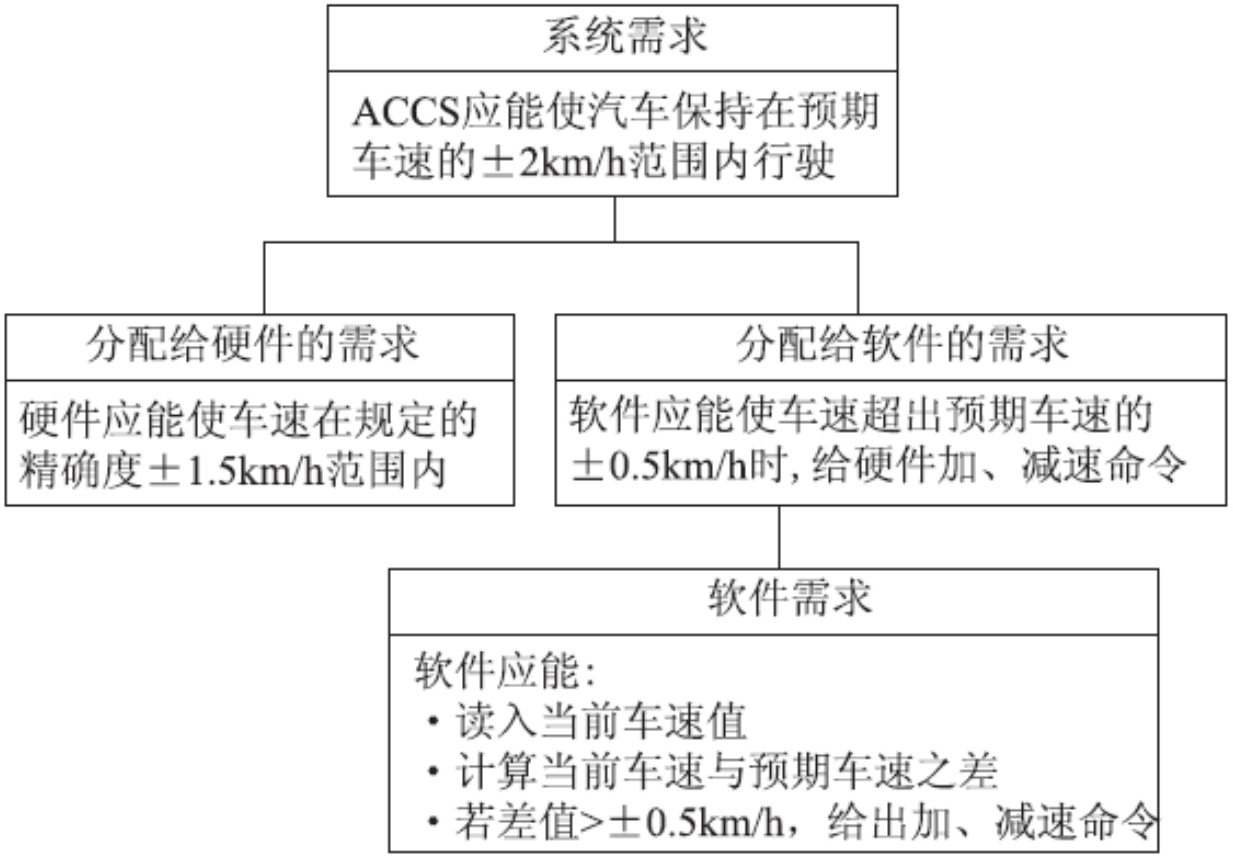
- **系统和系统需求分配**



系统需求分配

13.5.1 系统需求与软件需求

- **系统和系统需求分配**



汽车限速系统 ACCS 的需求分配

13.5.1 系统需求与软件需求

- **软件需求**

✓ 按IEEE STD 610标准的定义，软件需求是用户为解决某个问题或为实现某个目标，要求软件必须满足的条件或能力。

✓ 软件需求的3个层次如下：

1）**业务需求**：客户对软件的高层目标要求。

2）**用户需求**：用户使用软件必须达到的要求和完成的任务。通常在用例(use case)或方案脚本(scenario)中加以说明。

3）**功能和非功能需求**。规定了开发人员必须实现的需求，它的实现将满足上述业务需求和用户需求。通常以需求规格说明(requirement specification)的形式给以详尽描述。

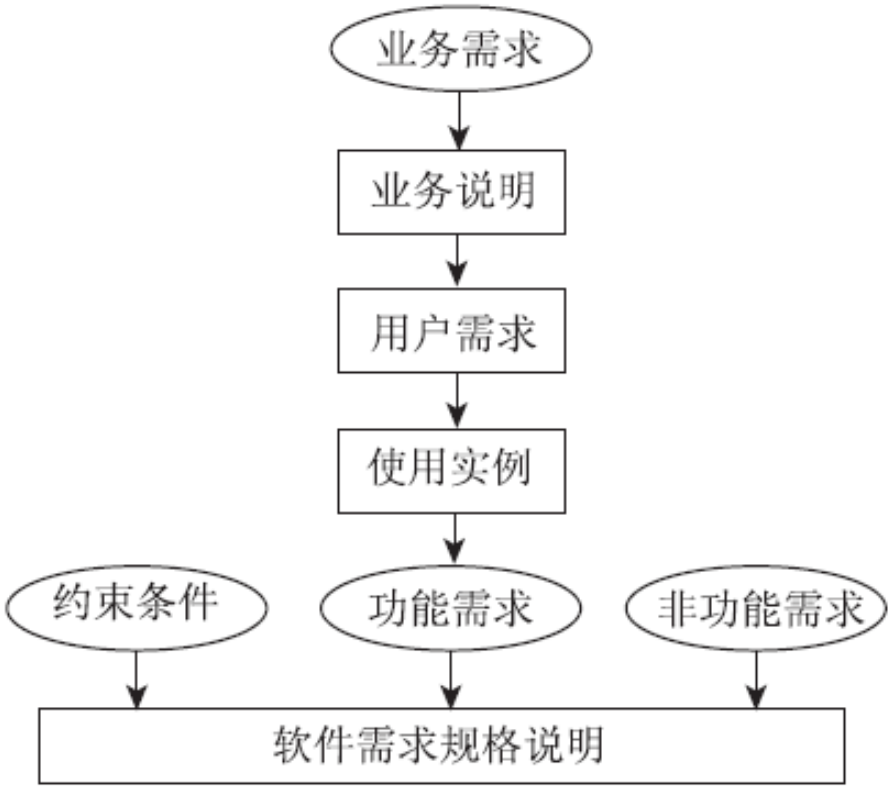
13.5.1 系统需求与软件需求

- **软件需求**

✓ 非功能需求包括以下两种：

1）**过程需求**：交付需求、实现需求、遵循的标准。

2）**外部需求**：互操作性、伦理性、机密性、安全性等。



软件需求的层次

13.5.1 系统需求与软件需求

- **软件需求**

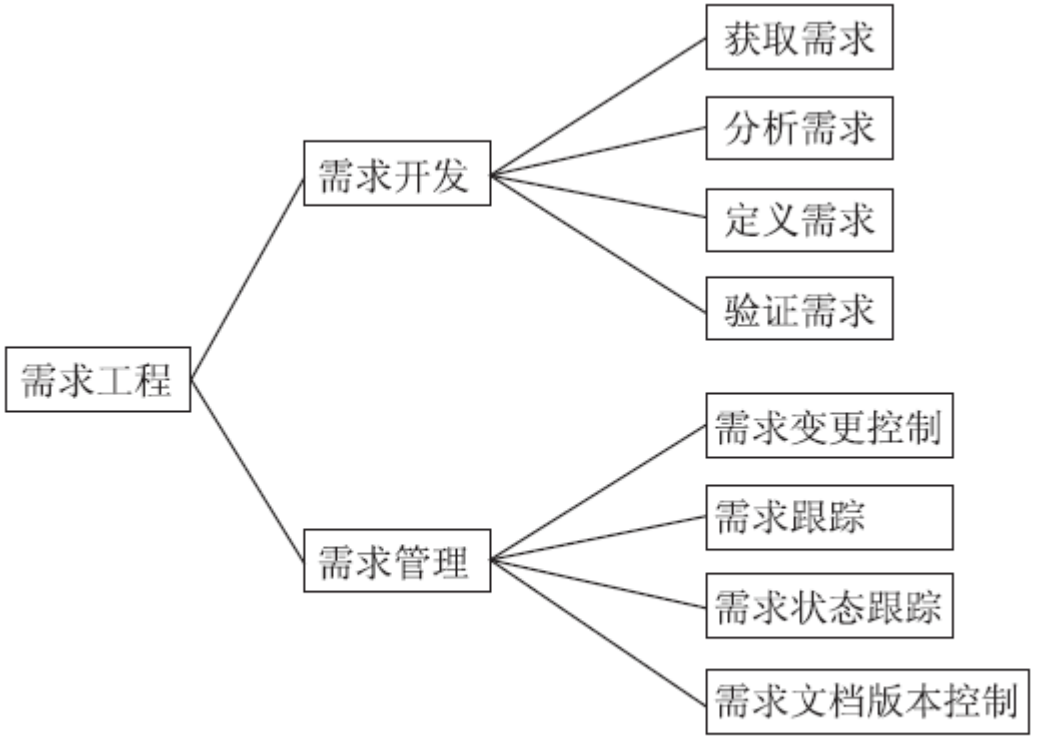
- ✓ 质量功能展开（QFD）：QFD是将客户的要求转化为软件技术需求的质量管理方法，于1970年在日本推出。

- ✓ QFD将客户的需求分为3类：。

1）常规需求：它是客户明确提出的需求，软件开发组织必须千方百计设法将其实现，使客户得到起码的满足。
2）期望需求：这是一些隐含而未被客户明确提出的需求。也可称此为客户的潜在需求。
3）兴奋需求：客户完全没有想到的需求，如果产品中未将其实现，客户并不抱怨；但若真的实现了，就会超出客户的想象，他们会感到十分惊讶和非常满意。

13.5.2 需求工程

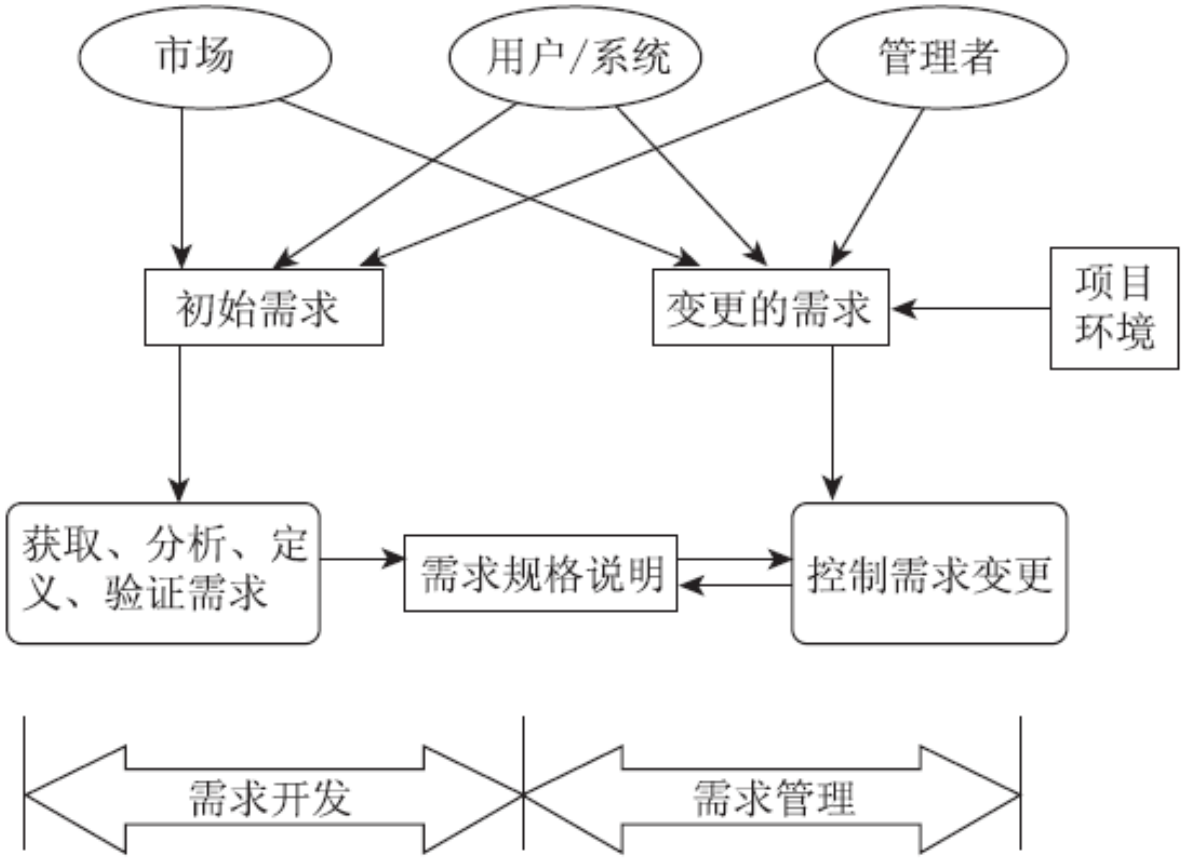
- **需求工程**是系统工程或软件工程中解决需求问题的一个崭新领域。其目标在于使得到的产品能够准确、真实地体现客户的需求，令客户满意。
- 需求工程包括两个方面：需求开发与需求管理。



需求工程的构成

13.5.2 需求工程

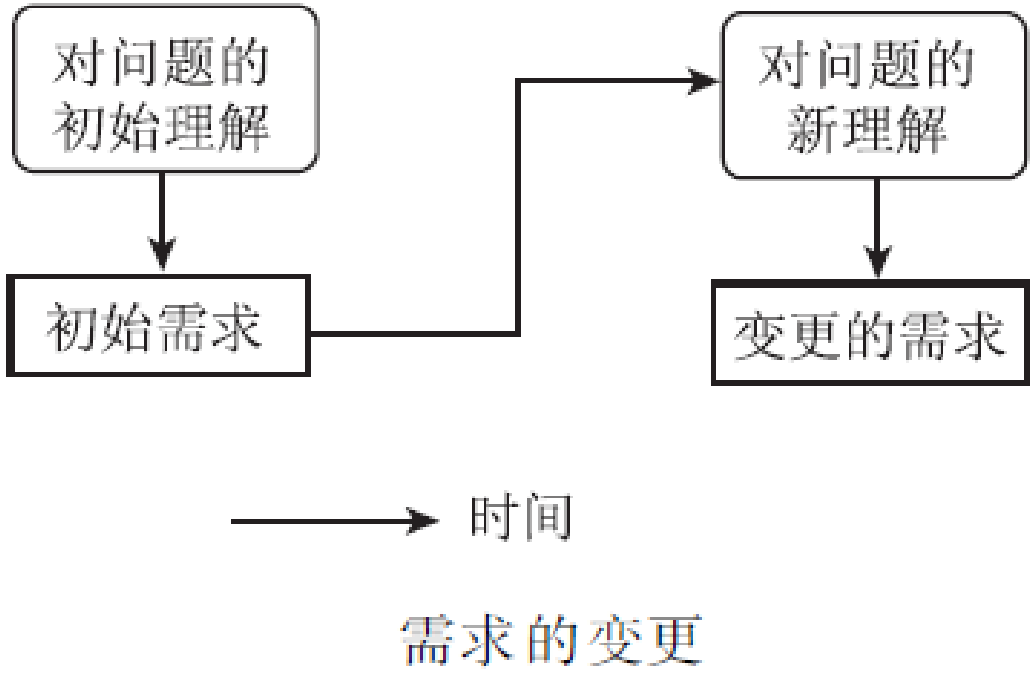
- 需求开发与需求管理的关系



需求开发与需求管理

13.5.3 需求变更

- **需求变更难于完全避免**
  - 系统需求或软件需求往往在开发工程中发生变更，提出变更有可能在开发的任何阶段。



需求的变更

13.5.3 需求变更

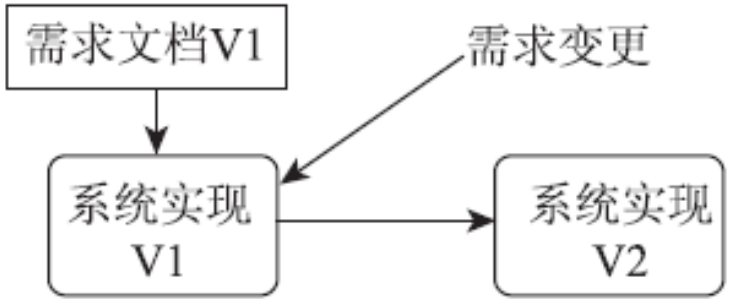
- **需求变更原因分析**
  - 单纯的用户因素。
  - 市场形势变化引发的需求变更。
  - 系统因素。在系统内部，如计算机硬件、系统软件或数据等的变更要求软件与其相适应。
  - 工作环境因素。与软件运行相关的工作制度或法规、政策的变更，或业务要求变更导致的需求变更。
  - 需求开发工作有缺陷，可能有两种情况：一是需求分析、定义和评审工作不够充分，致使需求规格说明中隐含着问题；二是需求开发中开发人员与用户沟通不够充分。

13.5.3 需求变更

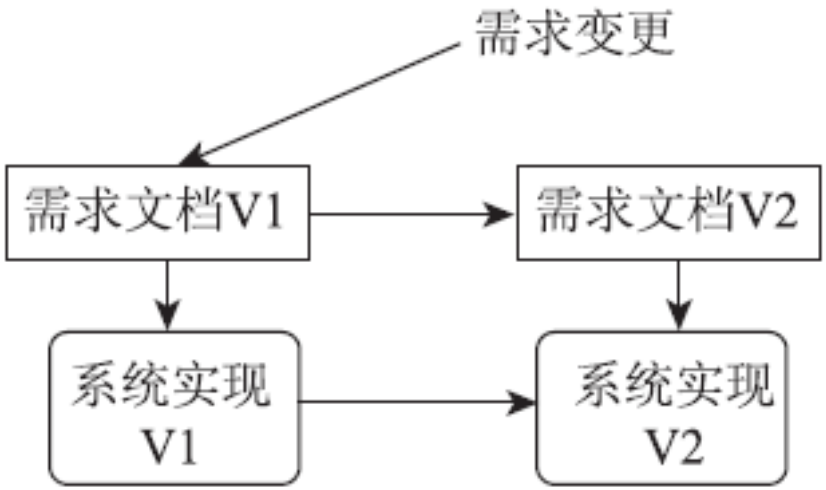
- **需求变更对软件开发工作的影响**
  - 使得变更前的开发工作和成果失效。
  - 使得返工成为不得不采取的对策。
  - 势必带来软件开发计划的相应变更、开发成本的相应增加和开发工作量及资源投入的追加。

13.5.3 需求变更

- **需求变更失控可能导致的后果**
  - 可能导致开发出的产品不符合变更了的需求，也就是得到的产品并不是用户所需要的产品这样的严重后果。



a) 未受控的需求变更引起需求和现实不一致



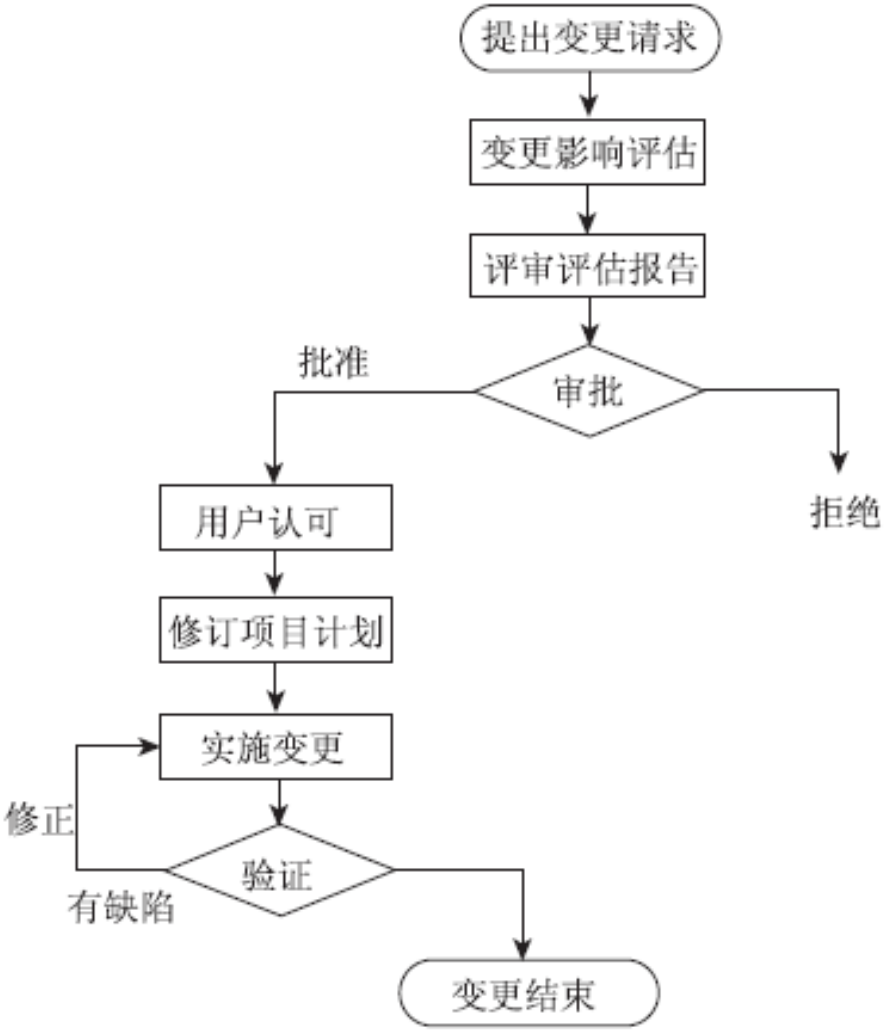
b) 受控的需求变更使需求和现实一致

13.5.3 需求变更

- **降低需求变更风险的策略**
  - 在需求开发工作中要与客户充分沟通。
  - 与用户共同确定需求。
  - 开发组织和用户双方签署的项目开发合同中应包括对出现需求变更的应对条款。
  - 如果项目自身具有需求不易确定的特点，在项目启动时最好采用快速原型方法或螺旋模型，以便在确认需求的基础上开发产品。
  - 在项目开始时，如估计到需求可能变更，则可在开发计划中适当留有余地，以防变更需求造成被动。
  - 严格实施变更控制，使产品质量不致因需求变更受到影响。

13.5.4 需求变更控制

- **变更控制的步骤**
  - 提出变更请求。
  - 审理变更请求，进行变更影响评估。
  - 批准变更请求。
  - 取得用户的认可。
  - 修订项目计划。
  - 实施变更。
  - 验证变更。



需求变更控制的流程

13.5.4 需求变更控制

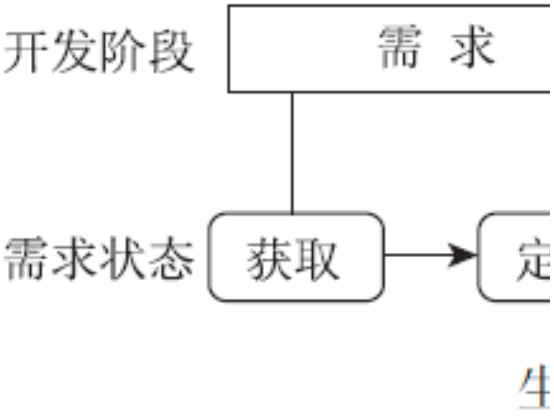
- 变更控制的实施
  - 需求变更请求。

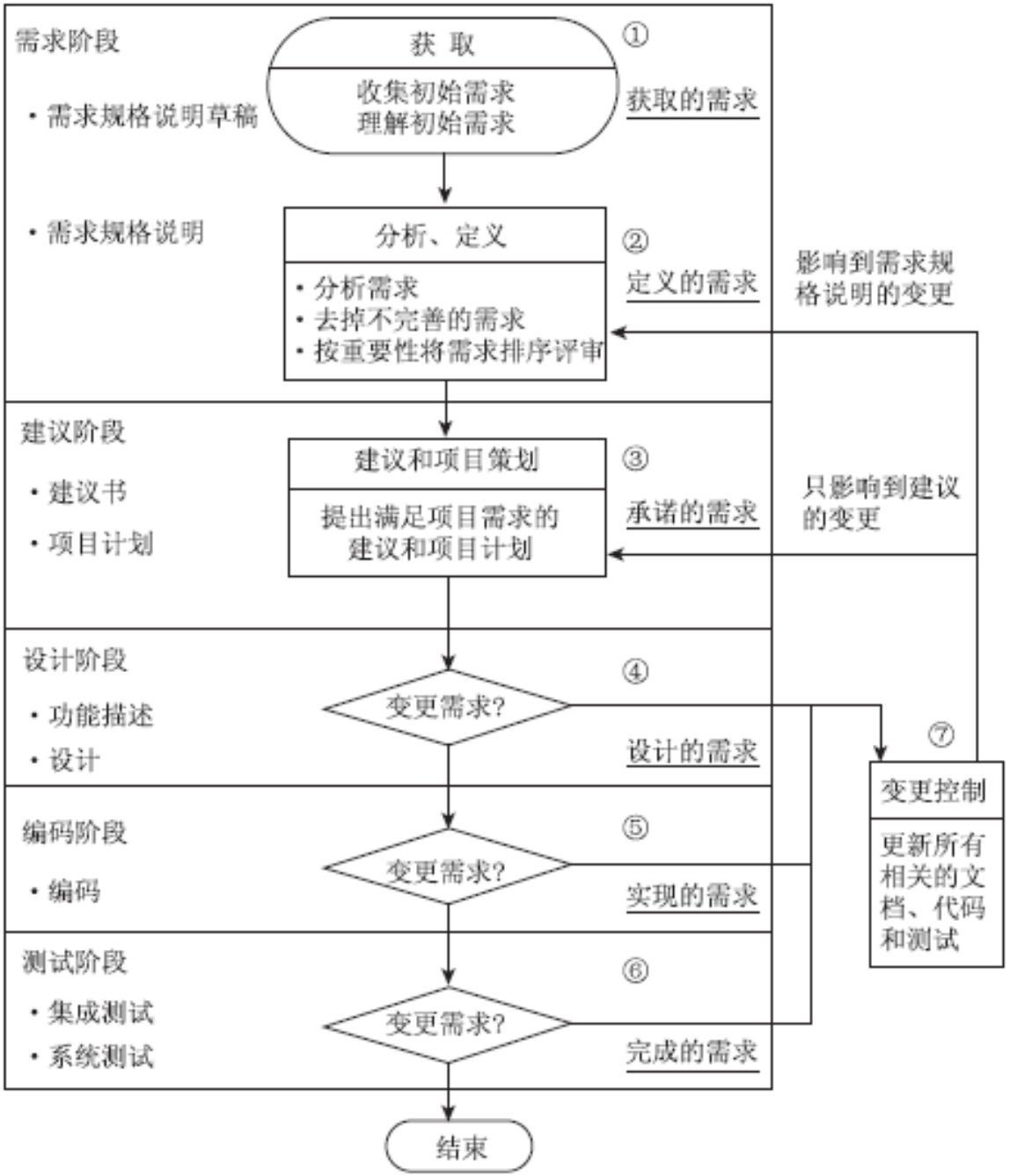
需求变更请求实例

项目名：XYZ			
变更申请号	11	日　　期	2007 年 2 月 23 日
变更说明	IS-41 分析器——IS-41 分析器对 CDMA 的影响		
影响分析	(1) 对 CDMA 的配置模块和分析器无影响 (2) TDMA 码可重用；SCRIPTS 也可复用 (3) 受影响的模块 ① CGAAPP 模块，需对 IS-41 单独进行规范性分析 ② CDMAPPO1 模块 • TRIS41ROI 按 TRCDMAIS41ROI 复制 • 使用纯虚拟对 TRCDMAROI 建立 Actual Call Mode Manager 并重新定义 ③ SILVER06GUIAPP++模块；在资源表中加入 IS-41		
工作量	5 人日		
计划时间	无需重大变更		
状态	将并入新的 CDMA 软件包		

13.5

- **变更控制的实施**
  - **需求控制流**。在
    行着状态的演变





13.5.5 可追溯性管理

- **需求可追溯性与需求变更控制**
  - 如果将笼统的需求状态演变概念加以具体化，考虑某一项特定的需求，它也必然随着开发工作的进展而逐步扩展和演化。
  - 如果以某种方式(例如以下给出的可追溯矩阵)对其做出确切的表达，那么需求变更无论出现在任何阶段，都能沿用着这一线索进行无遗漏的追踪，对相关部分实施修正和调整，最终做到变更控制。

13.5.5 可追溯性管理

- **可追溯性管理的目标**
  - 实施需求可追溯性管理应使每一项需求均能追溯到，包括对应的设计、实现该项需求的代码以及测试此项实现的用例。
  - 这样便可做到确保软件产品满足所有需求，并已测试了所有需求，从而使表现前后继承关系的脉络清晰可见。

13.5.5 可追溯性管理

- **两类不同的追溯**
  - 向前追溯：沿生存期从需求跟踪到设计、编码、测试等后继阶段所输出工作产品的相关元素。
  - 向后追溯：从各阶段工作产品的元素反向追溯，直至追溯到初始需求。

13.5.5 可追溯性管理

- 可追溯矩阵

追溯矩阵实例

①	②	③	④	⑤	⑥	⑦	⑧
需求号	需求描述	概要设计文档索引号	对应的设计（功能、结构、数据库）	实现（程序、类、继承类）	单元测试用例	集成/系统测试用例	验收测试用例
1.1.2	利用收集的数据实现亮点的实时集成	5.3.2	数据采集与亮度控制器接口	PB405 数据采集	#12	#46	#11
				CICS203 亮点控制器启动	#1	#47	#11

13.6 配置管理

- 软件工程项目随着工作的进展会产生多种信息，包括技术资料、管理资料等，如何管好这些资料是项目管理面临的重要问题。
- 另一方面，还必须考虑到，这些资料和信息不仅不断地产生，而且还在不断地演化和变更。
- 如何遵循一套严谨、科学的管理办法，使信息和资料的产生、存放、查找和使用既有序又高效，不致发生混乱和差错的现象，这正是**配置管理所要解决的问题**。

13.4 配置管理

- 配置管理概述
- 软件配置标识
- 变更管理
- 版本控制
- 系统建立
- 配置审核
- 配置状态报告

软件配置管理概述

- **软件配置管理的目的**是为某个过程或某个项目的软件项建立和保持完整性，以便相关方便于使用。
- 软件配置管理要**开展的活动**包括：**配置标识**、**配置控制**、**配置状态报告**、**配置评价**以及**发布管理**、**交付**等。

软件配置管理概述

我们将软件配置管理的对象称为软件配置项（software configuration item），包括：

（1）与合同、过程、计划和产品有关的文档及数据；

（2）源代码、目标代码和可执行代码；

（3）相关的产品，包括软件工具、库内的可复用软件、外购软件及顾客提供的软件。

软件配置管理概述

软件配置管理的主要任务如下：

（1）制订软件配置管理计划。包括：

① 配置标识规则；

② 如何建立配置数据库，并将配置项置于配置管理之下；

③ 配置管理人员的职责及配置管理活动；

④ 所采用的配置管理工具、技术和方法。

（2）实施变更管理，防止项目进行中因变更导致的混乱。

（3）实施版本管理和发布管理。

软件配置管理概述

软件配置管理的工作是要解决下列问题：

（1）采用什么方式去标识和管理数量众多的程序、文档等的各种版本？

（2）在软件产品交付用户之前和交付之后如何控制变更？实现有效的变更？

（3）谁有权批准变更以及安排变更的优先级？

（4）用什么方法估计变更可能引起的其他问题？

这些问题的解决正是软件配置管理应完成的任务：配置标识、版本管理、变更管理、配置审核及配置报告。

软件配置标识

软件配置是一个动态的概念。不仅随着开发工作的进展会出现许多需控制的文档，而且开发过程中会出现各种变更。为了达到特定的要求，必须对配置项进行控制，而实现控制的首先就要对它们命名，这正是**配置标识的任务**。

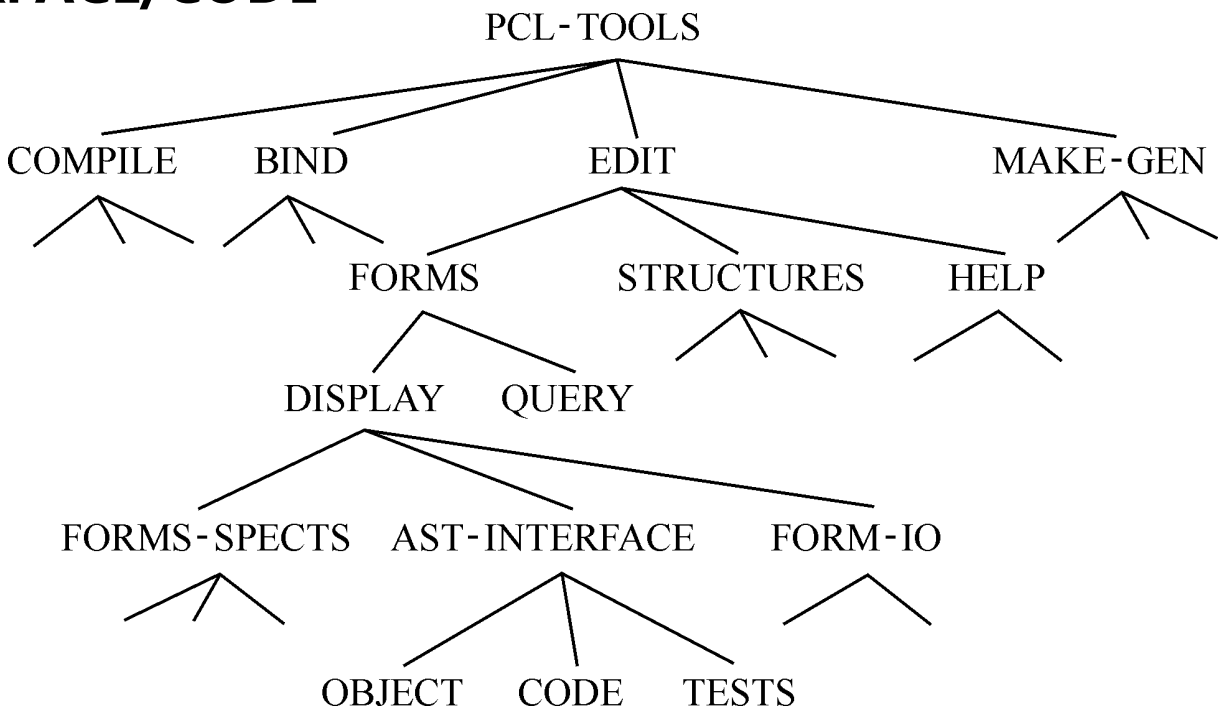
制订适当的命名规则是配置标识的第1步工作。命名不能任意、随机地进行，命名要求具有：

（1）**唯一性**：目的在于避免出现重名，造成混乱；

（2）**可追溯性**：使命名能够反映命名对象间的关系。

软件配置标识

- **例如，可以采用层式命名规则以利反映树状结构，某树状结构软件的结构图如图所示。CODE部分可沿树状结构命名为 PCL-TOOLS/EDIT/FORMS/DISPLAY/AST-INTERFACE/CODE**



软件配置标识

- 可以利用面向对象的方法进行标识。通常需标识两类型对象：**基本对象**和**复合对象**。

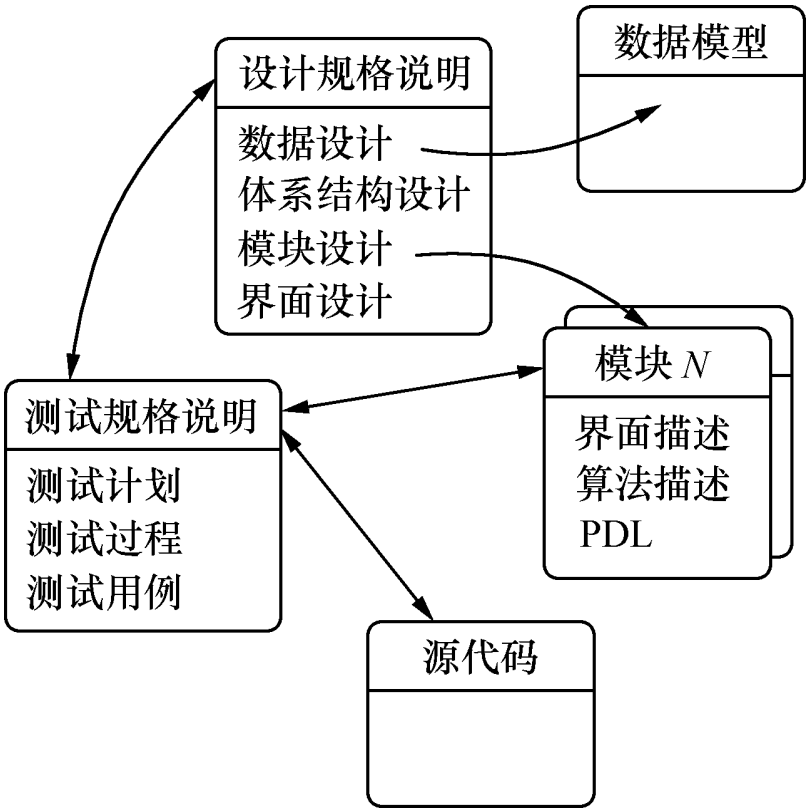
➢ **基本对象**是由软件工程师在分析、设计、编码和测试时所建立的“文本单元”。

例如，基本对象可能是需求规格说明中的一节，一个模块的源程序清单、一组用来测试一个等价类的测试用例。

➢**复合对象**则是基本对象和其他复合对象的集合。

软件配置标识

- **复合对象实例：“设计规格说明”是一个复合对象，它是一些基本对象（如“数据模型”和“模块N”）的集合。**



软件配置标识

每个对象可用一组信息来唯一地标识，这组信息包括：

（名字、描述、资源、实现）

- 对象的名字是一个字符串，它明确地标识对象。
- 对象描述是一个表项，它包括：对象所表示的软件配置项类型（如文档、程序、数据）、项目标识、变更或版本信息。
- 资源是“由对象所提供的、处理的、引用的或其他所需要的一些实体”。例如，数据类型、特定函数，甚至变量名都可以看做是对象资源。
- 对于一个基本对象来说，“实现”是指向“文本单元”的指针，而对于复合对象来说，则为null（空）。

软件配置标识

- 配置对象的标识还必须考虑在命名对象之间存在的联系。
  - <part of> 联系： 一个对象可以是一个复合对象的一个组成部分，使用联系<part of>进行标识。这个联系定义了对象的层次。例如，

    E—R diagram 1.4 <part of> data model;

    data model <part of> Design Specification;
  - <interrelated>联系：对象之间的联系可以跨越对象层次的分支相互关联。

    data model <interrelated> data flow model;

变更管理

1．变更不可避免

软件开发过程中变更是不可能避免的，变更控制就是要把变更严格地控制起来，随时保留变更的有关信息，把精确、清晰的信息传递到开发过程的下一活动或下一任务，防止出现混乱。**变更管理的任务**如下：

（1）分析变更，根据成本—效益和涉及的技术等因素判断变更实施的必要性，确定是否实施变更。

（2）记录变更信息，并追踪变更信息。

（3）确保变更在受控条件下进行。

为有效地实现变更控制需借助于配置数据库和基线的概念。

变更管理

2．配置数据库

设置配置数据库，使它发挥出以下作用：

（1）用其收集与配置有关的所有信息；

（2）评价系统变更的效果；

（3）提供配置管理过程的管理信息。

变更管理

配置数据库可分为开发库、受控库和产品库3类。

① 开发库：专供开发人员使用，其中的信息可以进行频繁的修改，对其控制相当宽松。

② 受控库：其中存放在生存期某一阶段工作结束时释放的阶段产品，这些是与软件开发工作相关的计算机可读信息和人工可读信息。软件配置管理正是对受控库中的各个软件项进行管理，受控库也称为软件配置管理库。

③ 产品库：在开发的软件产品完成系统测试后，作为最终产品存入产品库中，等待交付用户或现场安装。

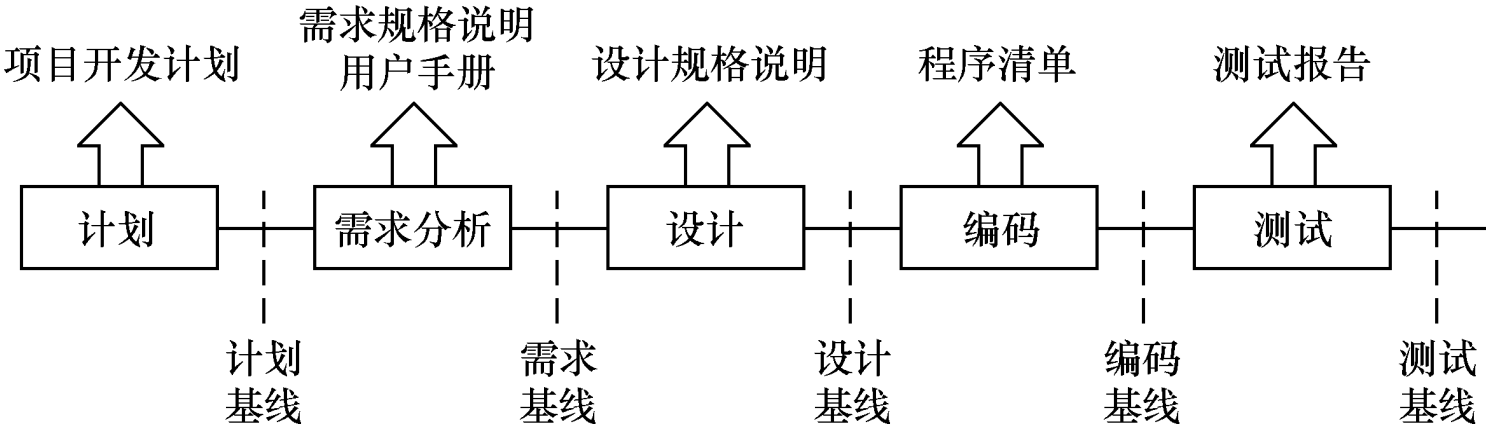
变更管理

3．基线和变更控制

- **基线**（baseline）是软件生存期各开发阶段末尾的特定点，也被称为**里程碑**（milestone）。
- 它的作用是把各阶段的开发工作划分得更加明确，使得本来连续的工作在这些点上断开，使之便于检验和确认阶段开发成果。
- 它对变更控制起的作用是，不允许跨越里程碑去修改另一阶段的工作成果。

变更管理

下图所示为软件开发过程的若干配置基线。以设计基线为例，若项目的进展已跨过了设计基线，开始了编码工作，那么设计的变更必须受到严格的控制，原则上已不允许，应该认为，此时的设计已被“冻结”。



变更管理

4．变更管理过程

变更管理过程可用下图给出的流程来说明。

```
提交变更请求表（CRF）
分析变更请求
如果 变更请求可接受 则
    估计变更如何实现
    估算变更成本
    将 CRF 送交变更控制委员会（CCB）
    如果 变更获准 则
        重复
            实施变更
            记录变更
            将变更的软件提交质量保证人员审查
        直到 软件质量达到要求
          由配置管理人员生成系统的新版本
    否则 拒绝变更请求
否则 拒绝变更请求
```

变更管理

- **变更请求表（change request form，CRF）的格式如下表所示。表中一些内容需由变更分析人员对变更进行分析和评估以后填写。**

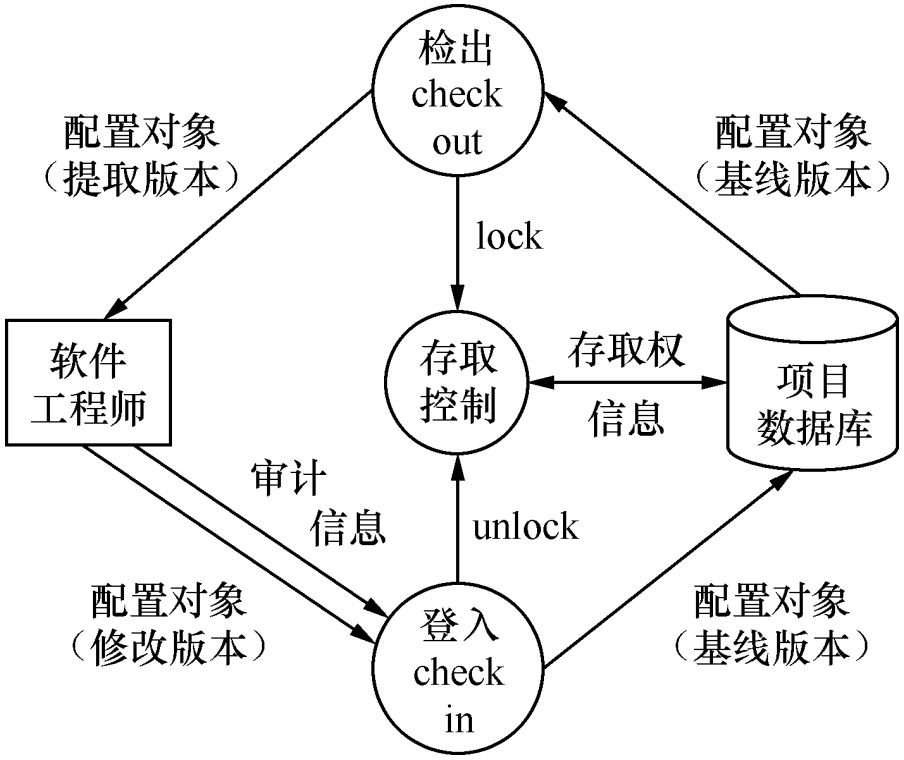
项目名_____ 变更请求人_____ 日期_____编号_____
要求的变更描述_____
变更分析员 _____ 分析日期_____
变更影响模块 _____
变更评估 _____
变更优先级 _____
变更实现 _____
评估工作量 _____
CCB 收到申请日期_____ CCB 决定日期_____
CCB 决定 _____
变更实施责任人 _____ 变更日期_____
递交 QA 日期 _____ QA 决定 _____
递交 CM 日期 _____

变更管理

- “检出”和“登入”处理实现了两个重要的变更控制要素，即**存取控制**和**同步控制**。
- 存取控制管理各个工程师存取或修改一个特定软件配置对象的权限；
- 同步控制可用来确保由不同的人所执行的并发变更不会产生混乱。

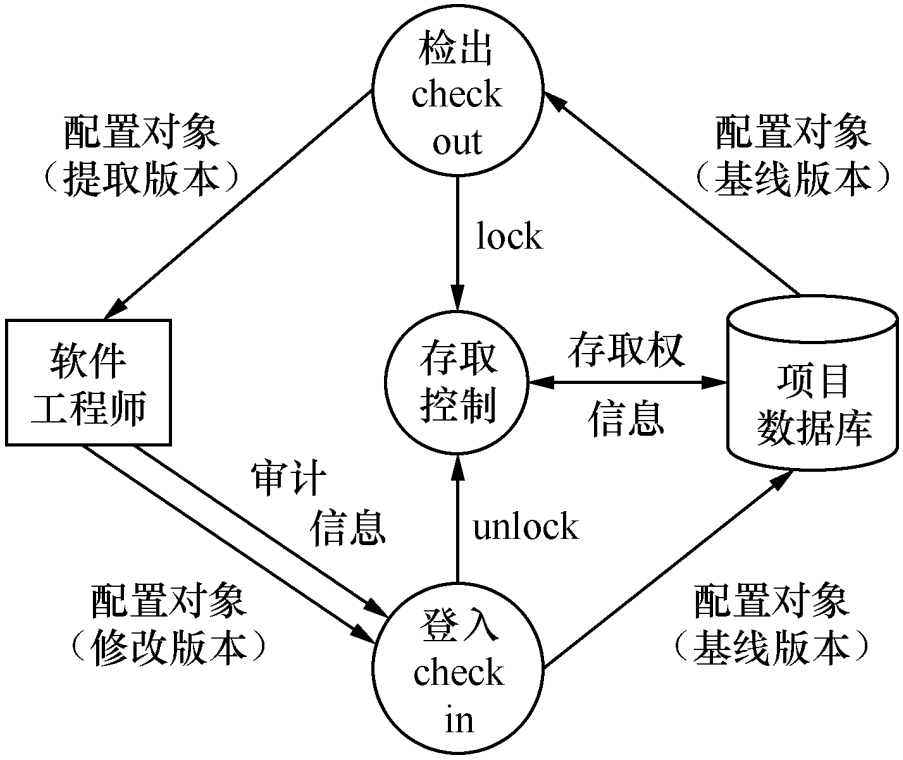
变更管理

- **存取和同步控制如图所示。根据经批准的变更请求和变更实施方案，软件工程师从项目数据库中检出要变更的配置对象。**



变更管理

- 存取控制功能保证了软件工程师有检出该对象的权限。
- 同步控制功能则封锁（lock）了项目数据库中的这个对象，使得当前检出的版本在没有被置换前不能再更新它。



变更管理

- **软件的变更通常有两类不同的情况：**
  - **为改正小错误需要的变更；**
  - **为了增加或者删掉某些功能，或者为了改变完成某个功能的方法而需要的变更。**

版本控制

1．版本管理和发行管理

（1）版本管理

- **版本管理**（version management）是对系统不同版本进行标识和跟踪的过程。
- 版本标识的目的是便于对版本加以区分、检索和跟踪，以表明各个版本之间的关系。
- 一个版本是软件系统的一个实例，在功能上和性能上与其他版本有所不同，或是修正、补充了前一版本的某些不足。
- 这些不同的版本可能在功能上是等价的，但它们分别适应于不同的硬件或软件环境的要求。

版本控制

（2）系统发行

系统发行（system release）是分配给客户一个版本，每次系统发行都应有新的功能或是针对不同的系统运行环境。通常软件系统的版本数要比发行次数多，因为有的版本并未发行。例如，有的版本仅供开发机构内部使用，或是专供测试等。

通常一次发行不仅只是提供一个可执行程序，或一套程序，可能还要包括：

- 配置文件：规定发行所作的特定安装；
- 数据文件：系统运行所需的数据；
- 安装程序：表明系统如何安装到目标机上；
- 电子文档或书面文档：这是对系统的描述。

版本控制

2．版本标识

版本标识（version identification）是由版本的命名规则决定的。由于前后版本存在着传递关系，因此，如何正确地反映这一传递关系，就应当体现在其命名中。

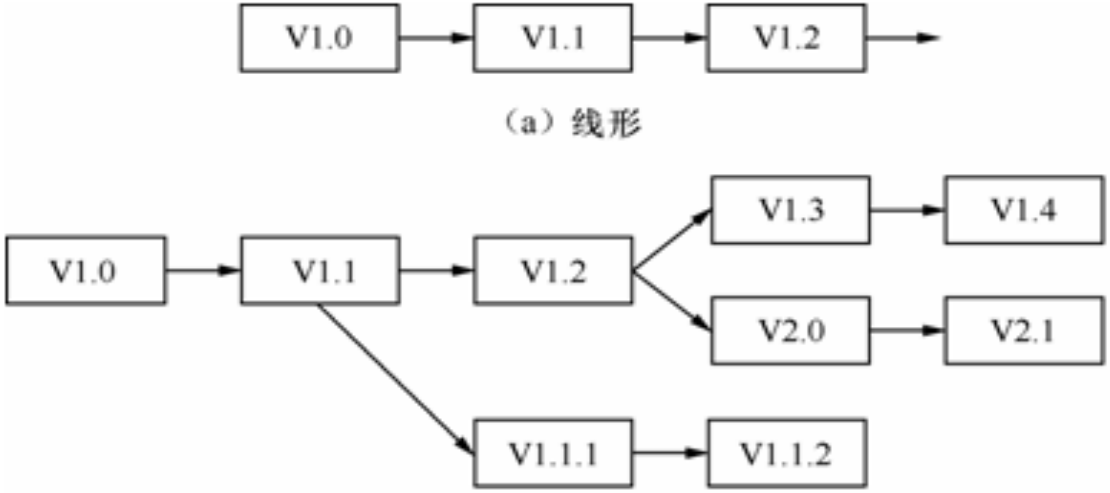
可能使用的命名规则有下面所述的几种：

- **号码顺序型版本标识**
- **符号命名版本标识**
- **属性版本标识**

版本控制

（1）号码顺序型版本标识

- **如下图所示。这种标识十分明显地给出了版本之间的传递关系，但是如果当前版本生出了多个新版本，标识就稍有困难。**



版本控制

（2）符号命名版本标识

用符号表达版本间的传递关系，如不用V1.1.2的形式，而采用V1/VMS/DB Server来表示一个在VMS操作系统上运行的数据库服务器版本。

（3）属性版本标识

属性版本标识是把有关版本的重要属性反映在标识中，可以包括的属性有：客户名、开发语言、开发状态、硬件平台、生成日期等。每个版本都由唯一的一组属性标识，即一组具有唯一性的属性值。

版本控制

3．发行管理

一个系统的新发行与新版本有着不同的含义。

- **新版本**是在修改发现的软件缺陷后，开发出新的程序、形成新的系统；
- **新发行**是除了写出新的程序，形成新系统之外，还要为用户准备数据、配置文件、编写新文档，准备新包装。
- 新发行要比新版本开销大。

版本控制

- **系统发行策略**

➢**无论是哪一种维护工作完成之后，配置管理人员都要分析变更影响的组件，确定何时生成新系统、何时进行系统发行。**

➢**通常一个系统改动越多，引入错误的机会也越多，发现的错误必须在下次发行时解决。为了把出现问题的机会分散开，往往把修补变更后的发行与系统功能变更的发行交叉起来，如将其按下图所示的顺序来安排。**

系统建立

- **系统建立**（system building）是将系统的组件组合成完整的程序，以执行某一特定目标配置的过程。
- 该过程中可能包括一些组件的编译以及将目标代码结合在一起，构成可执行系统的连接过程。

系统建立

- **系统建立必须要考虑的因素有：**

(1) 构成系统的所有组件是否都已包含在系统建立的指令中？

(2) 每个需要组件的适当版本是否包含在系统建立的指令中？

(3) 所有需要的数据文件都是可用的吗？

(4) 如果在一个组件内引用了数据文件，所有数据名与目标机上数据文件的名字是否一致？

(5) 编译程序和其他所需工具的适用版本是可用的吗？软件工具目前流行的版本是否与开发系统时所用的版本兼容？

配置审核

- 软件的完整性，是指开发后期的软件产品能够正确地反映用户所提出的对软件的要求。
- 软件配置审核（configuration audit）的目的就是要证实整个软件生存期中各项产品在技术上和管理上的完整性。同时，还要确保所有文档的内容变动不超出当初确定的软件要求范围。使得软件配置具有良好的可跟踪性。
- 这是软件变更控制人员掌握配置情况、进行审批的依据。

配置审核

- 软件的变更控制机制通常只能跟踪到工程变更顺序产生为止，那么如何知道变更是否正确完成了呢？一般可以用**正式技术评审**和**软件配置审核**两种方法审查。

  - 正式的技术评审着重检查已完成修改的软件配置对象的技术正确性，评审者评价软件配置项，决定它与其他软件配置项的一致性，是否有遗漏或可能引起的副作用。原则上，正式技术评审应对所有的变更进行。
  - 软件配置审核作为正式技术评审的补充，评价在评审期间没有考虑的软件配置项特性。

配置状态报告

为了清楚、及时地记载软件配置的变化，不致于到后期造成贻误，需要对开发的过程做出系统的记录，以反映开发活动的历史情况。这就是**配置状态报告的任务**。

报告的主要根据是变更控制小组会议的记录，报告对于每一项变更说明：

（1）发生了什么？

（2）为什么会发生？

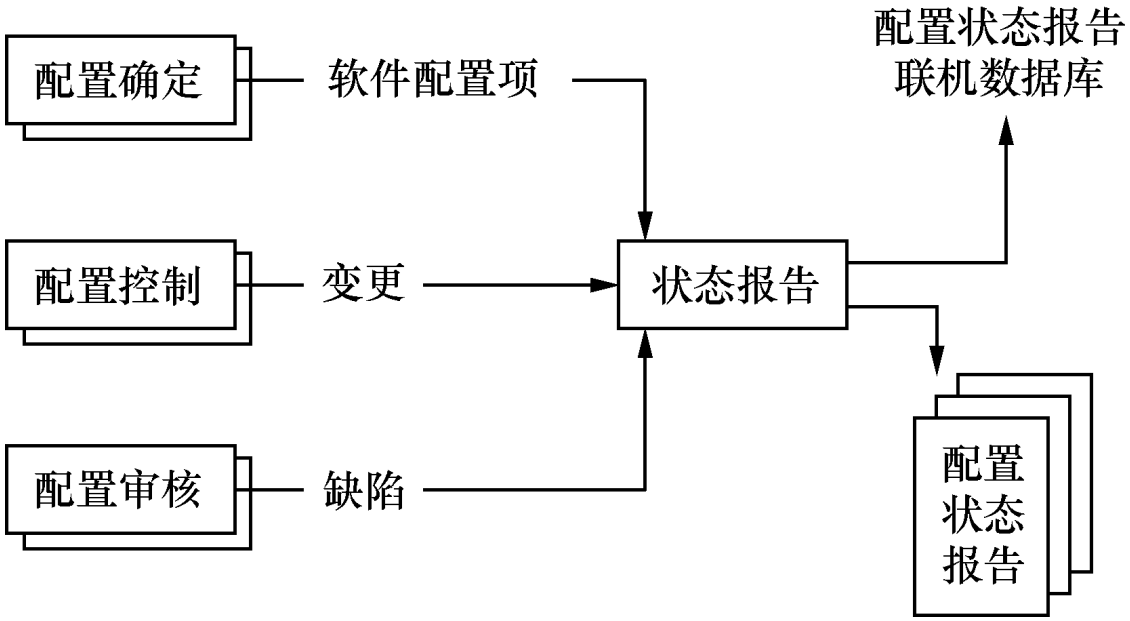
（3）谁做的？

（4）什么时候发生的？

（5）会有什么影响？

配置状态报告

下图描述了配置状态报告。每次新分配一个软件配置项或更新一个已有软件配置项的标识，或者一项变更申请被变更控制负责人批准，并给出一种工程变更顺序时，在配置状态报告中就要增加一条变更记录条目。一旦进行了配置审核，其结果也应该写入报告之中。



That's All!