东北大学

硕士学位论文

软件成本估计模型CoCoMoⅡ的研究与实践

姓名：孙熙玮

申请学位级别：硕士

专业：计算机应用

指导教师：程秋木

2002.1.1

软件成本估计模型 COCOMO II 的研究与实践

摘要

软件成本估计模型 COCOMO II 是提高软件能力成熟度（CMM）等级的一项重要的技术，它利用软件的规模来估计软件的工作量和进度，从而为软件的开发、度量和决策提供有用的信息。

本文首先介绍了一些关于软件成本估计的知识，以及与其相关模型和技术，而后着重讨论了 COCOMO II 模型及其扩展模型。本文还指出了在实际中如何使用 COCOMO II 模型来估计软件成本，如何利用实际的数据来校正模型参数。在对 COCOMO II 模型深入研究的基础上，本文分析和设计了一个用于软件成本估计的 COCOMO II 系统。该系统可使用 COCOMO II 模型技术来自动估计软件项目的成本，并利用实际项目数据对 COCOMO II 模型的参数进行校正。

关键字：COCOMO II　　软件成本估计　　CMM

Research and Practice of Software Cost Estimation Model COCOMO II

ABSTRACT

Software cost estimation model COCOMO II is an important technology to improve the software capability maturity (CMM) level, it uses the size of the software to estimate the effort and schedule of the development of the software, and provides useful information for the development, measurement and decision of the software.

Above all, this paper introduces some knowledge about software cost estimation and its some models and technologies, then emphatically discusses COCOMO II model and its extension models. This paper also points out how to use COCOMO II model to estimate software costs, and how to use practical data to calibrate the parameters of the model in reality. Based on the deep research of COCOMO II model, the paper analyses and designs a COCOMO II application system to estimate software costs. This system can use COCOMO II model technology to estimate the costs of software project automatically, and can make use of the practical project data to calibrate the parameters of the COCOMO II model.

Keywords: COCOMO II, Software cost estimation, CMM

第一章 绪论

在当今的计算机系统开发中，随着计算机硬件平台开发成本的大规模降低，软件系统开发成本却变得越来越高，因此我们越来越意识到了在软件和系统生命周期成本、周期时间、功能、性能和质量之间进行权衡能力的重要性和必要性。

软件成本估计就是成本管理和软件管理的核心任务之一，而且软件成本估计也是提高软件能力成熟度（CMM）等级的一项重要的技术。同时，软件成本估计是一个复杂的活动，它需要大量的关于被估计项目的关键属性的知识。成本估计有时被称为参数估计，因为估计的准确性需要对离散的参数之间关系的准确的理解。这些参数能单独地或整体地影响软件工程的产品，而创建准确的软件成本估计通常需要以下参数的知识。

- 主要提交产品的规模，例如规格、源代码以及手册。
- 在开发过程中需求可能改变的比率。
- 可能遇到的错误或缺陷的大概数目。
- 开发小组的能力。
- 与开发小组有关的工资和超额成本。
- 将会使用的正式的开发方法。
- 项目中将使用的工具。
- 将执行的开发活动序列。

1.1 软件成本估计的目的

软件成本估计可用于以下几个目的。它包括：

- 预算：基本的但不是唯一重要的用途。全局估计的准确是最被期望的能力。
- 权衡和风险分析：一个重要的附加能力，用于说明软件工程决定（规模、人员、工具、可复用性等等）的成本和进度的明感性。
- 工程策划和控制：一个重要的附加能力，用于通过组件、阶段和活动来提供成本和进度的拆分。

- 软件改进投资分析：一个重要的附加能力，用于估计成本，同时也估计象工具，可复用性和过程成熟度这些策划的益处。

1.2 软件成本估计的六种形式

手工软件估计方法：

- 使用简单规则的手工工程级估计。
- 使用比率和百分比的手工阶段级估计。
- 使用工作拆分结构的手工活动级估计。

自动软件估计方法：

- 自动工程级估计（宏观）。
- 自动阶段级估计（宏观）。
- 自动活动级或任务级估计（微观）。

1.2.1 手工软件估计方法简介

手工软件工程估计方法是使用简单的规则来形成简单的软件成本估计，虽然它不那么准确，但因为简便，现在这种方法仍在广泛使用。

手工阶段级估计通常包括对 5 到 8 个阶段的估计，这些阶段一般为

- 需求收集。
- 分析和设计。
- 编码。
- 测试。
- 安装及培训。

但它的问题是：

- 实际上每个活动的百分比差别很大。
- 许多种的软件工具跨越多个阶段或运行在整个工程周期中。
- 不是阶段的活动可能偶然被从估计中删除。它虽然比整个工程估计更加有用，而且容易准备，然而它还是不能满足合同、预算或关键的商业目的的准确性要求。

另外，虽然使用工作拆分结构的手工活动级或任务级估计是手工方法中最准确的，但它是刚开始时费时，而且不易做改动。

1.2.2 自动软件估计方法简介

在自动软件估计方法中，前两种方法是宏观的估计，它用于对于完整工程级的估计，以及对于阶段级的估计（它通常使用内嵌的基于每个阶段的比率或百分比的假设来估计）。

宏观方法可做以下调整以适应不同情况：

- 对于不同的人员经验等级。
- 对于不同的软件开发过程。
- 对于使用的特别的编程语言。
- 对于软件应用的尺寸大小。
- 对于不同的工作习惯和超时。

而基于细节的工作拆分结构的自动化估计工具是微观工具。

宏观工具是应用于整个工程的广泛公式，它可以使用各个阶段所占的比率或百分比将资源和时间分配到特定的阶段。而微观工具则恰恰相反，它们先创建一个细节的工作拆分结构，然后评价每个活动，当所有的活动级或任务级估计结束后，估计工具将总结这些部分结果来形成完整的对于人员、能力、进度以及成本需求的估计。

1.3 COCOMO II 的简要历史及其早期模型简介

1.3.1 COCOMO II 的简要历史

所谓 COCOMO 模型就是 Boehm 提出的构造性成本模型。而 COCOMO II 模型实际上是当今世界上最通用的新型的软件成本估计模型，COCOMO II 的简要历史如下：

- 1981 年，Barry Boehm 在他的《软件工程经济》一书中介绍了早期的 COCOMO，这个模型现在被称为 COCOMO.81。
- 1987 年，ADA COCOMO 和增量式 COCOMO 出现。
- 1988、1989 年，对 ADA COCOMO 进行改进。
- 1995、1996 年，描述 COCOMO II 的早期论文出版。
- 1997 年， Boehm 开发出第一个 COCOMO II 的校正版，被命名为

COCOMO II.1997。

- 1998 年，发行了 COCOMO II 的第二个校正版，被命名为 COCOMO II.1998。
- 1999 年，COCOMO II.1998 改名为 COCOMO II.1999，后来又改名为 COCOMO II.2000（所有这三个模型是相同的）。

1.3.2 早期 COCOMO 模型的简介

在早期的 COCOMO 模型中，考虑开发环境，软件开发项目的类型，模型可以分为三种：

- 组织型(organic)：相对较小、较简单的软件项目。程序的规模不是很大（<50000 行）。
- 嵌入型(embedded)：要求在紧密联系的硬件、软件和操作的限制条件下运行，通常与某种复杂的硬件设备紧密结合在一起。对接口，数据结构，算法的要求高。软件规模任意。
- 半独立型（semidetached）：介于上述两种软件之间。规模和复杂度都属于中等或较高。最大可达 30 万行。

在这种模型中软件开发工作量表示成关于估计的应该开发的代码行数的非线性函数：

$$MM = C_1 * KLOC^a * \prod_{i=1}^{15} f_i \qquad (1.1)$$

其中，MM 是开发工作量（以人月为单位）；C_1 是模型乘法因数，根据项目类型不同，取相应的数值；KLOC 是估计的代码行数（以千行为单位）；a 是模型指数，根据项目类型不同，取相应的数值；f_i 是成本因素。

而且，每个成本因素都根据它的重要程度和影响大小而被赋予一定的数值。成本因素影响所有工程项目，即使不使用 COCOMO 模型估算成本，也应该注意这些因素，可以把成本因素划分成生产因素、计算机因素、人员因素和项目因素等等，下面简单列出主要的成本因素：

生产因素：要求的软件可靠性（RELY）、数据库规模（DATA）、软件产品复杂程度（CPLX）；

计算机因素：执行时间的约束（TIME）、存储约束（STOR）、环境变更率（VIRT）、计算机换向时间（TURN）；

人员因素：系统分析员的能力（ACAP）、应用经验（AEXP）、程序员的能力（PCAP）、环境知识（VEXP）、语言知识（LEXP）；

项目因素：程序设计实践（MODP）、软件工具（TOOL）、进度约束（SCED）；

其他因素：语言、实时应用、软件类型、经验、文档数量、用户需求和开发环境的稳定程度、管理等等。

另外，COCOMO 模型是层次型模型。COCOMO 模型按其详细程度可以分为三级：基本 COCOMO 模型、中间 COCOMO 模型、详细 COCOMO 模型。其中基本 COCOMO 模型是是一个静态单变量模型，它用一个以已估算出来的源代码行数(LOC)为自变量的经验函数来计算软件开发的工作量。 中级 COCOMO 模型在基本 COCOMO 模型的基础上，再用涉及产品、硬件、人员、项目等方面的影响因素来调整对工作量的估算。详细 COCOMO 模型包括中间 COCOMO 模型的所有特性，但更进一步考虑了软件工程中每一步骤（如分析、设计）的影响。最上层是对各种规模的软件的宏观估计模型，最下层是微观估计模型。

1.4 研究内容和论文结构

在实习期间，我主要做了以下的工作：CMM的理解和研究，软件配置管理工具VSS的功能扩展的分析和实现，软件的度量方法和手段的研究，软件成本估计理论的研究，软件成本估计的相关模型和技术方法的研究以及COCOMO II的研究与实践。

本文以东软集团东软股份公司的过程改善中心（原软件工程过程组 SEPG）为 CMM 认证而开发的项目过程管理系统中的软件成本估计系统为背景，研究和探讨了软件成本估计模型 COCOMO II 的分析设计与实践改进。

本文的组织结构如下：第二章介绍了软件成本估计的相关模型和技术；第三章先介绍了 COCOMO II 模型的体系结构，其中讨论了 COCOMO II 模型对实际软件开发的估计方法；第四章给出了一些对于 COCOMO II 模型的扩展的简介；第五章论述了本 COCOMO II 系统的分析设计和校正改善，并研究了 COCOMO II 系统的实际执行规程；最后总结全文，并指出今后应开展的工作展望。

第二章 软件成本估计的相关模型和技术

本章总结了软件成本估计的几类模型和技术：参数模型、基于专家技术、面向学习技术、基于动态模型、基于回归模型和用于集成基于专家和基于回归模型的组合 BAYESIAN 判决规则技术。到现在为止，经验显示神经网络和基于动态的技术没有上面提及的其它几类技术成熟。同时，软件技术领域的快速的变革正在挑战所有种类的技术。初步的结论是没有哪个单个技术都适用于所有的情况，而且对于几个不同方法结果的仔细比较是最可能产生实用的估计结果。

2.1 基于模型的技术

在过去的几十年里我们已经开发了一些软件评估模型，它们中的许多模型是属性型的模型。实验的理论决定了这些模型的功能形式。

2.1.1 PUTMAN 软件生命周期模型

在 20 世纪 70 年代后期，数量化软件度量组织的 Larry Putman 开发了软件生命周期模型（SLIM）。它是基于所谓的 Rayleigh 的项目人员等级与时间的分布的 Putman 的生命周期分析。它支持大多数受欢迎的规模度量方法，包括源代码行、功能点等等。它利用了一种称为 Rayleigh 曲线来评估项目的能力、进度和缺陷比率。

在 SLIM 中，生产率用于将基本的 Rayleigh 人力分布模型连接到规模和技术的软件开发特征。生产率 P 是软件产品规模 S 和开发工作量 E 的比值， 得到如下的公式。

$$P = \frac{S}{E} \tag{2.1}$$

用于定义工作量分布的 Rayleigh 曲线可用微分方程模拟如下：

$$\frac{dy}{dt} = 2Kate^{-at^2} \tag{2.2}$$

这里可以假设 K=1.0，a=0.02，t=0.18。

Putman 假设 Rayleigh 曲线的尖峰人员水平对应的时间即为开发时间（t_d）。Putman 发现 E 大约是 K 的 40%，这里 K 是指整个生命周期工作量。

2.1.2 BAILEY-BASILI 模型

John Bailey 和 Vic Basili 试图为开发一个局域的资源估计模型来建立一个模型生成过程。他们把在工作量与规模之间的背景公式形式化表示如下：

$$\text{工作量（人月）} = 0.73 * (\text{以 } SLOC \text{ 提交的尺寸})^{1.16} + 3.5$$

这个公式能预言用于完成一个普通项目的工作量。这个过程的下一步是制定出一系列的因数，它们用于区别不同的项目，并且帮助解释实际工作量与背景公式估计的工作量之间的差异。Bailey 和 Basili 挑选了最有影响力的环境属性作为解释预言工作量差异的因素。这个子集的属性应由专家、因素分析和相关标准的使用来确定，他们还将这些属性分成组，从而便于解释它们在工作量上正面或负面的作用。最终，他们确定了仅使用 21 个初始属性的 3 个组。具体如下：

整个方法（METH）：

- 树图。
- 从上到下的设计。
- 设计公式。
- 代码阅读。
- 主程序员组。
- 正式测试计划。
- 单元开发文件夹。
- 正式培训。

累积复杂度（CMPLX）：

- 用户接口复杂度。
- 用户初始的设计更改。
- 应用过程复杂度。
- 程序流复杂度。
- 内部交流复杂度。
- 外部交流复杂度。
- 数据库复杂度。

累积经验（CEXP）：

- 程序员资格。
- 程序员对机器的经验。
- 程序员对语言的经验。
- 程序员对应用的经验。
- 以前工作在一起的组。

估计公式如下：

$$\begin{aligned} Effort &= (Size)^A * METH \\ Effort &= (Size)^A * METH * CMPLX \\ Effort &= (Size)^A * METH * CMPLX * CEXP \end{aligned} \tag{2.3}$$

2.1.3 COCOMO II 模型

本文将在以后的章节详细论述这个模型及其实际估计方法。

2.1.4 GULEZIAN 模型

Ronald Gulezian 开发了一种利用 COCOMO 输入来进行成本估计的通用的软件开发成本估计模型。他重新形式化了初始的中间 COCOMO.81 模型，模型公式如下：

$$Effort_{estimated} = a(Size)^b \left(c^{\mathrm{mod}\,e}\right) \prod_{i=1}^{15} d_i^{EffortMultiplier_i} \tag{2.4}$$

其中，a 和 b 是指额定估计公式的系数，c 是指对应不同模型的系数估计，d_i 是指对应于第 i 个工作量乘法因数的系数估计。

这个公式可转化成一个线形公式，从而它的系数可用线性回归技术来校正。

2.2 基于专家的技术

这个软件估计技术是使用以前的专家积累的知识来开发的。基于他们的经验和对项目的理解，专家得到了成本和进度的估计。

2.2.1 DELPHI 技术

DELPHI 技术始建于 1848 年的 RAND 公司，这是个有效的方法来取得组的一致性。它减少了个人的偏见所带来的问题。

WIDEBAND DELPHI 方法：

1) 协调者提供 DELPHI 表格给每个检查的参与者。
2) 协调者举行一个会来讨论相关的问题。
3) 参与者完成 DELPHI 表格，并将它返还给协调者。
4) 协调者反馈参与者反映的情况。
5) 协调者举行另一个会议来讨论参与者反映的差异，达到一个可能的一致的结果。
6) 协调者请求参与者再次估计，再次取得一致，重复 4 至 6 步，直到达到满意的结果。

2.2.2 基于规则的系统

这个技术采用了人工智能技术，它利用事实来产生规则，规则反过来支持事实。当没有新规则产生时，这个系统就可以用于估计了，例如：

IF 要求的软件可靠度=非常高 AND 个人能力=低 THEN
　风险水平=高。

2.2.3 工作拆分结构

这个技术的软件估计包含了将产品进行拆分，直到拆分成的部件可以被独立估计为止。这个估计方法是基于对完整部件的现存数据库的类比，或专家估计，或上面描述的 DELPHI 技术。一旦所有的部件都被估计了，一个项目级的估计结果便可以综合这些部件的估计结果而得到了。

基于工作拆分结构（WBS）的技术是适用于计划和控制的。一个软件 WBS 实际包括两层，一展表示软件产品自身的结构，另一展表示用于建造这个产品的活动结构。产品层描述了软件的基础结构，它显示了不同的软件部件如何组成整个系统的。活动层显示了与每个给定软件部件相关的活动。

除了帮助估计，WBS 还可以用于成本计算和报告。WBS 的每一部分被分

配相应的预算和成本控制数字，这容许了成员可以报告他们花费在任何给定项目或部件的时间数量，而这些信息汇总后，便可以达到管理预算控制的目的。

如果组织对于它的所有的项目都使用标准的 WBS，那么随着时间的过去，这将产生一个非常有价值的数据库，它反映了组织的软件成本分布。这个数据库能帮助组织来开发相对于组织自己经验和实践的软件成本估计模型。

2.3 基于学习的技术

基于学习的技术是利用预先的和现在的知识来开发一个软件估计模型。

2.3.1 神经元网络

大多数使用神经元网络开发的软件模型都是利用后反馈来训练前项输入的网络。这些网络使用适合的神经元布局。网络先用一系列输入来训练网络参数，纠正由训练数据得到的输出，从而最小化预言误差。一旦训练完成了，适合的网络弧的权重是得到了。这样就可以利用这个网络来产生相应的估计了。

2.3.2 基于事例的推理

基于事例的推理是类比估计方法的增强形式。很明显，这种估计的准确性依赖于已完成项目的完成程度和数据的准确程度，因此使用这种估计方法要求有一个内容丰富、准确、可靠的软件过程数据库。

2.4 基于动态的技术

Forrestor 首先通过使用在信息反馈和循环特例中交互的连续的数值来形式化模型。它也被称为模拟方法。

系统动态方法包含了以下的概念：

- 以时序图的形式来动态地定义问题。
- 努力找到重要动态系统的内部活动视图。
- 把所有的系统概念认为是连续的数量。
- 定义出系统中独立的层次和它们的输入流和输出流比率。

- 形式化模型发现关系自己的动态问题的能力。
- 从结果模型得到理解和可应用的策略见解。
- 实施基于模型的理解和见解得到的更改。

2.5 基于回归的技术

基于回归的技术是最受欢迎的建立模型的方式。这个技术包括标准回归，鲁棒性回归等等。

2.5.1 标准回归方法

标准的回归（OLS：Ordinary Least Squares）指的是使用最小平方的通用线性回归模型的典型统计方法。它基于通用的最小平方法。

一个使用 OLS 的方法可表示为：

$$y_t = \beta_1 + \beta_2 x_{t2} + \ldots + \beta_k x_{tk} + e_t \tag{2.5}$$

这里 $x_{t2}...x_{tk}$ 是对于第 t 个观察的预言变量，$\beta_2...\beta_t$ 是反应系数，β_1 是截取参数，y_t 是对于第 t 个观察的反映值。误差 e_t 是一个概率分布的随机变量。OLS 方法通过最小化最小平方误差 r_i^2 来估计反应系数和截取参数，其中 r_i 是指对于第 i 个观察观测值与模型预言值的差异。

OLS 方法适合于如下的情况：

- 大量的数据提供。
- 没有数据项目丢失。
- 没有个别的例子。
- 预言变量互不相关。
- 当使用时，这些预言变量很容易理解。
- 回归变量是连续的或者离散的变量。

上面的每一条都是利用软件工程数据来产生鲁棒性好的、易懂的、结构化的成本估计模型的关键。

2.5.2 鲁棒性回归方法

它是对标准的 OLS 方法的提高。由于对软件标准定义理解的歧义、几个软件开发过程的并存、难得到的符合要求的质量和数量的数据的提供，从而软件工程数据通常包括大量的例外，鲁棒性回归方法消除了在观察软件工程数据中出现例外（outliers）的问题。

这里有两个统计的方法可用于鲁棒性回归方法。一个是基于最小中值平方方法，它与 OLS 方法非常的相似，唯一的不同是消除了所有 r_i^2 的中值；另一个方法是使用位于两个标准的偏差的平均反应变量，这个方法自动地消除例外，但只能在有足够观察数目的情况下才可以用，因此使用的自由度不高。

2.6 组合技术

组合技术融合了两种以上的技术来规范化适合的用于估计的模型形式。本文将只讨论 BAYESIAN 方法。

它包含了标准的回归技术的优点，又包含了专家的知识。它尽量地减少了不完整的数据带来的风险。本文将在 COCOMO II 系统校正改善的章节中具体地说明 BAYESIAN 方法。

第三章 COCOMO II 模型及其实际估计方法

COCOMO II 是对于早期 COCOMO 的升级。它关注于如下一些专题：非顺序和快速开发的过程模型；包含商业插件（commercial off the shelf ：COTS）包、复工程化、应用拆分和应用生成技术的复用驱动的方法；分布式中间件支持的面向对象的方法；软件过程成熟度作用和过程驱动质量估计。

这里总结了可对这些新型软件开发方法进行成本和进度估计的 COCOMO II 模型的研究，这也包括了模型决策的基本原理。COCOMO II 模型主要是一个可裁剪的软件规模模型的家族，包括对象点，功能点和源代码行；用于软件复用和复工程化的非线型模型；用于模型化相对软件非经济性的规模的指数驱动方法；以及对以前 COCOMO 工作量乘法因数成本驱动的几处添加、删除和更新。这个模型可以认为是一个框架，它可用于估计数据的收集和分析能力的更深的精确化，以及对模型估计能力的校正。

COCOMO II 模型的主要目标：

- 开发对应于现代的生命周期实践的软件成本和进度估计模型。
- 开发用于模型能力提高的软件成本数据库和工具支持手段。
- 提供数量化的分析框架和用于评价在软件生命周期成本和进度上软件技术提高的作用的一系列工具和技术。

终端用户程序部分不需要使用 COCOMO II 模型，因为它的应用开发通常是以小时和天来计时开发的，因此简单的基于活动的手工估计已经足够了。

COCOMO II 模型主要包括三个子模型：应用拆分模型、早期设计模型和后体系结构模型。他们可应用于不同的情况下。

对于估计应用生成器，系统集成或低层结构开发的能力，COCOMO II 模型是基于一个可裁剪的混合的应用拆分模型（对于早期原型化能力），以及对于并发的生命周期部分的两个不断细化的估计模型(早期设计模型：early design model 和后体系结构模型：post-architecture model)。

对于应用拆分模型（application composition model）部分，COCOMO II 模型是使用基于对象点的应用拆分模型。

3.1 COCOMO II 策略和基本原理

3.1.1 COCOMO II 模型策略

COCOMO II 模型策略（strategy）的四个主要的部分：

- 保持原有 COCOMO 的公开性
- 调节 COCOMO II 结构到将来的软件开发方法
- 调节 COCOMO II 子模型的输入和输出到可提供信息的层次
- 让 COCOMO II 子模型能被裁剪成项目的特定过程策略

COCOMO II 遵循了以前 COCOMO 的公开性的原则。因此，所有它的关系和算法可以公开的得到。而且所有的接口都可被设计成公开的、很好定义的和参数化的，因此补充的预处理程序（模拟，基于实例，或其它规模估计模型）、后处理程序（项目计划和控制工具，项目动态模型，风险分析）以及高层包（项目管理包，产品协议帮助）都能直接同 COCOMO II 相融合。

COCOMO II 的开发和提高进化策略：

- 增量式地进行。
- 测试模型和他们的概念来获得一手的经验。
- 创建一个 COCOMO II 会员计划。
- 提供一个外部和内部开放的模型。
- 避免与 COCOMO. 81 的不必要的不兼容。
- 与几个模型的扩展进行实验。
- 平衡专家决策和数据决策的模型化。
- 开发一系列不断精确化的模型。
- 调整 COCOMO II 模型到将来软件生命周期实践的项目开发。

3.1.2 COCOMO II 模型的基本原理

提出这个可裁剪的混合的 COCOMO II 模型的基本原理(rationale)是基于以下三个主要假设的。

- 不象早期的 COCOMO 情况——当时只有单个的可采用的软件生命周期模型，现在的和将来的软件项目将对应于它们的特定过程驱动来裁剪

它们的过程。

- 被采用的软件成本估计的粒度须与支持软件成本估计所提供的信息粒度保持一致。
- 已知了假设 1 和 2 中的情况，COCOMO II 能使项目在项目初期使用粗粒度的成本驱动，而在后面的阶段中，逐渐使用细粒度的信息。

COCOMO II 提供了以下三个阶段系列的模型用于估计应用生成器，系统集成和低层软件项目：

- 最早的阶段或螺旋周期一般将涉及原型化，使用应用拆分模型能力。COCOMO II 的应用拆分模型支持这些阶段，以及生命周期中后续发生的其它原型化活动。
- 下一个阶段或螺旋周期一般涉及体系结构的选择或增加式开发模型的搜索。为了支持这些活动，COCOMO II 提供了早期估计模型。这个模型中的细节层次是与可提供信息的细节层次和本阶段要求的设计准确的细节层次相一致的。
- 一旦项目准备开发和维护一个领域的系统，它应该有一个完整的体系结构，它用于提供更准确的成本驱动输入的信息，从而加强成本估计的准确性。为了支持这个阶段，COCOMO II 提供了后体系结构模型。

3.2 开发工作量和进度的估计

3.2.1 工作量的估计

这里的工作量是使用软件的规模（size，以千代码行 KSLOC 为单位）来进行估计的。

早期设计模型应用于软件项目的初期，这时我们对于将要开发的产品的规模，目标平台的本质，参与项目人员的本质，或将使用的过程的细节特征都不太知道。这个模型可应用于应用生成器，系统集成和低层开发部分。

早期设计模型使用 7 个工作量乘法因数 EM（包含 SCED）来调整额定的工作量，公式如下（上面公式为额定工作量公式，下面公式为调整的工作量的公式）：

$$PM_{NS} = A * Size^{E} * \left[\prod_{i=1}^{6} EM_i\right]$$
$$PM_{ADJUSTED} = PM_{NS} * SCED \qquad (3.1)$$

后体系结构模型是最详细的估计模型，当软件生命体系结构已经被开发后，它便可被使用了。这个模型可用于应用生成器，系统集成和低层开发部分的开发和维护中。

后体系结构模型使用 17 个工作量乘法因数 EM（包含 SCED）来调整额定工作量。如此多的乘法因数是源于开发后期的大量可提供的知识，公式如下（上面公式为额定工作量公式，下面公式为调整的工作量的公式）：

$$PM_{NS} = A * Size^{E} * \left[\prod_{i=1}^{16} EM_i\right]$$
$$PM_{ADJUSTED} = PM_{NS} * SCED \qquad (3.2)$$

3.2.2 进度的估计

COCOMO II 也提供了进度估计的能力。对于所有三个 COCOMO II 阶段的初始基准进度公式如下（上面公式为额定进度公式，下面公式为调整的进度的公式）：

$$TDEV_{NS} = C * (PM)^{(D+.02*(E-B))}$$
$$TDEV_{ADJUSTED} = TDEV_{NS} * \frac{SCED\%}{100} \qquad (3.3)$$
$$C = 3.67, D = 0.28, B = 0.91$$

这里调整后的 TDEV 是从产品的需求基准决定到验证产品达到它的需求的验收活动结束为止的以月计算的日历时间。PM 是估计的人月数（不包括 SCED），D 是 TDEV 规模的基指数，E 是由项目规模因素之和得来的工作量规模指数，B 是用于工作量公式中的规模的基指数。上面这几个参数都可以进行校正。SCED%是对于 SCED 工作量乘法因数的压缩/扩张的百分比。

3.2.3 应用拆分模型的工作量和进度的估计

应用拆分模型使用对象点来估计工作量和进度。对象点估计是相对较新的软件规模化方法，而且它非常适合用于应用拆分部分的实践估计。它也很适合相关的原型化工作量的估计。

COCOMO II 的对象点估计过程如下：

1) 得到对象的数目：估计构成这个应用的窗口（SCREEN），报告（REPORT）和 3GL 部件的数目。
2) 依据特征，将这些对象实例分类成简单、中等和困难复杂水平。具体使用如表 3.1。

表 3.1 对象实例复杂等级表

Table3.1 Object complexity level

对于窗口				对于报告			
包含的视图数目	数据表的数目			包含的节数目	数据表的数目		
	<4	<8	>=8		<4	<8	>=8
<3	简单	简单	中等	0 或 1	简单	简单	中等
3~7	简单	中等	困难	2 或 3	简单	中等	困难
>=8	中等	困难	困难	>=4	中等	困难	困难

3) 使用表 3.2 权衡每个单元中的数目，得到响应的数值。表中权值表示了要实现此复杂层上的一个实例所需的相对工作量，即权重。

表 3.2　复杂权重表

Table3.2　complexity weight

对象类型	复杂度权重		
	简单	中等	困难
窗口	1	2	3
报告	2	5	8
3GL 部件			10

4) 决定对象点：累加所有带权的对象实例从而得到对象点的数目。
5) 估计在这个项目中能达到的复用的百分比 REUSE。计算将开发的实际对象点数目 NOP =（对象点数目）（100-%REUSE）/100。
6) 利用表 3.3，决定生产率 PROD = NOP /人月。

表 3.3　生产率表

Table3.3 Productivity ratio

开发者经验和能力	非常低	低	额定	高	非常高
ICASE 成熟度和能力	非常低	低	额定	高	非常高
PROD	4	7	13	25	50

7) 计算出估计的人月数 PM = NOP/PROD。

3.2.4 输出范围

一些 COCOMO II 的用户希望 COCOMO II 的输出为估计的范围，而不是点估计。一旦从被选的模型中得到了最可能的工作量估计 E，便可利用下表来决定一系列的乐观和悲观的估计范围。

表 3.4 估计范围表

Table3.4 The scale of estimation

模型	乐观估计	悲观估计
应用拆分模型	0.50E	2.0E
早期设计模型	0.67E	1.5E
后体系结构模型	0.80E	1.25E

进度公式可使用这些工作量的范围值，从而得到相应的进度范围值。

3.3 新生成的代码规模

功能点成本估计方法是基于软件项目中功能的数目和一系列个人项目因素。因为它们是基于项目早期提供的信息的，所以实际中功能点是非常有用的估计方法。简要的功能点的总结和支持 COCOMO II 的功能点的计算方法如下所述。

功能点估计通过数量化处理与主要的外部数据、控制输入或输出以及文件类型相关的功能的信息来估计软件项目的规模。

五个功能类型定义如下：

- 外部输入：每个用户的数据或用户控制输入类型，它们指的是输入正在估计的软件系统的外部边界，以及添加或改变逻辑内部文件中的数据。
- 外部输出：离开正在估计的软件系统的外部边界的每个用户数据或控制输出类型
- 内部逻辑文件：作为逻辑内部文件类型，在软件系统中用户或控制信息的每一个主要的逻辑组。它包含软件系统产生、使用和维护的每个逻辑文件。
- 外部接口文件：软件系统间传送或共享的文件被计算为系统的外部接口文件类型。
- 外部查询：作为外部查询类型，每个单独的输入输出混合体，一个输入将导致和产生一个立即的输出。

通过复杂程度，我们将这些功能类型的实例进行分类。复杂程度定义了一系列的权重，它们将被用于调整功能点的数量。这就是 COCOMO II 使用的功能点规模规范。通常，功能点过程包含了相对应于每个特征的等级范围 0.0 到 0.05 的 14 个应用特征的影响程度（DI）。这 14 个影响程度值加到一起，再加上基值 0.65，便得到通常的特征调整因素的范围为 0.65 到 1.35。

这 14 个特征的每一个，例如：分布式功能、性能和可复用性，都对估计的工作量有着最大 5%的影响。但这与 COCOMO 的经验是不一致的，因此 COCOMO II 使用未调整功能点来规模化软件项目，并且在 COCOMO II 模型的公式中应用复用因素、成本驱动工作量乘法因数和指数规模因素来调整工作量的估计结果。

计算未调整功能点的 COCOMO II 的过程如下，它可用于早期设计模型和后体系结构模型。

1) 利用类型决定功能计算量：未调整功能点应该由一位领导技术人员基于软件需求和设计文档中的信息来计算。五个用户功能类型的每个的数量都应该被估算（内部逻辑文件 ILF、外部接口文件 EIF、外部输入 EI、外部输出 EO、外部查询 EQ）。
2) 决定复杂等级：依据包含的数据成分类型的数量和借鉴的文件类型的数量，将每个功能计算分类成低、中、高三个复杂等级。具体使用如表 3.5。

表 3.5 功能点复杂度等级表

Table3.5 Function point complexity level

对于 ILF 和 EIF				对于 EO 和 EQ				对于 EI			
记录分量数目	数据分量数目			文件类型数目	数据分量数目			文件类型数目	数据分量数目		
	1~19	20~50	>50		1~5	6~19	>=20		1~4	5~15	>=16
1	低	低	中	0~1	低	低	中	0~1	低	低	中
2~5	低	中	高	2~3	低	中	高	2~3	低	中	高
>=6	中	高	高	>=4	中	高	高	>3	中	高	高

3) 应用复杂权重：使用如表 3.6，权衡每个单元的数量。权重反映了对于用户功能的相对的数值。

表 3.6 功能点复杂度权重表

Table3.6 Function point complexity weight

功能类型	复杂度权重		
	低	中	高
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

4) 计算未调整功能点：将所有的带权重的功能点加到--起，就得到了未调整功能点的数目。

另外，还要将功能点转化为代码行。为了为估计模型确定工作量和进度提供合适的规模输入，未调整功能点必须转化为实现语言的源代码行，这样才能评价每功能点实施的相对准确性。在早期设计模型和后体系结构模型中，COCOMO II 使用下表来转化未调整功能点到等量的 SLOC，然后除以 1000 转化为 KSLOC，这样便可代入估计公式进行工作量和进度的估计了。

表 3.7 部分编程语言的功能点与代码行转化表

Table3.7 Converting unadjusted function point to SLOC

语言	SLOC/UFP
汇编	320
C	128
C++	29
BASIC（编译）	91
BASIC（解释）	128
PASCAL	91
PROLOG	64

3.4 破损量

COCOMO II 使用一个破损量百分比 REVL 来调整产品的规模。破损量反映了在一个项目中的需求不稳定性。这是由于需求不稳定性而导致的丢弃的代码的百分比。例如：一个项目提交了 100，000 条指令，但丢弃了 20，000 条指令，因此此项目的 REVL 为 20。这将用于为了 COCOMO II 估计而将项目的有效规模调整到 120，000 条指令。REVL 因素不用于应用拆分模型，因为在应用拆分模型中，一定程度的产品迭代已经被估计，并包含在数据校正之中了。

则有如下的利用 REVL 计算软件规模的公式：

$$Size = \left(1 + \frac{REVL}{100}\right) * Size_D \tag{3.4}$$

这里的 $Size_D$ 是指复用等量的提交的软件规模。

3.5 对于复用的调整

COCOMO II 对软件复用的处理使用一个非线型估计模型，公式如下：

$$EKSLOC = AKSLOC * AAM$$

$$AAM = \begin{cases} \dfrac{AA + AAF * (1 + [0.02 * SU * UNFM])}{100}, & \text{当} AAF \le 50\text{时} \\ \dfrac{AA + AAF + (SU * UNFM)}{100}, & \text{当} AAF > 50\text{时} \end{cases} \tag{3.5}$$

$$AAF = 0.4 * DM + 0.3 * CM + 0.3 * IM$$

另外，我们可使用 AT 来进一步细化复用模型代码的开发。其中，AT 是指由使用自动转换的复工程得来的改编 AKSLOC 的百分比。如下式，EKSLOC 即为人工完成的代码规模（除去自动转换的代码量）。

$$EKSLOC = AKSLOC * \left(1 - \frac{AT}{100}\right) * AAM$$

$$AAM = \begin{cases} \dfrac{AA + AAF * (1 + [0.02 * SU * UNFM])}{100}, & \text{当} AAF \le 50\text{时} \\ \dfrac{AA + AAF + (SU * UNFM)}{100}, & \text{当} AAF > 50\text{时} \end{cases} \tag{3.6}$$

$$AAF = 0.4 * DM + 0.3 * CM + 0.3 * IM$$

这包含了采用改编的软件数量的估计（AKSLOC）和三个更改程度参数：设计更改百分比（DM），代码更改百分比（CM）和用于集成复用软件而对初始集成工作量进行更改的百分比（IM）。

软件理解增量（SU）如下表。SU 以百分比的数值形式来表示。SU 由三个种类的主观均值来得到。

表 3.8 SU 等级表

Table3.8 SU level

	非常低	低	额定	高	非常高
结构	非常低的内聚力，高的耦合，线形编码	适当低的内聚力，高耦合	合理地结构化，一些弱的区域	高内聚力，低耦合	强壮的模块化，信息隐藏在数据控制结构里
应用清晰性	在程序和应用之间没有匹配	程序和应用之间的一些相关	程序和应用之间适当的相关	程序和应用之间好的相关	程序和应用之间清晰的匹配
自己描述	模糊的代码；文件丢失，模糊的或过时的	一些代码评论和头部；一些有用的文件。	适当等级的代码评论、头部和文件	好的代码评论和头部；有用的文件；一些弱的领域	自己描述代码；现代的很好组织的带有设计基本原理的文件。
EKSLOC 的 SU 增加量	50	40	30	20	10

其它的非线型复用增量与评价和同化（AA）的程度有关。AA 用于决定是否一个完全复用的软件模型是与应用成比例的，以及在整个产品描述中集成它的描述。下表为评价和同化增量提供了等级范围和数值，同样 AA 也是百分比。

表 3.9 AA 等级表

Table3.9 AA level

AA 增加量	AA 工作量等级
0	无
2	基本模块搜索和文档化
4	一些模块测试和评价(T&E)，文档化
6	考虑的模块 T&E，文档化
8	扩展的模块 T&E，文档化

用于更改现存软件的工作量数量的公式是个包含多个参数的函数，它的参数不仅仅是更改数量（AAF）和对现存软件的理解力（SU），也包括程序员与软件的相对不熟悉度（UNFM）。UNFM 参数以 SU 工作量增量的乘数来使用。UNFM 如下表：

表 3.10 UNFM 等级表

Table3.10 UNFM level

UNFM 增加量	不熟悉的等级
0.0	完全熟悉
0.2	大多数熟悉
0.4	一些熟悉
0.6	可考虑地熟悉
0.8	大多数不熟悉
1.0	完全地不熟悉

这个复用模型的公式用于决定与改编指令相当数量的源代码行（EKSLOC）。而我们得到的 EKSLOC 将作为规模参数输入到 COCOMO II 中。

在公式的实际使用中，如果没有将被使用的部件进行了更改，即没有使用 DM 和 CM，则也没必要使用 SU。如果部件的代码进行了更改，那么 SU 可应用于此规模估计公式。

而对于复工程或自动转化的工作量应做如下调整，即附加上自动转化的工作量。

$$PM_{AUTO} = \frac{AdaptedSLOC \times \left(AT/100\right)}{ATPROD} \tag{3.7}$$

COCOMO II 复工程和自动转化估计方法包含了参数 AT，即自动转换和复工程化代码的百分比（具体取值如表 3.11）。ATPROD 表示自动转化的生产率，单位为代码行/人月。AdaptedSLOC 为自动转化的代码行数。

表 3.11 自动复工程化转化表

Table3.11 automatic re-engineering translation ratio

复工程化目标	自动转化率 AT%
批处理	96%
SORT 批处理	90%
DBMS 批处理	88%
SORT&DBMS 批处理	82%
互动式	50%

我们可以通过如下的公式来估计代码自动转换的持续时间 T_{AUTO} 或平均代码自动转换的人员水平 FSP_{AUTO}。

$$PM_{AUTO} = T_{AUTO} * FSP_{AUTO} \tag{3.8}$$

3.6 对软件维护的成本估计

当添加或更改的基代码占原来整个基代码的百分比少于或等于 20%，或者添加新代码时，即当不改变软件的主要功能时，COCOMO II 将使用复用模型来对该软件应用进行成本估计。基代码指的是已存在，而为了现在的项目将要更改的代码。当更改大于 20%时，COCOMO II 使用维护方法来估计成本。初始的维护规模可分别利用如下两个公式计算出来。当基代码规模已知和对基代码更改百分比已知的情况下，我们使用下面的公式。

$$\begin{aligned}(Size)_M &= [(BaseCodeSize)\cdot MCF]\cdot MAF \\ (Size)_M &= (SizeAdded + SizeModified)\cdot MAF\end{aligned} \tag{3.9}$$

公式 3.9 中的第二个子公式是由第一个子公式通过下面的公式 3.10 转化过来的。

$$\begin{aligned}MCF &= \frac{SizeAdded + SizeModified}{BaseCodeSize} \\ MAF &= 1+\left(\frac{SU}{100}UNFM\right)\end{aligned} \tag{3.10}$$

其中，MCF 为维护更改因素，MAF 为维护调整因素。

软件维护的工作量和进度的求法如下：

软件维护成本估计的工作量估计公式与 COCOMO II 后体系结构开发模型中的工作量估计公式一样。但成本驱动因素中不包含 SCED 和 RUSE，因为这两个因素已不适合应用在软件维护的估计。另外 RELY 也将使用一系列不同的工作量乘法因数来估计软件维护。

RELY 维护成本驱动的取值如表 3.12：

表 3.12　RELY 等级表

Table3.12 RELY level

	非常低	低	额定	高	非常高
RELY 要求的软件可靠性	微小的麻烦	低的，容易恢复的损失	适当地，较容易恢复的损失	高金融风险损失	人类生命的风险损失
	1.23	1.10	1.00	0.99	1.07

工作量估计公式如下：

$$PM_M = A * (Size_M)^E * \prod_{i=1}^{15} EM_i \tag{3.11}$$

我们可以通过如下的公式来估计维护活动的持续时间 T_M 或平均维护人员水平 FSP_M。

$$PM_M = T_M \cdot FSP_M \tag{3.12}$$

3.7 规模因素和工作量乘法因数

3.7.1 规模因素

COCOMO II 模型的每个规模因素都有一定范围的等级层次，从特别低到特别高。每个等级层次有一个权重 W，权重的特定值称为一个规模因素值。我们可以把所有的一个项目的规模因素值 W_i 加起来，从而得出一个规模指数 E，它将作为软件规模的指数来调整工作量和进度的估计结果。

COCOMO II 模型包括五个规模因素，它细化了早期模型的规模因素。其中前例性（PREC）和开发灵活性（FLEX），这两个规模因素很好地抓住了早期 COCOMO 的组织式，半独立式，嵌入式这三种模式的不同。体系结构/风险解决（RESL）这个因素融合了 ADA COCOMO 中的两个规模因素。组凝聚力（TEAM）规模因素解释了由于同步化项目的参与者（用户、客户、开发员、维护员、接口设计员，及其他）的困难而产生的项目动荡和熵（平均信息量）的根源。这些困难可能源于参与者的目标和文化的不同；协调目标的困难；参与者缺乏作为一个组的经验和操作的熟悉等等。过程成熟度（PMAT）决定 PMAT 的过程是围绕软件工程协会的能力成熟度模型（CMM）开展的。这里有两种等级化过程成熟度的方法。第一种采用了基于 CMM 的组织评价的结论。第二种是围绕软件工程协会的能力成熟模型（CMM）的 18 个关键过程域（KPA）开展的，决定 PMAT 的过程就是决定对于每个 PKA 的依从的百分比。

3.7.2 工作量乘法因数

早期设计模型中我们使用了工作量乘法因数的一个简练集。早期设计成本驱动可由综合的后体系结构模型工作量乘法因数而得到，如下表 3.13：

表 3.13 早期设计和后体系结构工作量乘法因数表

Table3.13 Early design and post-architecture effort multiplier

早期设计成本驱动	综合的相应的后体系结构的成本驱动
产品可靠性和复杂性 RCPX	RELY，DATA，CPLX，DOCU
必须的复用 RUSE	RUSE
平台困难 PDIF	TIME，STOR，PVOL
人员能力 PERS	ACAP，PCAP，PCON
个人经验 PREX	AEXP，PEXP，LTEX
工具 PCIL	TOOL，SITE
进度 SCED	SCED

COCOMO II 后体系结构模型使用 17 个工作量乘法因数来调整额定工作量，以便于准确地反映产品开发的工作量。它们被分为四类：产品、平台、人员和项目。

产品因数：要求的软件可靠性（RELY）、数据库规模（DATA）、产品复杂度（CPLX）、要求的复用能力（RUSE）、匹配生命周期需要的文件（DOCU）；

平台因数：执行时间限制（TIME）、主要存储的限制（SOTR）、平台不稳定性（PVOL）；

人员因数：分析员能力（ACAP）、程序员能力（PCAP）、应用经验（AEXP）、平台经验（PEXP）、语言和工具经验（LTEX）、人员的持续性（PCON）；

项目因数：软件工具的使用（TOOL）、多场所开发（SITE）、要求的开发进度（SCED）。

COCOMO II 早期设计和后体系结构模型的每个工作量乘法因数都有一定范围的等级层次，从特别低到特别高。每个等级层次有一个权重。我们将把所有的工作量乘法因数的乘积作为乘数来调整开发工作量和进度。

3.7.3 举例说明

这里，我将举个例子来说明如何来确定早期设计模型的工作量乘法因数 PERS（人员能力）的取值。我们可以通过下表 3.14 各个等级所符合的条件来确定 PERS 的等级，然后便可以查到相应的数值，从而利用这个确定的 PERS 的数值代入公式来对工作量进行估计。

表 3.14 PERS 等级表

Table3.14 PERS level

	特别低 2.12	非常低 1.62	低 1.26	额定 1.00	高 0.83	非常高 0.63	特别高 0.50
ACAP，PCAP，PCON 等级的总和	3，4	5，6	7，8	9	10，11	12，13	14，15
合并 ACAP 和 PCAP 百分比	20%	39%	45%	55%	65%	75%	85%
每年的人员替换率 PCON	45%	30%	20%	12%	9%	5%	4%

这种方法类似地应用于其它规模因素和工作量乘法因数的取值过程。

3.8 COCOMO II 模型公式总结

3.8.1 规模化公式

$$
\begin{aligned}
Size &= \left(1+\frac{REVL}{100}\right)*(NewKSLOC+EquivalentKSLOC)\\
EquivalentKSLOC &= AdaptedKSLOC*\left(1-\frac{AT}{100}\right)*AAM\\
AAM &= \begin{cases}\dfrac{AA+AAF*(1+[0.02*SU*UNFM])}{100}, & \text{当}AAF\le 50\text{时}\\ \dfrac{AA+AAF+(SU*UNFM)}{100}, & \text{当}AAF>50\text{时}\end{cases}\\
AAF &= 0.4*DM+0.3*CM+0.3*IM
\end{aligned}
\tag{3.13}
$$

其中，AA 是指评价和同化的百分比；AAF 是指改编调整因素；AAM 是指改编调整修改量；AT 是指由使用自动转换的复工程得来的 AdaptedKSLOC 的百分比；CM 是指改动代码的百分比；DM 是指改动设计的百分比；IM 是指改编软件要求的集成改动的百分比；KSLOC 是指千源代码行；REVL 是指需求改进和不稳定的百分比；SU 是指软件理解度的百分比；UNFM 是指程序员对软件的不熟悉度；AdaptedKSLOC 是指改编的千源代码行；

EquivalentKSLOC 是指经改编转换的等量千源代码行；NewKSLOC 是指新建的千源代码行；Size 是指软件的规模（以千源代码行 KSLOC 为单位）。

3.8.2 后体系结构模型公式

$$PM = A * Size^{E} * \prod_{j=1}^{17} EM_i + PM_{Auto}$$
$$E = B + 0.01 * \sum_{j=1}^{5} SF_j \qquad (3.14)$$
$$PM_{Auto} = \frac{AdaptedKSLOC * \left(\frac{AT}{100}\right)}{ATPROD}$$

其中，A 是指工作量系数，现在设为 2.94；AT 是指由使用自动转换和复工程化得来的 AdaptedKSLOC 的百分比； ATPROD 是指自动转换生产率；B 是指用于工作量的规模基指数，现在设为 0.91；E 是指用于工作量的规模指数；EM_i(i=1,2,...,17)是指 17 个工作量的乘法因数；PM 是指开发新和改编代码的人月工作量；PM_{Auto}是指自动转换活动的人月工作量；SF_j(j=1,2,...,5)SF 是指 5 个规模因素；AdaptedKSLOC 是指改编的千源代码行；Size 是指软件的规模（以千源代码行 KSLOC 为单位）。

3.8.3 早期设计模型公式

$$PM = A * Size^{E} * \prod_{i=1}^{7} EM_i + PM_{Auto} \qquad (3.15)$$

其中，A 是指工作量系数；E 是指用于工作量的规模指数；EM_i(i=1,2,...,7)是指 7 个工作量的乘法因数；PM 是指开发新和改编代码的人月工作量；PM_{Auto}是指自动转换活动的人月工作量； Size 是指软件的规模（以千源代码行 KSLOC 为单位）。

3.8.4 进度公式

$$TDEV = \left[C * (PM_{NS})^{F}\right] * \frac{SCED\%}{100} \qquad (3.16)$$
$$F = (D + 0.2 * [E - B])$$

其中，B 是指用于工作量公式的规模基指数，现在设为 0.91；C 是指进度系数，现在设为 3.67；D 是指用于进度的规模基指数，现在设为 0.28；E 是指用于工作量公式的规模指数；F 是指用于进度的规模指数；PM_{NS} 是不包括 SCED 成本驱动和 PM_{Auto} 的估计的人月数（同时适应于早期设计模型和后体系结构模型的 PM 的额定值）；SCED 是指要求进度压缩的百分比；TDEV 是指以月为单位的进度。

第四章 COCOMO II模型的扩展

4.1 构造性阶段进度和工作量估计模型

COPSEMO（即 Constructive phased schedule and effort estimation model，构造性阶段进度和工作量估计模型）起初是作为构造性快速开发模型（CORADMO：Constructive RAD model）的基础来开发的，而 CORADMO 把估计的工作量和进度分配到软件生命周期的各个阶段。COPSEMO 弥补了对于 COCOMO.81 中的系统级和更小更短的开发项目的瀑布开发过程模型的估计结果的不完整性。虽然实际上是作为早期 CORADMO 的部分开发的，现在 COPSEMO 已经作为单个的、独立的模型了，它被当作其它 COCOMO II 模型扩展的基础。

COPSEMO 逻辑模型图示如下：

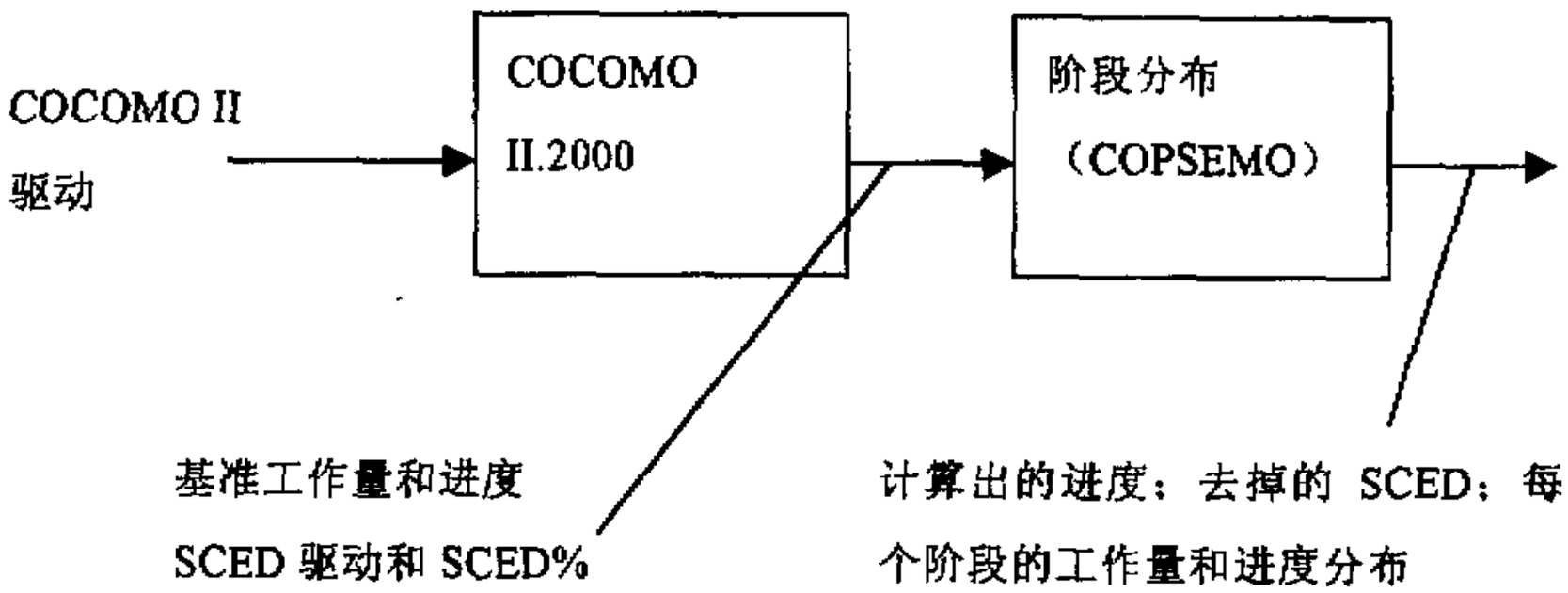

图 4.1 COPSEMO 逻辑模型图

Fig.4.1 COPSEMO logical model

4.2 构造性快速应用开发估计

CORADMO（即 Constructive rapid application development estimation，构造性快速应用开发估计）和它的同伴模型构造性阶段进度和工作量模型

（COPSEMO）用于消除经典 COCOMO 模型中的缺点。这些缺陷存在于：瀑布模型较弱的体系、没有反映现代的减少工作量的驱动以及小工作量的项目成本估计不完整性。CORADMO 是基于 RAD 的概念的，它作为用于加速应用开发的一系列策略。

这里增加了五个新的驱动，它们是复用和非常高级的语言（RVHL）、开发过程的复工程化（DPRS）、合作效率（CLAB）、体系结构及风险解决（RESL）和预安置资产（PPOS）。

CORADMO 逻辑模型图示如下：

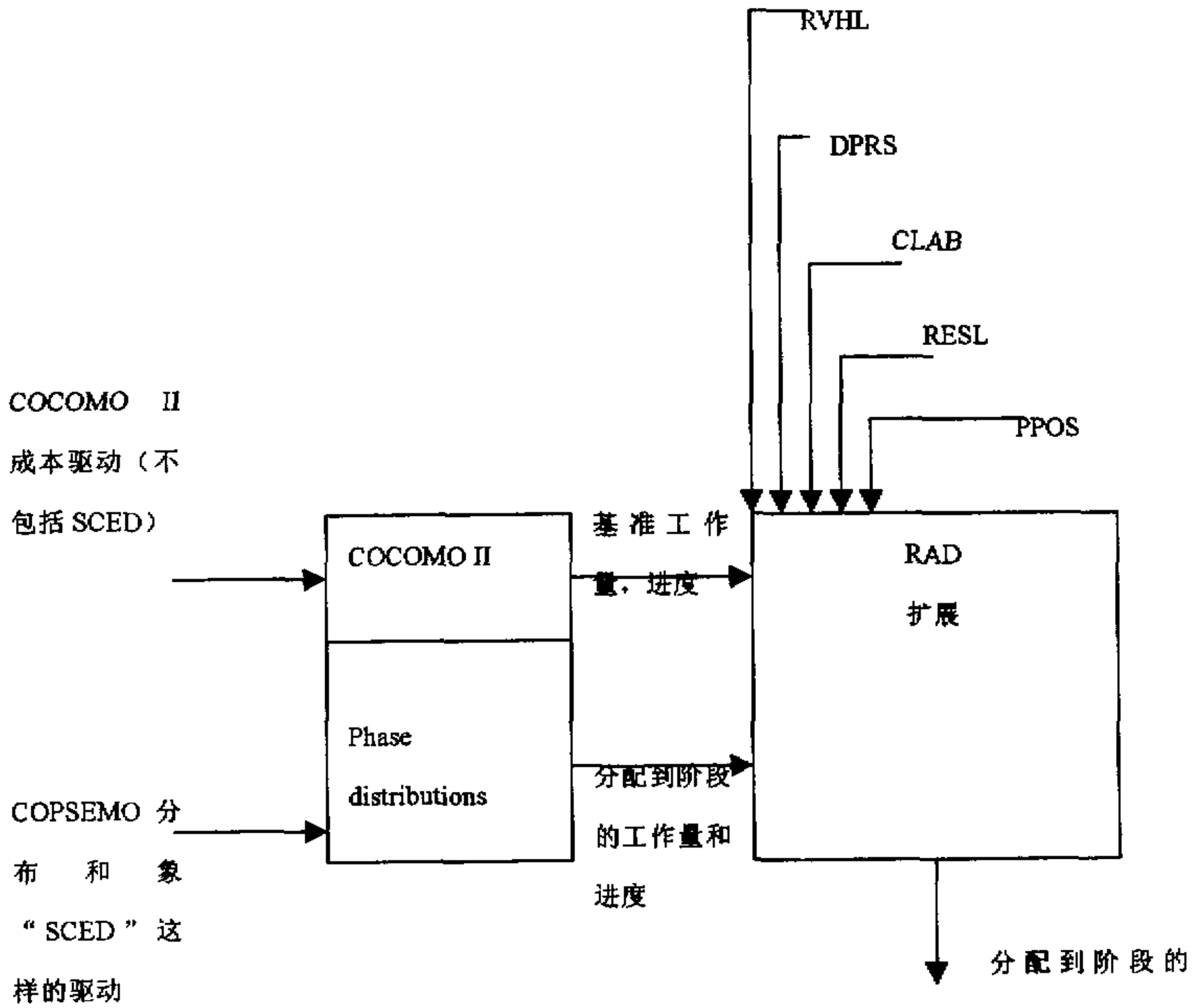

图 4.2 CORADMO 逻辑模型图

Fig.4.2 CORADMO logical model

4.3 COTS 集成估计

COCOTS（即 COTS integration estimation，COTS 集成估计）包含了四个相关的子模型，每个表示了 COTS 软件集成成本的四个主要来源之一。

1) 存取子模型

$$\text{初始过滤工作量 } (IFE) = \sum_{ALLCLASSES}\left[(\text{类中的}COTS\text{候选})(\text{类的平均初始过滤工作量})\right]$$

$$\text{详细存取工作量 } (DAE) = \sum_{ALLCLASSES,\ BYPROJECTDOMAIN}\left[(\text{类中的}COTS\text{候选})(\text{类的细节存取工作量})\right]$$

$$\text{最终项目存取工作量} = IFE + DAE$$

2) 裁剪子模型

$$\text{项目裁剪工作量} = \sum_{ALLCLASSES,\ BYPROJECTDOMAIN}\left[(\text{类中的}COTS\text{候选})\begin{pmatrix}\text{相对于复杂度,}\\ \text{类的平均裁剪工作量}\end{pmatrix}\right]$$

3) 粘合代码子模型

$$\text{粘合代码工作量} = A*\left[(SIZE)(1+CREVOL)\right]^{B} * \prod \text{工作量乘法因数}$$

其中，A 是指线性规模常量；

SIZE 是指以代码行或功能点衡量的粘合代码数；

CREVOL 是指由于需求变更或不稳定导致的粘合代码的重建的百分比；

B 是指体系结构的非线性的规模因素；

工作量乘法因数是指 13 个乘法因数的工作量调整因素。

4) 系统不稳定子模型

$$\text{系统不稳定工作量} = (\text{应用工作量})*\left\{\left[1+\left(\frac{SCREVOL}{1+REVL}\right)\right]E-1\right\}*(COTS\text{工作量乘法因数})$$

其中，应用工作量是指独立于 COTS 集成作用的新编码工作量；

SCREVOL 是指由于 COTS 不稳定和 COTS 需求变更引起的重建百分比；

REVL 是指由于需求变更而独立于 COTS 作用的重建的百分比；

E 为乘法因数。

而整个 COTS 集成工作量即为以上四个子模型的工作量的总和。

4.4 质量估计

专家决定的缺陷传入和缺陷消除子模型，它组成了对 COCOMO II 扩展的质量模型。它与 COCOMO II 的集成如下图 4.3。

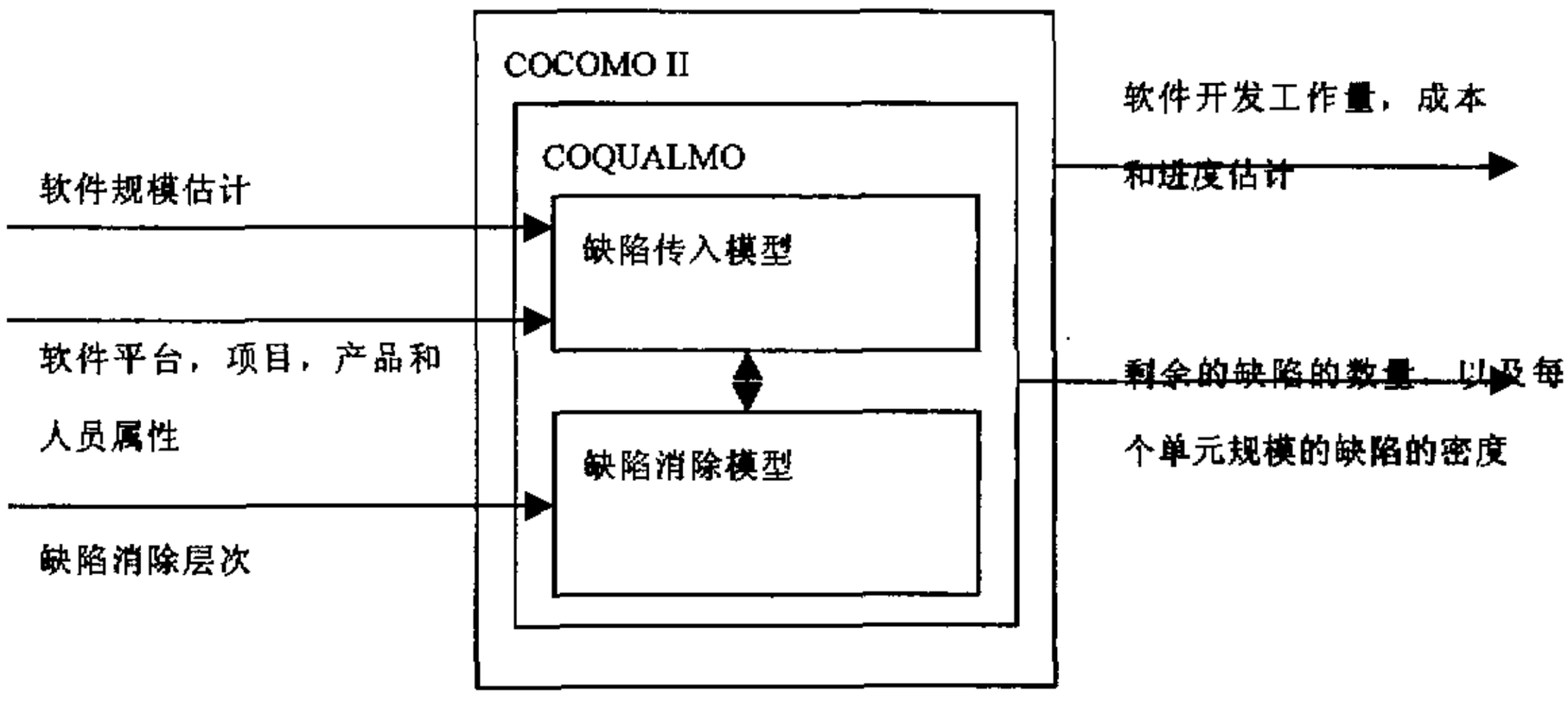

图 4.3 COCOTS 模型图

Fig.4.3 COCOTS model

4.5 生产率估计

COPROMO （即 Productivity estimation）是用于软件工程高级管理的计划决策帮助模型。这个模型是基于 COCOMO II 和 CORADMO 之上的，作为生产率估计的模型。

COPROMO 逻辑模型图示如下：

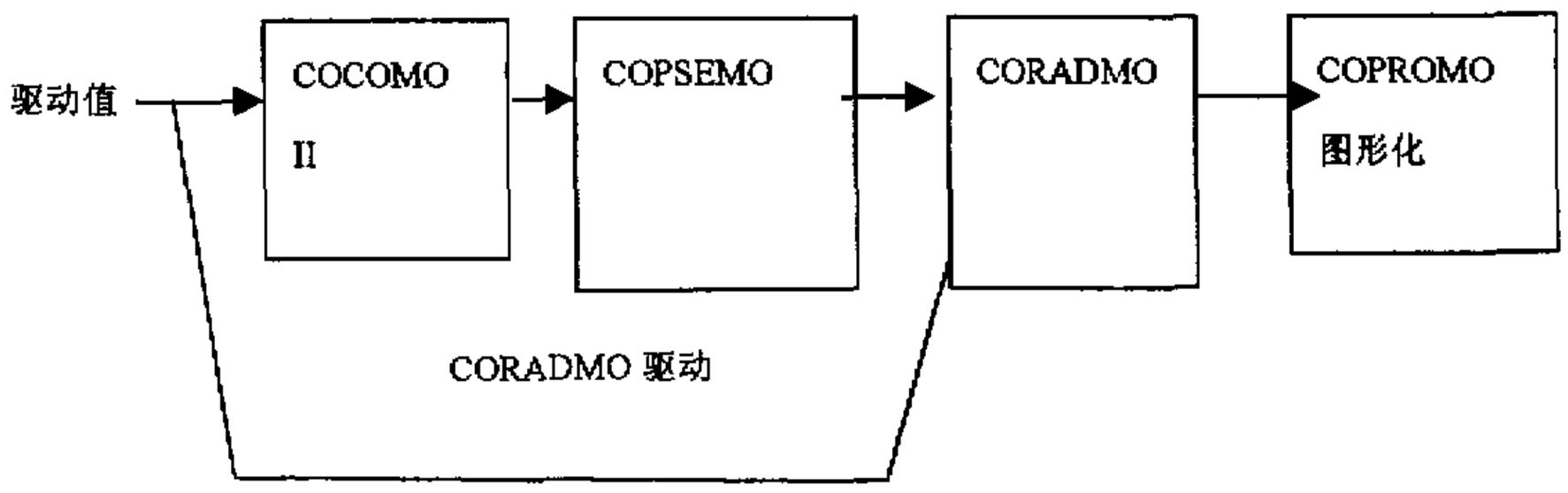

图 4.4 COPROMO 逻辑模型图

Fig.4.4 COPROMO logical model

4.6 风险评估

EXPERT COCOMO 就是风险评估模型（Risk assessment model）。专家 COCOMO 是对 COCOMO II 的一个扩展，它用于通过标识、分类、数量化和优先级化项目风险来进行项目计划。它可以检测出输入的异常，以及提供超出通用 COCOMO 估计的成本和进度的风险控制建议。

风险描述：风险是指无法在给定的成本和进度内，完成满意的软件产品。一条相关的规则如下：

IF （要求的开发进度<额定） AND （应用经验<额定）

THEN 项目有风险.

风险数量化公式如下：

$$\mathrm{Pr}ojectRisk = \sum_{j=1}^{\#categories} \sum_{i=1}^{\#categories\ risks} \left(风险等级_{i,j} * 工作量乘法因数之积_{i,j} \right)$$

其中，典型的风险等级分配如下：横轴为属性 i 的等级，纵轴为属性 j 的等级；

表 4.1 风险等级表

Table4.1 Risk level

	非常低	低	额定	高	非常高	特别高
非常低				适宜	高	非常高
低					适宜	高
额定						适宜
高						
非常高						

风险等级为：1 表示适宜的风险，2 表示高风险，3 表示非常高的风险；

工作量乘法因数之积=（驱动#1 工作量乘法因数）*（驱动#2 工作量乘法因数）……*（驱动#n 工作量乘法因数）；

如果风险包含了进度限制 SCED，则上面的公式转化为：工作量乘法因数之积=（SCED 工作量乘法因数）/（相对的进度）*（驱动#1 工作量乘法因数）*（驱动#2 工作量乘法因数）……*（驱动#n 工作量乘法因数）。

然后，我们可将风险等级之和进行一定的处理，从而得到有意义的相对风险预测。

第五章 COCOMO II 系统的分析设计与校正改善

本章将系统地阐述实际应用的 COCOMO II 系统的分析设计和实际使用规程，同时将结合东软软件股份公司的 CMM 和项目开发的实际情况对 COCOMO II 模型的参数进行了初步的校正和改善，并进一步论述如何才能更好地进行软件成本的工作量和进度的估计。

5.1 系统的实施背景

1986 年美国卡内基梅隆大学软件工程研究院（SEI）应联邦政府的要求，着手研究，并于 1987 年 9 月发布了软件过程能力成熟框架的大纲，在大纲中定义了软件过程评估、软件能力评价两种方法及相应的问卷调查表。经过 4 年的实践，SEI 将其发展成软件能力成熟模型 SW-CMM（简称为 CMM）。

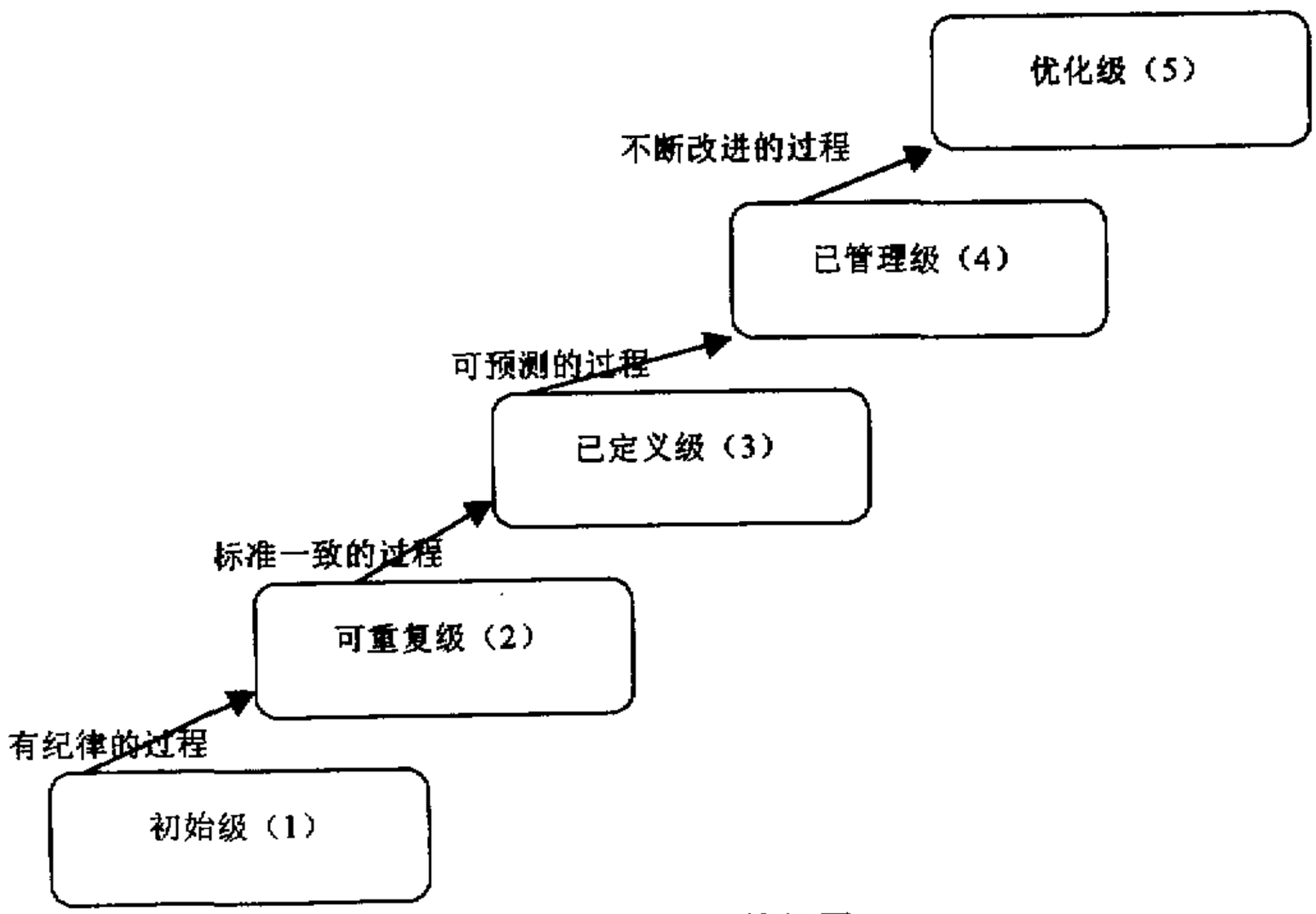

图 5.1 CMM 等级图

Fig.5.1 CMM level

SW-CMM 描述了一个软件组织进行软件过程定义、项目实施与测试、过程控制与改善时必须经历的 5 个渐进的成熟等级及每一成熟等级相应的关键过程域(key process area)和关键实践(key practice)。软件组织可以依据 SW-CMM 标识现有软件过程中存在的关键问题，制定相应的软件过程改善策略，使软件组织的过程能力不断成熟，从而提高一个软件组织始终如一地、可预见性地生产高质量软件产品的能力。

SW-CMM 为每个软件组织建立和改进他们的软件过程提供了一个阶梯式的过程成熟度的框架，这个阶梯是由五个成熟度等级所构成。每个成熟度等级为持续过程改进提供了一个渐进的基础。CMM 并不主张跨等级的跳跃，因为每个等级形成了一个必要的基础，从此基础出发才能到达下一个等级。每个成熟度等级又包括若干关键过程域，而每个关键过程域都包含一系列的目标，当这些目标全部实现时，软件过程中的一些重要的成分就稳定下来，就达到了一个成熟度等级。每达到一个成熟度等级，就建立起软件过程的一个不同的成分，导致组织过程能力的全面增长。

- 初始级（Initial）
- 可重复级（Repeatable）
- 已定义级（Defined）
- 已管理级（Managed）
- 优化级（Optimizing）

除了初始级以外，其它的成熟度等级都包含了若干个关键过程域，这些关键过程域都包含了若干关键实践，这些关键实践按照一些共同特点来进行组织。关键实践是对关键过程域起非常重要作用的一些基础设施或活动，只要这些关键实践认真地贯彻执行，就能帮助达到关键过程域的目标，目标实现了，组织的软件过程能力就相应地增强。

其中，CMM 的第 4 级是定量管理级。处于这一级的组织已经能够为软件产品和软件过程设定定量的质量目标，并且能对跨项目的重要软件过程活动的效率和质量予以度量；可以利用本组织的软件过程数据库汇集各项目软件过程产生的数据并加以分析。处于第 4 级时，该组织的各个软件过程是用各种确切定义并且一致的尺度来说明的；这些尺度是用以评价项目的软件过程和产物的定量基础。在第 2，3 级的基础上，第 4 级包括两个关键过程域，共含 32 个关键实践。第 4 级的两个关键过程域：定量过程管理 （包含 19 个关键实践）和软件质量管理（包含 13 个关键实践）。

东软集团东软股份公司于 2001 年 6 月已经通过了 CMM 三级认证，东软

软件股份公司的过程改善中心现在正处于从 CMM 三级到四级的过渡阶段。而 COCOMO II 系统是作为 CMM 四级的定量软件管理的一主要构成部分而进行开发的。

5.2 系统分析

COCOMO II 软件成本估计系统是公司为提高 CMM 管理水平而开发的项目过程管理系统的一个主要构成部分。而项目过程管理系统目的是建立一套便于使用的项目软件过程管理工具，保证项目按照组织定义的软件过程有序的进行，并符合组织的标准和要求。目前项目过程管理系统定位在管理和控制项目的软件过程和工作产品、与软件过程相关的文档等内容，并提供充分的过程管理功能，让使用者能够方便的用其来定制项目的软件过程，并对过程中的各个环节进行控制和监督，使得项目的软件过程符合组织的要求。项目过程管理系统的目的是使项目开发规范化，最终符合 CMM 的要求，因此系统的构建必须完全满足 CMM 所作的规定。

COCOMO II 系统将根据项目中软件规模和相关因素的描述，进而对模型的各个参数赋值，然后估计出该项目的工作量和进度。COCOMO II 系统的开发将有助于软件成本估计的有效进行，从而为以后的软件和过程的度量，分析以及决策提供大量的有用的定量的信息。COCOMO II 系统的层次结构图如下：

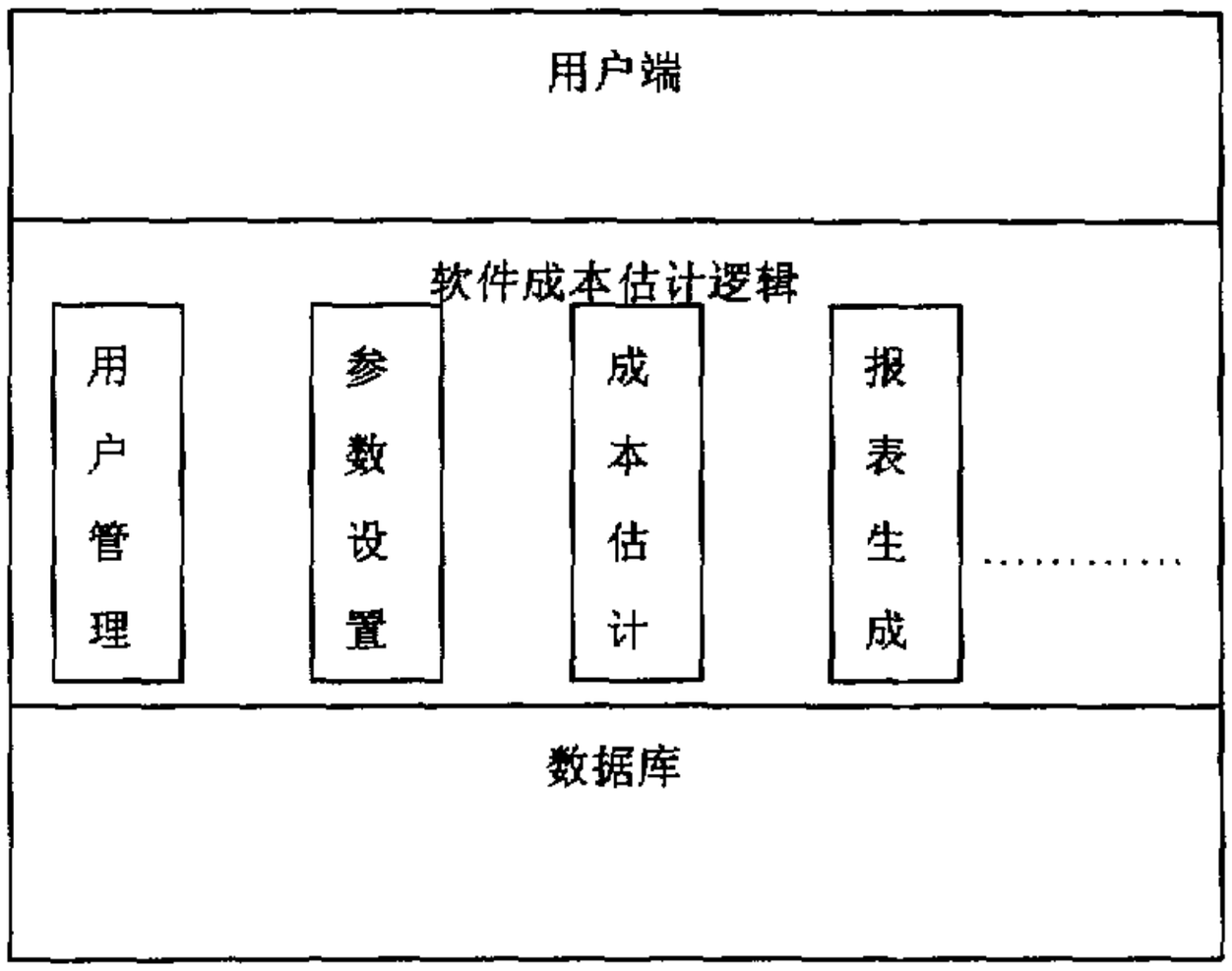

图 5.2 COCOMO II 系统层次图

Fig5.2 COCOMO II system hiberarchy graphy

5.2.1 系统开发的环境条件

为了更好地建设 COOCMO II 系统，公司应采取以下步骤：

1) 提高认识。提高公司领导和员工对 COCOMO II 软件成本估计系统建设的意义、重要性和必要性的认识程度，这是公司 COCOMO II 软件成本估计系统建设的思想基础。
2) 公司应建立好的管理机制。调动员工的积极性和创造性，培养员工对 COCOMO II 软件成本估计系统的兴趣。
3) 科学决策。公司应组织自己的技术和管理力量，聘请有一定水平的 COCOMO II 专家组成一个工作小组，对与 COCOMO II 系统建设有关的重大问题进行探讨。
4) 制定正确的软件成本估计策略。

只有正确地处理好上述几个关系，并认真考虑了上述几个方面的问题，公司的软件成本估计 COCOMO II 系统才会建设的更有成效。

5.2.2 COCOMO II 系统分析

本人使用 RATIONAL ROSE 对 COCOMO II 软件成本估计系统进行了系统分析。以下为一些主要的 COCOMO II 系统分析设计。

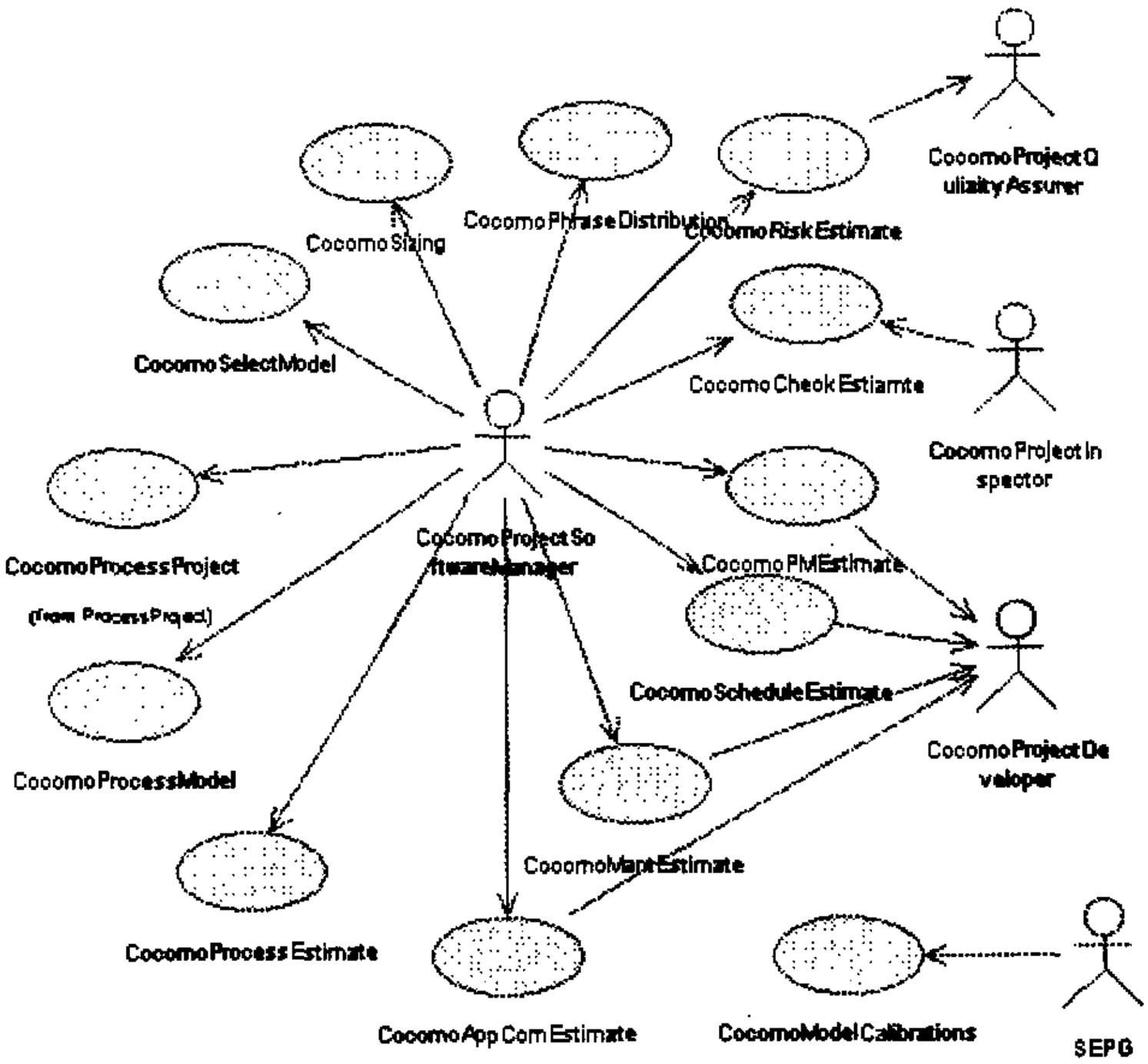

图 5.3　COCOMO II 系统用例视图

Fig.5.3 COCOMO II system usecase view

用例视图常常用来对系统要实现的功能做完整的分析和设计。这个用例视图（图 5.3）描述了整个系统的人员和用例之间的关系，它描述了整个系统需要实现的各个功能，例如：软件规模化、各个模型对应的工作量和成本估计以及模型校正等。本系统涉及的有如下的角色：

CocomoProjectSoftwareManager：项目软件经理

CocomoProjectInspector：项目估计监督员

CocomoProjectQualityAssurer：项目质量保证员

CocomoProjectDeveloper：项目开发者

SEPG：软件工程过程组（现在更名为过程改善中心）

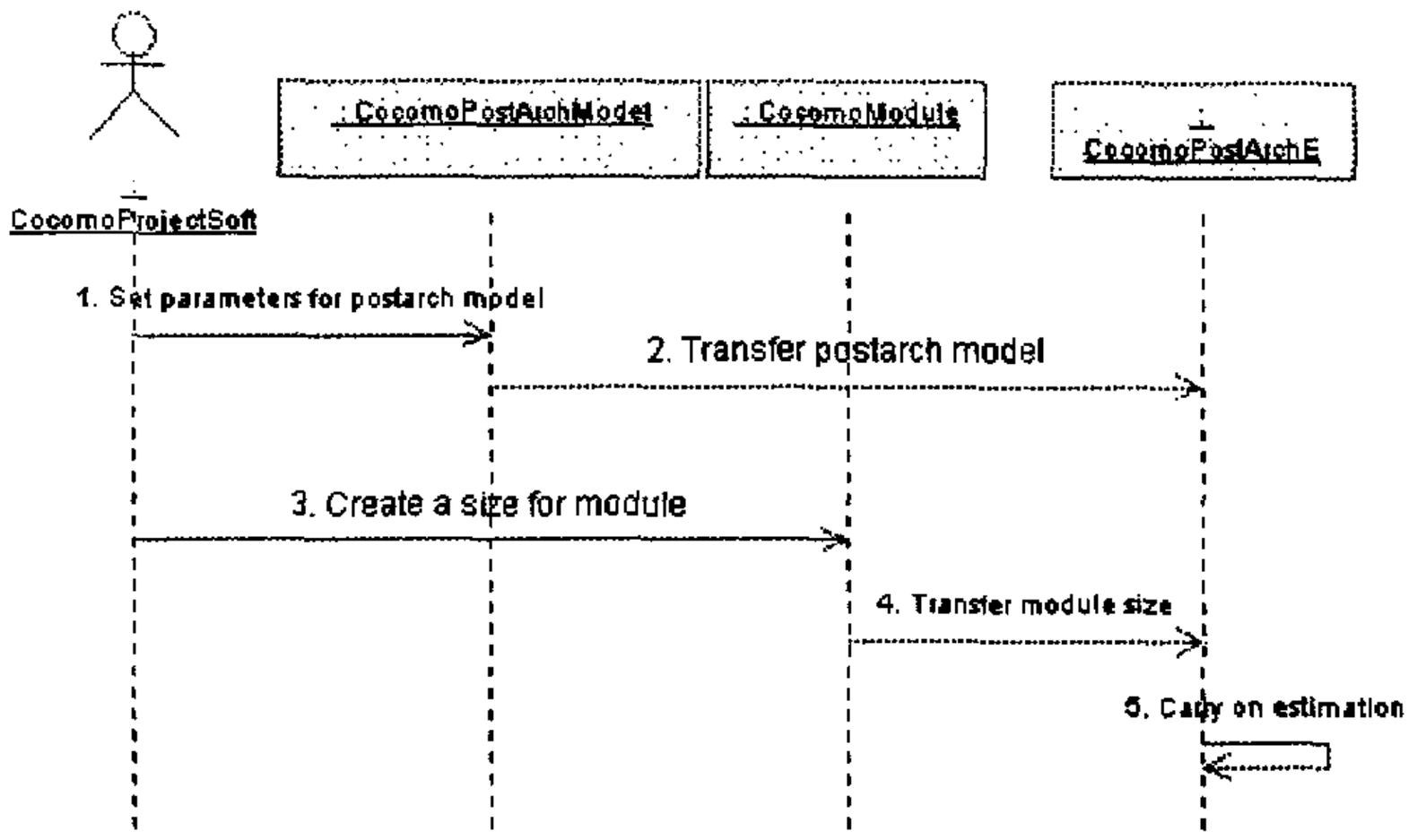

图 5.4 后体系结构模型的成本估计序列图

Fig.5.4 Sequence view of cost estimation in postarchitecture model

序列图常常用来显示某个用例的过程流程，是对系统用例的细化显示，框图顶部显示了涉及的角色，每个箭头表示角色与对象或对象与对象之间为完成所需功能而传递的消息。这个序列图（图 5.4）描述了如何使用后体系结构模型来估计项目工作量和进度。其它主要的序列图如下所述：

FunpointSizing：描述了如何利用功能点法来求出模块的新生成代码的规模。

ReuseSizing：描述了如何利用复用和复工程化的方法来求出模块的生成代码的规模。

AppComEstimate：描述了如何运用 COCOMO II 的应用拆分模型来进行项目初期的工作量和进度的估计。

PostArchModelGPBCalibration：描述了如何运用扩展的 BAYESIAN 分析的方法来对后体系结构的参数进行校正，并对 COCOMO II 中的后体系结构模型的校正参数的设置进行必要的更改，给出一定的模型校正报告。

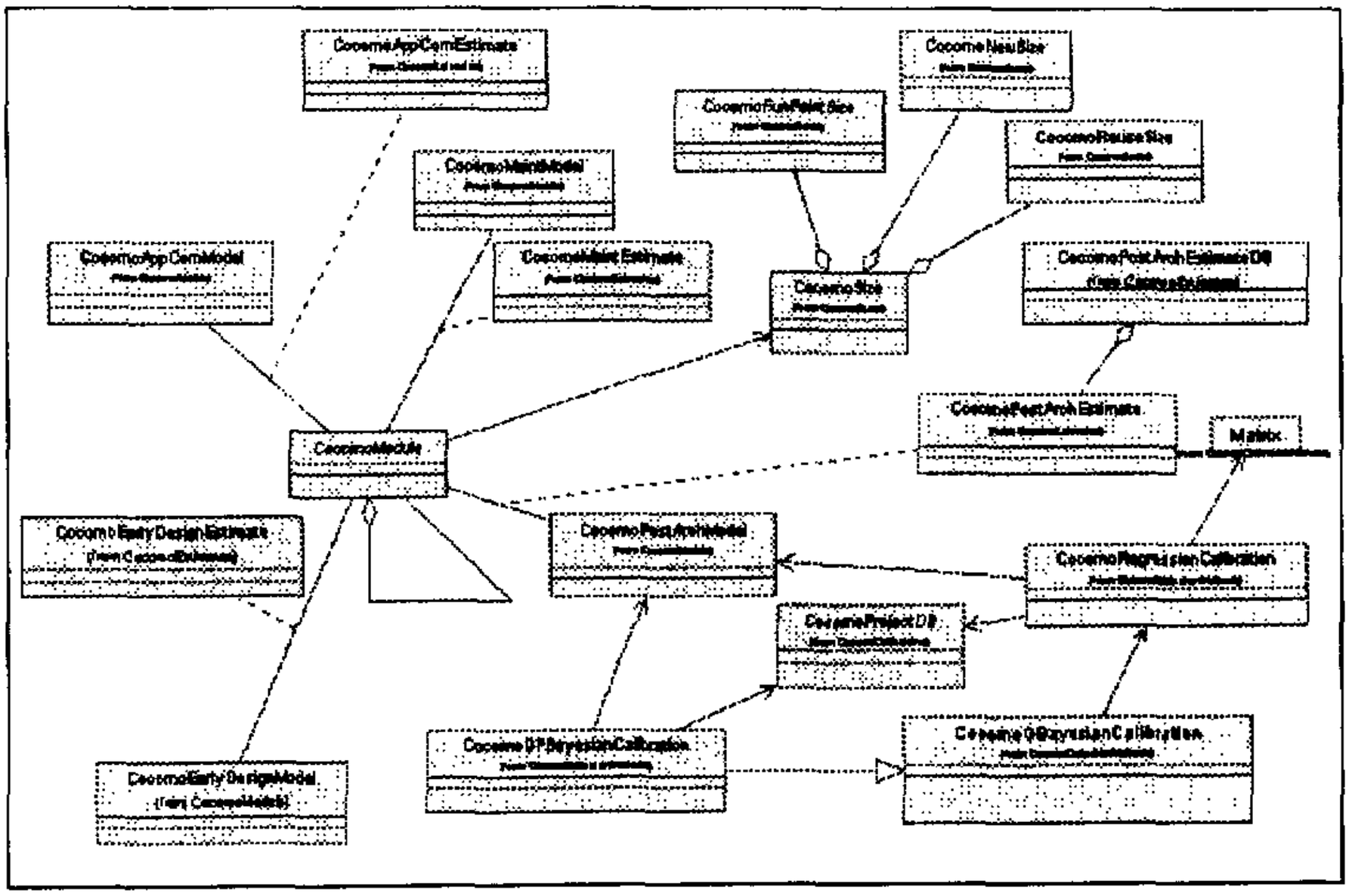

图 5.5　系统分析总体逻辑视图

Fig.5.5 Overall logical view of System analysis

逻辑视图显示了系统中类与类之间的交互，是对实现系统中所包含实现类及其关联的显示。这个逻辑视图（图 5.5）定义了整个系统的主要组成类，及其关系。

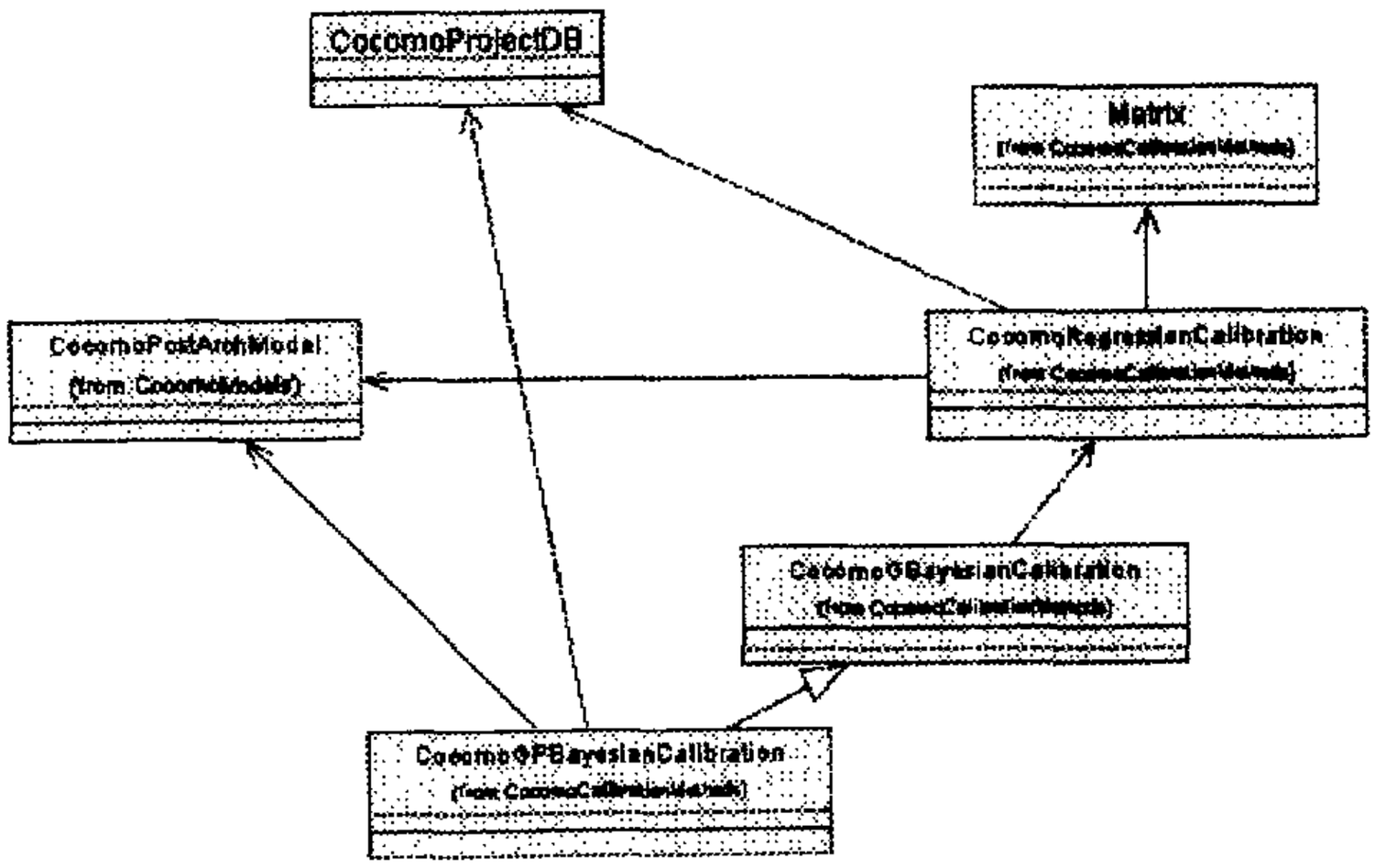

图 5.6 COCOMO II 系统模型参数校正逻辑视图

Fig.5.6 Logical view of model parameters' calibration in COCOMO II system

这个逻辑视图（图 5.6）定义了用于模型参数校正的几个类，及其关系。这里的 CocomoGPBayesianCalibration 类继承了 CocomoGBayesianCalibration 类，它使用存储在数据库中项目开发的实际数据，利用融合专家校正和实例数据校正的后验 BAYESIAN 方法来校正模型的参数，从而使校正修改后的模型具有更好的估计结果的准确性（具体内容将在后面论述）。

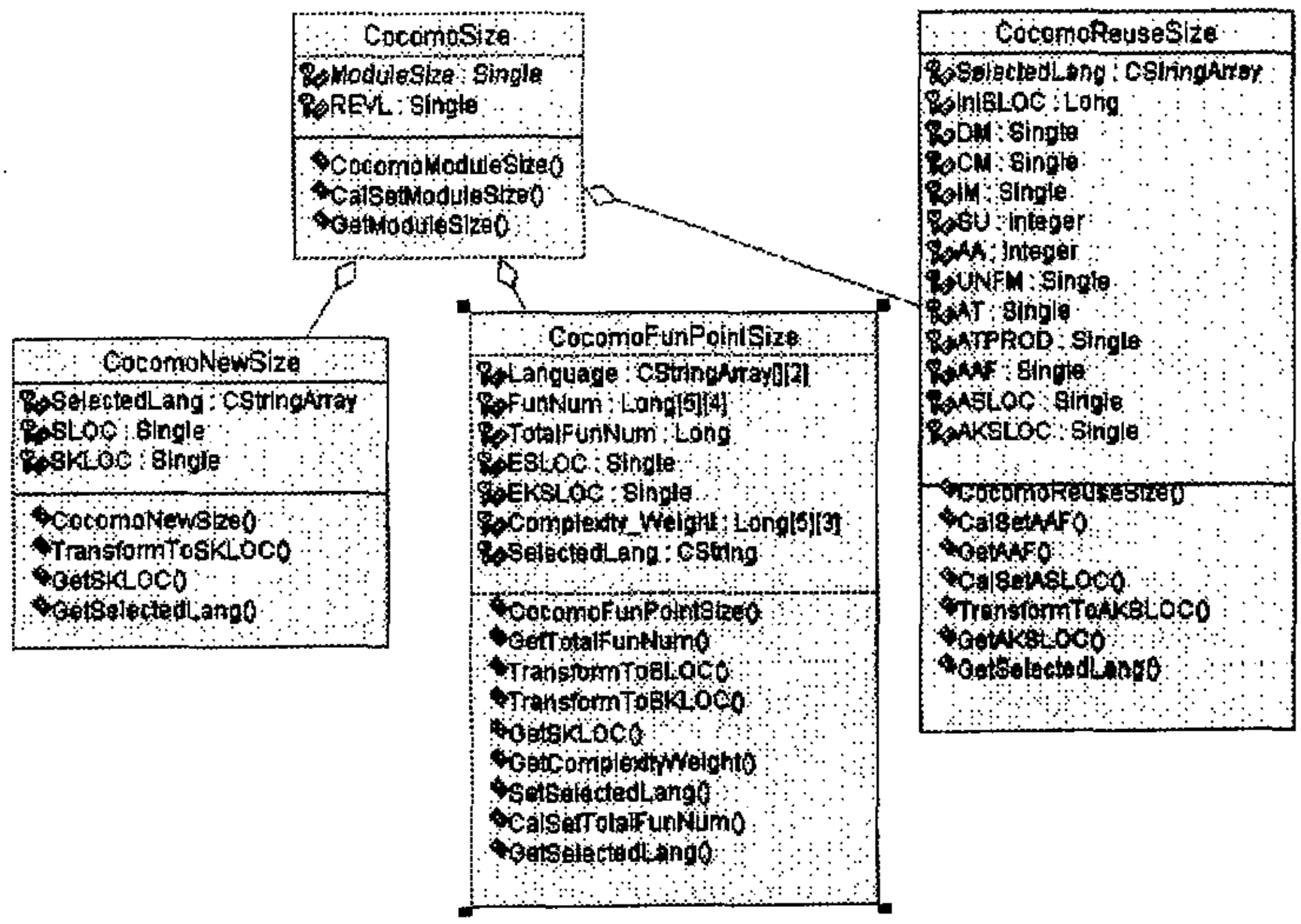

图 5.7 COCOMO II 系统规模化逻辑视图

Fig.5.7 Logical view of sizing in COCOMO II system

类包含信息和处理信息的功能。如图 5.7 中的 CocomoModule 类，即项目的软件开发模块类。它定义了软件模块的各个参数，及其设置，例如属性 ModuleSize（模块规模）、REVL（破损量）等；及其方法 CalSetModuleSize()（用于计算模块的总规模，并对 ModuleSize 赋值）、GetModuleSize()（用于得到 ModuleSize 的值）等。系统中主要的类如下：

CocomoMaintModel 类：维护估计模型类，它定义了用于估计维护工作的工作量和进度的模型的各个参数，及其设置。

CocomoPostArchModel 类：后体系结构模型类，它定义了后体系结构模型的各个参数，及其设置。

CocomoEarlyDesignModel 类：早期设计模型类，它定义了早期设计模型的各个参数，及其设置。

CocomoEarlyDesignEstimate 类：早期设计估计类，它定义了利用早期设计模型提供的参数来估计早期设计的项目的工作量和进度的方法。

CocomoAppComEstimate 类：应用拆分估计类，它定义了利用应用拆分模型提供的参数来估计应用拆分的项目的工作量和进度的方法。

CocomoMaintEstimate 类：维护估计类，它定义了利用维护模型提供的参数来估计维护的项目的工作量和进度的方法。

CocomoPostArchEstimate 类：后体系结构估计类，它定义了利用后体系结构模型提供的参数来估计后体系结构的项目的工作量和进度的方法。

CocomoGPBayesian 类：扩展的 BAYESIAN 分析类，它使用扩展的 BAYESIAN 分析的方法来校正参数。

CocomoMatrix 类：矩阵类，它定义了矩阵的基本的运算方法。

5.3 系统设计

5.3.1 系统开发的环境条件

- 操作系统：

  Windows98、Windows NT

- 数据库：

  Oracle 数据库

- 开发软件：

  VC++

- 其它支撑软件：

  Rational rose

5.3.2 系统的设计实现目标

本系统是采用面向对象方法和模块化程序设计的应用软件，具有很强的项目成本估计与项目数据管理功能，融项目信息管理、成本估计、风险估计、项目估计报表等功能于一体。通过建立 COCOMO II 系统，公司用户可方便地进行项目的成本估计，从而有效地控制项目的开发过程。

本系统是一个比较专业的软件成本估计的工具软件。它提供了用户管理、系统参数设置、项目开发成本估计和风险估计等功能，系统充分考虑了现在软件的多种开发模式、开发方法和工具的特点，因而系统的功能完善，使用方便。

5.3.3 系统的实际规程

系统首先对用户分配一定的使用权限，用户可以在权限内利用系统来估计软件成本、参数设置以及查询报表等。系统主要流程及其主要模块关系请参照下图。

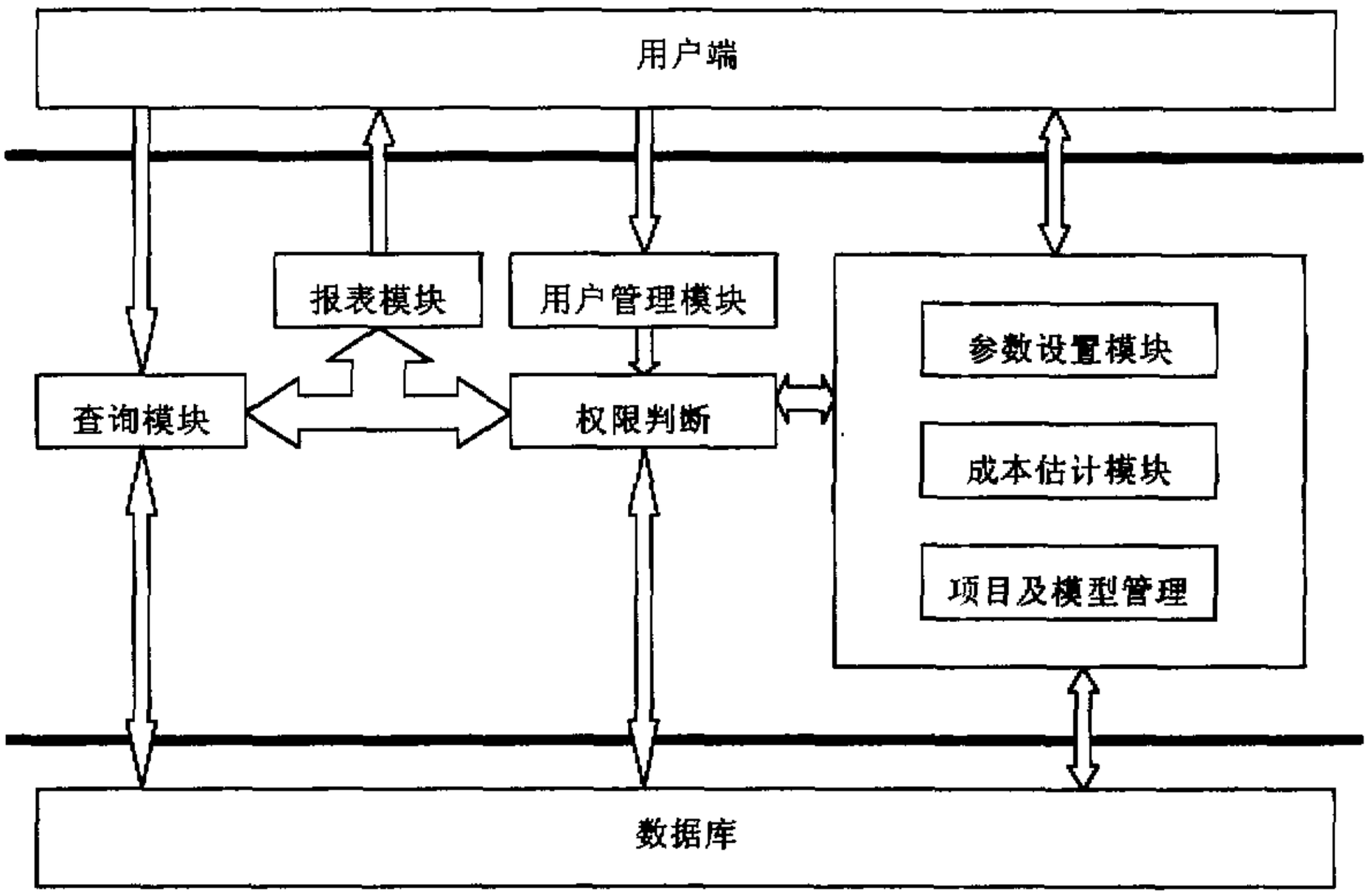

图 5.8　COCOMO II 系统主要模块关系图

Fig5.8 Relationship of modules in COCOMO II system

COCOMO II 系统的实际规程描述的是实际中使用 COCOMO II 系统来进行软件估计活动的步骤和方法。根据本规程可以通过使用 COCOMO II 系统来估计软件产品的规模、工作量、成本、关键计算机资源和进度。下面将简要地介绍本规程。

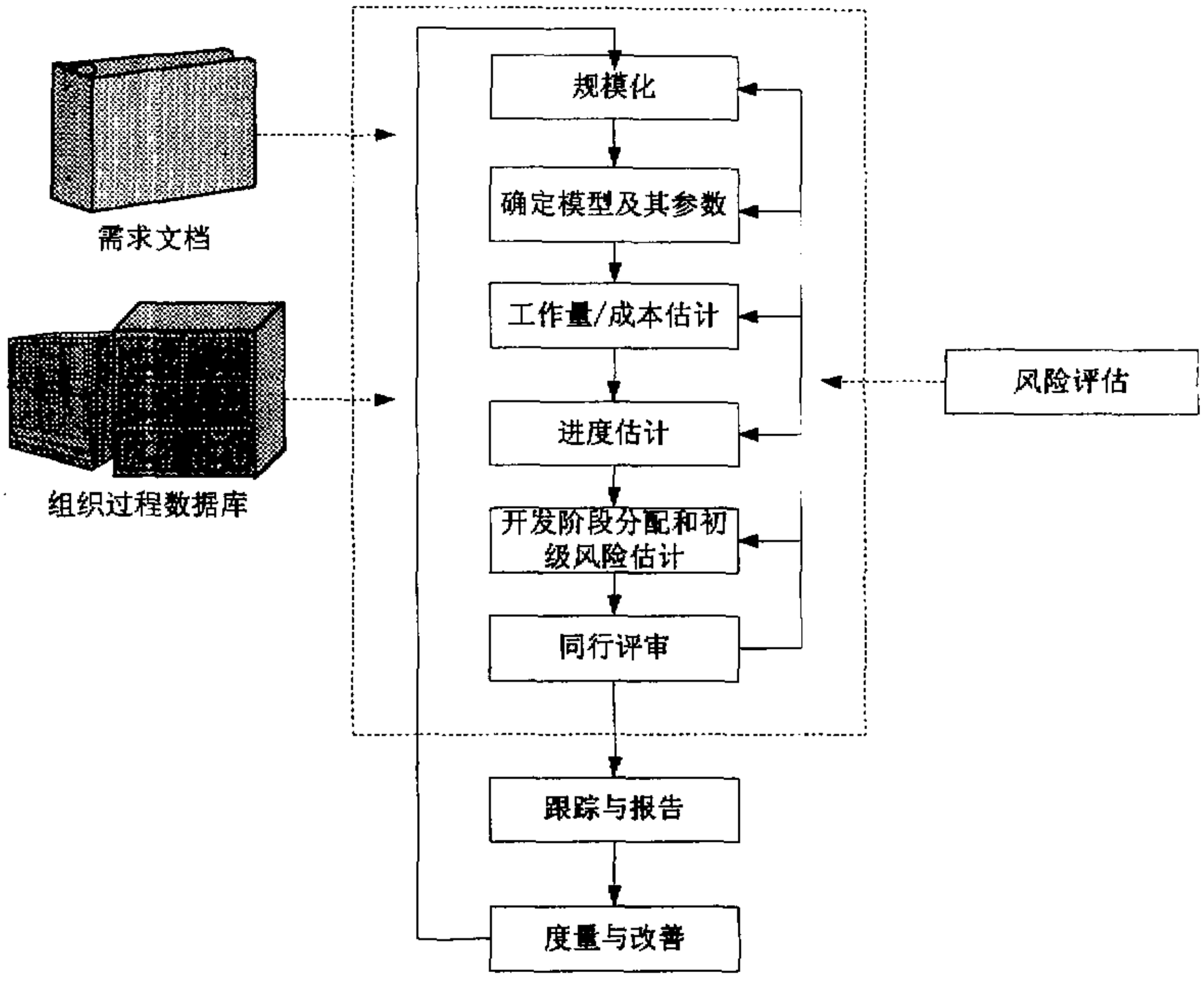

图 5.9 COCOMO II 系统估计规程图

Fig.5.9 The estimation process of COCOMO II system

1) 规模化

a) 利用功能点方法估计每个部件的新生成的代码规模（KSLOC）。

b) 对每个部件的复用代码的规模估计。

c) 对部件破损量的估计。

d) 求出每个部件的总共的规模。

e) 将所有的部件的规模累加，得到整个项目的规模。

2) 确定模型及其参数

a) 根据现在软件项目的特征，确定估计应使用的模型为早期设计模型还是后体系结构模型。

b) 依据对工作量乘法因数和规模因素的陈述，确定出本软件项目相应的模型参数数值。

3) 工作量和成本估计

a) 应用项目级驱动、规模因素和 SCED 成本驱动到合计规模，从而得到对于整个项目的全部的基本工作量。

b) 算出每个部件的基本工作量。

c) 应用部件级成本驱动（不包括 SCED）和相应的代码自动转换工作量到每个部件的基本工作量，计算出每个部件的工作量。

d) 计算出每个部件的工作量，从而累加得到合计的工作量。

4) 进度估计

a) 应用项目级驱动、规模因素到合计规模（不包括 SCED），从而得到对于整个项目的全部的基本工作量。

b) 算出每个部件的基本工作量。

c) 应用部件级成本驱动（不包括 SCED）每个部件的基本工作量，计算出每个部件的工作量。

d) 计算出每个部件的工作量，从而得到合计的工作量。

e) 套用进度公式，来估计进度。

5) 开发阶段分配和初级风险估计

a) 对工作量和进度分配到相应的开发阶段。这样便于制定项目开发计划。

b) 根据风险估计扩展模型，对工作量和进度的当前估计进行初级的风险估计。这样便于组织制定项目开发计划时，了解项目开发的初级风险度。

6) 评审

对于软件估计的评审需要由有类似系统经验的人来参加，主要评审以下内容：

a) 确认规模化、工作量和成本、进度估计所采用的方法和参数是恰当的。

b) 确定估计的每个步骤执行是正确的。

c) 确定软件估计结果是合理和正确的。

d) 确定估计的内容是完整的。

5.3.4 系统的功能说明

系统的完成功能的基本结构如下图：

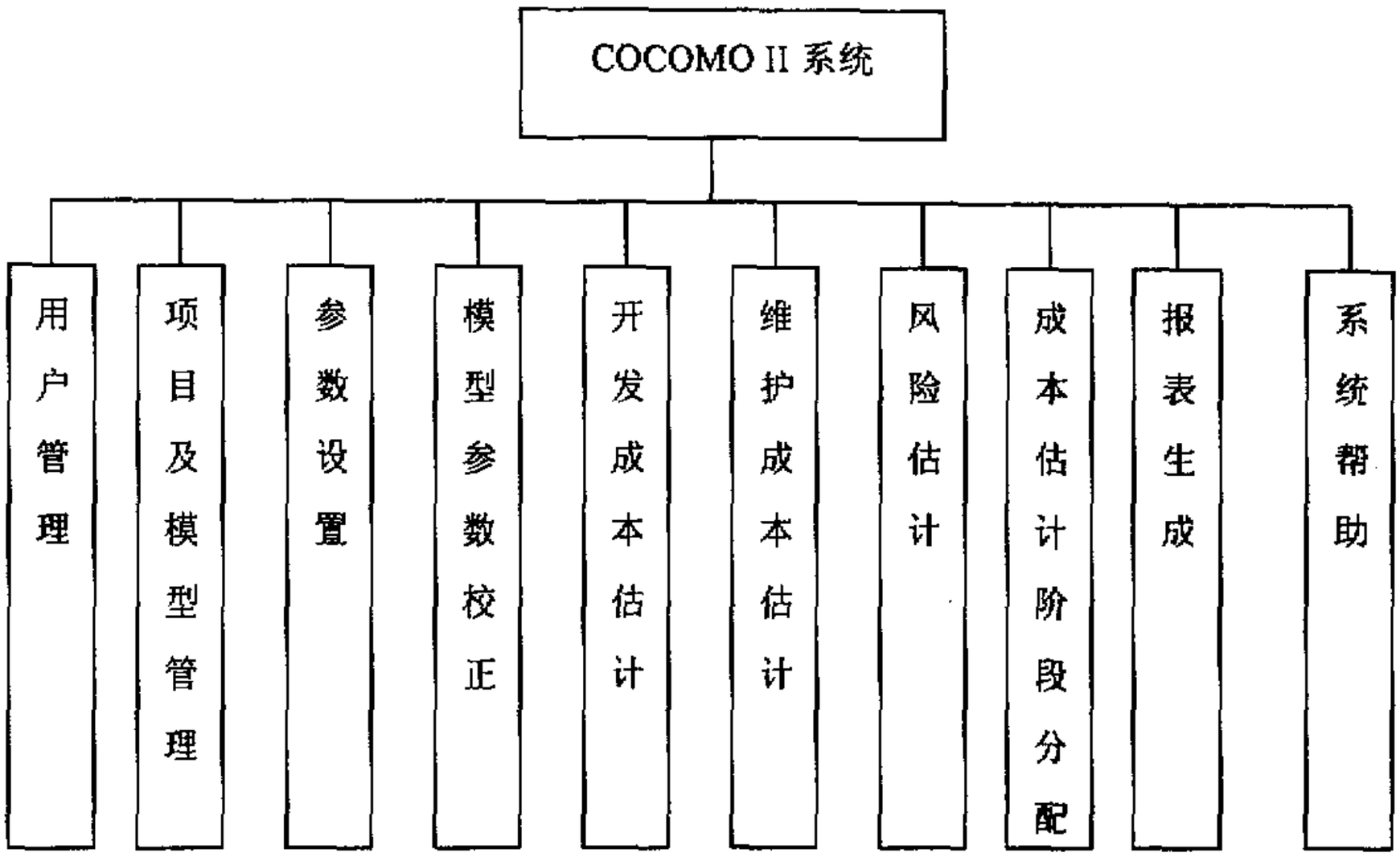

图 5.10 COCOMO II 系统基本结构图

Fig5.10 COCOMO II system architecture

COCOMO II 系统包括十个模块：用户管理、项目及模型管理、参数设置、模型参数校正、开发成本估计、维护成本估计、风险估计、成本估计阶段分配、报表生成和系统帮助。各个模块的功能简述如下：

1) 用户管理

主要完成对本系统使用用户的管理。这主要是对用户使用权限的控制。

2) 项目及模型管理

主要完成对本系统使用的项目信息和模型不同版本参数的管理。这主要包括项目和模型的信息的添加，修改和删除等处理。

3) 参数设置

主要完成对本系统使用模型初始参数和相关参数的设置。这主要包括规模因素和工作量乘法因数参数矩阵的设置，功能点参数的设置和公式的常数因子的设置等。

4) 模型参数校正

主要完成对本系统使用模型的参数校正。这包括利用后面章节介绍的校正方法对各个公式的参数的校正和相应的参数调整。

5) 开发成本估计

主要完成对项目开发阶段成本的估计。这包括对项目开发工作量和进度的估计。

这里着重对多模块以及增量式开发的估计进行一下说明。

首先讨论一下对多模块开发的估计方法步骤。通常软件系统是由多个子系统或部件组成的。COCOMO II 模型并不使用对每个部件估计求和的方法，因为这将忽略用于集成这些部件的工作量。这里的 COCOMO II 系统用于估计多模块的方法步骤如下：

a) 计算所有部件的规模 $Size_i$，从而求和，得到合计的规模数。

$$Size_{Aggregate} = \sum_{i=1}^{n} Size_i \tag{5.1}$$

b) 应用项目级驱动、规模因素和 SCED 成本驱动到合计规模，从而得到对于整个项目的全部的基本工作量。

$$PM_{Basic} = A * \left(Size_{Aggregate}\right)^E * SCED \tag{5.2}$$

c) 算出每个部件的基本工作量。

$$PM_{Basic(i)} = PM_{Basic} * \left(\frac{Size_i}{Size_{Aggregate}}\right) \tag{5.3}$$

d) 应用部件级成本驱动（不包括 SCED）和相应的代码自动转换工作量到每个部件的基本工作量，计算出每个部件的工作量。

$$PM_i = PM_{Basic(i)} * \sum_{j=1}^{16} EM_j + PM_{AUTO(i)} \tag{5.4}$$

e) 计算出每个部件的工作量，从而得到合计的工作量。

$$PM_{Aggregate} = \sum_{i=1}^{n} PM_i \tag{5.5}$$

进度的估计也可利用如上的步骤2~5，但在第2步，不包含SCED成本驱动，以及在第 4 步，不包含 $PM_{AUTO(i)}$。从而得到无 SCED 和 $PM_{AUTO(i)}$ 的合计工作量 $PM'_{Aggregate}$，套用进度公式，来估计进度。

下面讨论对增量式开发的估计方法步骤。增量式软件开发作为可及早地提交部分的软件项目的方法，而被软件开发组织广泛地接受，因而对这样软件成本的估计有一定的必要性，但同时由于其开发阶段进度的灵活性，这样对于软件成本估计也有一定的难度。对增量式开发的估计步骤如

下：

a) 计算每次增量的规模和总共的规模。

一个增量的规模是由新代码和对以前代码的增量代码所组成的。

$$Adj.Size_i = Size_i + \left(AAM * \sum_{j=1}^{i-1} Size_j \right) \tag{5.6}$$

$$AAM = \begin{cases} \dfrac{AA + AAF * (1 + [0.02 * SU * UNFM])}{100}, \text{当} AAF \le 50\text{时} \\ \dfrac{AA + AAF + (SU * UNFM)}{100}, \text{当} AAF > 50\text{时} \end{cases}$$

$$AAF = 0.4 * DM + 0.3 * CM + 0.3 * IM$$

b) 使用 COCOMO II 计算出对于整个单个产品的整个工作量和进度。

c) 使用 COCOMO II 计算出增量的估计的工作量和进度。

d) 计算出增加的集成工作量。

$$PM_{integration} = PM_{total} - \sum PM_{eachIncrement}$$
$$PD_{integration} = PD_{total} - \sum PD_{eachIncrement} \tag{5.7}$$

e) 在剩余的增量之间分配增加的集成工作量。

$$Size_{Total\,Remaining} = \sum Size_{eachIncrement}$$
$$PM_{IncrementAddedIntegration} = \frac{Size_{eachIncrement}}{Size_{total\,Remaining}} * PM_{AddedIntegration} \tag{5.8}$$

f) 计算所有的剩余增量的工作量和进度。

$$PM_{IncrementAdjusted} = PM_{Increment} + PM_{IncrementAddedIntegration} \tag{5.9}$$

g) 在阶段之间分配每个增量的工作量和进度。

h) 计算每个增量的结果累积的工作量和进度。

6) 维护成本估计

主要完成对项目维护阶段成本的估计。这包括对项目维护工作量和进度的估计。

7) 风险估计

主要完成对项目开发风险的估计。这里使用上面介绍的 COCOMO II 扩展模型中的风险评估模型的方法来给出一些简要的初步风险估计信息。

8) 成本估计阶段分配

主要完成对项目成本估计结果在开发的不同阶段上进行初步的分配。这包括对项目估计结果的阶段和活动初步的分配。这里使用简单的工作拆分结构的工作量和进度在各个阶段所占百分比的方法来分配成本估计结

果。

9) 报表生成

主要完成对项目估计结果的报表的生成。

10) 系统帮助

主要完成对使用本系统的用户提供相应的如何使用 COCOMO II 系统的帮助信息。

5.4 系统对模型参数的校正改善

5.4.1 裁剪 COCOMO II 模型到特定的组织

用自己组织的数据来校正的 COCOMO II 将产生更好的估计的结果。主要的好处如下：

- 校正模型到现存的项目数据。

  自己组织的校正通常会提高预言的准确性，因为

  （1）COCOMO II 的等级规模的确定是主观的，这导致了在不同组织间的不一致性。

  （2）不同组织间，COCOMO II 覆盖的生命周期活动可能有稍微的不同。

  （3）不同组织间，用 COCOMO II 来做的定义有稍微的不同。
- 巩固（consolidating）或消除多余的参数。
- 添加模型中未显式表示的重要的成本驱动，即在特定的项目中，加入用户特定的参数。

通常我们只利用校正方法来校正模型公式的乘法系数 A 和基指数 B，这对于一个项目组织就已经足够了。而且我们建议对于校正乘法系数 A 至少需要 5 个数据点，对于校正乘法系数 A 和基指数 B 至少需要 10 个数据点。

5.4.2 校正方法

5.4.2.1 DELPHI 方法

这里对一般的 DELPHI 方法进行了改进，改进的 DELPHI 方法如下：

1) 协调者提供 DELPHI 调查表给每个调查的参与者。
2) 协调者和参与者举行一个会来讨论相关的问题。
3) 参与者使用单个生产率（PR）数值（即最高和最低等级对应数值的比值，这个值将用于确定各个因素的新对应的数值）来简化过程。他们先得到一个初始的范围作为出发点。参与者定义出与更适合他们经验的每个 PR 关联的差异。参与者完成 DELPHI 调查表格，并将它返还给协调者。
4) 协调者反馈参与者反映的情况。
5) 协调者举行另一个会议来讨论参与者反映的差异，达到一个可能的一致的结果。
6) 协调者请求参与者再次估计，再次取得一致，重复 4 至 6 步，直到达到满意的结果。

这里将讲述实际的 PR 的使用。PR 的求法如下：

$$\text{（对于规模因素）}\ PR_{SF_n} = \frac{(100)^{0.91+(0.01*SF_{n_{MAX}})}}{(100)^{0.91}} \tag{5.10}$$

$$\text{（对于工作量乘法因数）}\ PR_{EM_n} = \frac{EM_{n_{MAX}}}{EM_{n_{MIN}}} \tag{5.11}$$

在已知确定 PR 值的条件下，各等级系数的初步求解方法如下：

1) 求工作量乘法因数各等级对应的系数，假设中间值为额定等级系数值为 1，而且每个等级系数间的差距相等，左右对称，这样可求出等级系数最小端（与额定等级相差 n 个等级差）和最大端（与额定等级相差 m 个等级差）的等级系数，假设 DELPHI 得出的 PR 值为 2，则利用公式 $\frac{1+mx}{1-nx} = PR = 2$，求出 x 值，此值即为每个等级差所对应的系数大小，这样通过额定等级系数为 1，和等级差所对应的系数大小，便可以求出每个等级所对应的系数了。
2) 求规模因素各等级对应的系数，假设最小值为特别高等级系数，值为 0，而且每个等级系数间的差距相等，这样可求出最大端（与特别高等

级相差 m 个等级差）的等级系数，假设 DELPHI 得出的 PR 值为 2，则利用公式 $PR=\frac{(100)^{0.91+(0.01*x)}}{(100)^{0.91}}=2$，求出 x 值，此值即为每个等级差所对应的系数大小，这样通过最小值为特别高等级系数，值为 0，和等级差所对应的系数大小，便可以求出每个等级所对应的系数了。

这里得到的初步的系数可利用其它校正，进一步提高其精度。

5.4.2.2 多元线性回归方法

我们以工作量估计模型公式参数的校正为例来说明用多元线性回归方法来进行校正的过程，进度参数的校正同样可使用这样的校正方法。

初始的 COCOMO II 的后体系结构模型的形式如下：

$$Effort=A*[Size]^{B+\sum_{i=1}^{5}SF_i}*\prod_{i=1}^{17}EM_i \tag{5.12}$$

这里，A 是乘法因数常量；B 是规模的基指数；SIZE 是以源代码行或功能点或对象点为单位来度量的软件项目的规模；SF_i 是规模因素；EM_i 是工作量乘法因数。

上面的公式可线性化如下：

$$\begin{aligned}&\ln(PM)=\beta_0+\beta_1 B\ln(Size)+\beta_2 SF_1\ln(Size)\\&+\cdots+\beta_6 SF_5\ln(Size)+\beta_7\ln(EM_1)+\beta_8\ln(EM_2)+\cdots+\beta_{23}\ln(EM_{17})\end{aligned} \tag{5.13}$$

这个公式有着多元线性回归模型的形式。但是，通常回归结果往往生成几个不合理的参数估计。这些负面的作用是由于主观和回归参数的不合理而产生的。因为其中的几个参数是相互关联的、数据不足够多以及可能有较极端的例子数据，这些都违反了回归分析的条件。我们可以将关联的参数合并的方法来消除关联，从而进行更好的估计。例如将 TIME 和 STOR 合并为 RCON（资源限制），将 ACAP 和 PCAP 合并为 PERS（人员因素）。而对于某些错误估计的参数（如系数为负数），我们可以利用实际的数据加以调整。例如使用专家判定的方法。

利用上面的公式，我们用得来的参数 $\beta_0,\beta_1\ldots\beta_{23}$ 作为乘法因数或指数来校正规模因素 SF_i 和工作量驱动 EM_i。

我们开发了一个实际有效的简化过程来估计模型的参数。它使用了带有权重的基于专家结果的方法，即结果中 10%为回归分析估计的值，90%为专家判断估计的值。这很好地调整了参数的估计。

5.4.2.3 BAYESIAN 方法

虽然用于混合专家判断和实例数据的 10%的加权过程提供了一个可运行的

初始的模型，我们仍要开发一个正式框架来校正参数。它能显式地确认关于个人工作量乘法因数，以及规模因素的信息进行适当的改变。带有先验信息的 BAYESIAN 分析就是这样的框架。

BAYESIAN 分析就是一种诱导式推理的模型，它可以以逻辑一致的方式来综合实验（数据）和先验值（专家判断）信息。使用 BAYESIAN 理论，先验值转化成模型参数的后验值。这个转化被视为一个学习的过程。权重的分配取决于先验值和实验数值的变化大小。如果先验值的变化小于实验数值的变化，则将分配给先验值的权重提高，反之，则提高实验数值的权重。使用 BAYESIAN 理论，我们能综合我们的两个信息源，如下：

$$f(\beta|Y)=\frac{f(Y|\beta)f(\beta)}{f(Y)} \tag{5.14}$$

这里 β 是我们感兴趣的参数向量，Y 是从关联密度函数 $f(\beta|Y)$ 得来的实验观察向量。$f(\beta|Y)$ 是关于 β 的后验证密度函数，它总结了所有的关于 β 的信息。$f(Y|\beta)$ 是实验信息，在数值上与关于 β 的可能性函数是相当的。$f(\beta)$ 是先验信息，它总结了专家关于 β 的信息。上面的公式可以表述如下：

$$f(\beta|Y)\propto l(\beta|Y)f(\beta) \tag{5.15}$$

换句话说，这个公式意味着：

$$Posterior \propto Sample \times Prior \tag{5.16}$$

BAYESIAN 方法提供了两个独立源信息的优化综合。后验系数 b^{**} 和变量 Var(b^{**})定义如下：

$$\begin{aligned} b^{**}&=\left[\frac{1}{\sigma^2}XX+H^*\right]^{-1}\times\left[\frac{1}{\sigma^2}XXb+H^*b^*\right] \\ Var(b^{**})&=\left[\frac{1}{\sigma^2}XX+H^*\right]^{-1} \end{aligned} \tag{5.17}$$

这里 X 是实例变量的矩阵，b 是由实例数据利用最小二乘法得到的系数，δ^2 是实例数据分析生成过程的方差；H^* 和 b^* 分别是先验信息的精度（方差的倒数）和系数。

公式(5.17)表示了随着相对于实例信息精度（$\frac{1}{\delta^2}X'X$）（或变化）的先验信息的精度提高（或先验信息变化降低），后验参数更加接近先验值，最后到达一定精度时，即为结果。

通常，先验值的协方差被假设为零。这个假设可以简化原型模型。实际上，对于完整的 COCOMO II 模型，这个假设常常并不被这么做。相反的，先验值

的协方差被设定等于实例数据的协方差，或者使用象如下的 G-PRIOR 这样的技术来处理先验值的协方差。

5.4.2.4 改进的 **BAYESIAN** 方法

虽然存在大量的关于模型参数校正改进的方法，我们最终选择了 Sellner 的 G-PRIOR 方法，因为它不要求困难地为每个 β 的分量指定先验协方差的任务。G-PRIOR 方法假设 β 的先验协方差等于实例数据提供的协方差。换句话说，先验精度矩阵表示如下：

$$H^* = \frac{g}{\sigma^2} XX \tag{5.18}$$

因此，使用 G-PRIOR 方法的 β 的后验估计为：

$$b^{**} = \frac{b + gb^*}{1+g} \tag{5.19}$$

$$b = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix} \quad , \quad b^* = \begin{pmatrix} \ln A \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

这里 b 是由实例数据利用最小二乘法得到的系数向量，b^*是专家期望值的系数向量。g 的量级与分给先验值的精度相对应。例如：当 g=0.1 时，先验值 b^*收到 1/9 的权重，而实验值 b 收到 8/9 的权重。这个版本的 G-PRIOR 便可以假设同样的权重分给每个参数，即 g=0.5。

另外，考虑到个人工作量乘法因数和规模因素的信息变化，我们现在用扩展 Sellner 的 G-PRIOR 方法来分配不同权重给参数。我们知道，对我们数据集的反映不可能产生一致理解的每个工作量乘法因数和规模因素。因此我们要逻辑地分配不同的权重给一些参数。例如，对于在 RUSE 上收集的数据，一些回应为“NOMINAL”，其实这是他们的实际回应是“I DO NOT KNOW”。其他的我们遇到困惑的因素包括 AEXP，LTEX，FLEX 和 TEAM。结果呢，这五个因素的先验估计被给的权重要高于分配给其他因素（更多的精确的数据可提供）的先验知识的权重。

我们得出了使用扩展 G-PRIOR 方法的 β 的后验估计为：

$$b^{**} = \left[gZX'XZ + X'X\right]^{-1} \times \left[gZX'XZb + X'Xb^*\right] \tag{5.20}$$

$$Z=\begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_k \end{pmatrix}$$

这里 Z 是分析向量，它表示对于工作量乘法因数和规模因素的不同的权重。当 Z 是单元向量时，即，z_i=1，我们就使用如上的通常 G-PRIOR 方法实例。

我们实验了 g 和 z_i的好多种组合，最后我们发现最好的结果是：

对于 G-PRIOR 方法：g=0.1。

对于扩展的 G-PRIOR 方法：g=0.1；对于 AEXP、LTEX、FLEX、TEAM 和 RUSE，$z_i=\sqrt{5}$；而对于其它的成本因素，z_i=1。

我们用得来的参数 $\beta_0,\beta_1...\beta_{23}$ 作为乘法因数或指数来校正规模因素 SF_i 和工作量驱动 EM_i。

5.4.3 校正的验证

通常利用软件成本估计数据的分析来得到模型参数的方法是一个很好的方法。不幸的是，这样的模型往往显示了较差的对于新数据的预见能力。因此我们要对这样的模型进行交叉验证。我们可以利用实例数据中的部分新数据来验证我们的模型。

相对错误等级 MRE（magnitude of relative errors）：

$$MRE=\left|\frac{Estimated-Actual}{Actual}\right| \tag{5.21}$$

对于上面的几个模型参数校正方法，我们得到了如下的验证结果。它表示了在 30%的 MRE 内，各个校正方法的估计预言的准确率。

表 5.1 校正方法对比表

Table5.1 Calibration methods contrast

	一般	最小	最大
最小二乘法 OLS	64%	54%	76%
G-PRIOR	70%	57%	79%
扩展 G-PRIOR	76%	62%	84%

因此，为了有效地提高校正后模型对软件成本估计的准确率，COCOMO II 系统对模型参数的校正方法主要采用了扩展 G-PRIOR 的 BAYESIAN 方法。

5.5 系统估计结果及其分析举例

下面我将举个例子来简要地验证和分析一下 COCOMO II 系统的软件成本估计的可行性和准确性。这个例子是利用 COCOMO II 对 KiwiDiscTool 工具套件进行软件成本估计，并对估计结果进行了分析。

KiwiDiscTool 是一个用来检验 Kiwi 数据盘正确与否的工具套件，它包括 FocusEye、BinaryEye 和 StatisticEye 三个工具。FocusEye 是通过地图描画，地图选择等操作来验证数据盘的正确性，BinaryEye 是通过数据的二进制显示来检验，StatisticEye 是通过数据统计来检验数据盘中保存的所有数据。

利用上面介绍的功能点规模化方法和成本调整因素取值方法，COCOMO II 成本估计系统估计出这个软件项目共包含 316 个未调整的功能点，因为此软件主要由 C 语言开发，查表得到一个功能点可转化为 128 代码行，所以估计出该软件项目包含 40448 行代码行（实际为 39400 代码行），即 40.448KSLOC，将它和表 5.2 中的调整因素值代入后体系结构模型公式，计算得出工作量。

表 5.2 COCOMOII 估计成本调整因素值

Table 5.2 Adjusted factors of cost estimation in COCOCMO II

因素	取值	等级
RELY	0.82	非常低
DATA	1.00	额定
CPLX	0.87	低
RUSE	1.15	非常高
DOCU	0.81	非常低
TIME	1.00	额定
STOR	1.00	额定
PVOL	0.87	额定
ACAP	0.85	高
PCAP	0.76	非常高
APEX	0.88	高
PLEX	0.91	高
LTEX	0.91	高
PCON	1.29	非常低
TOOL	0.90	高
SITE	0.80	特别高
SCED	1.00	额定
PREC	1.24	非常高
FLEX	3.04	额定
RESL	4.24	额定
TEAM	1.10	非常高
PMAT	3.12	高

表 5.3 是 COCOMO II 对 KiwiDiscTool 的工作量估计结果对比表。

表 5.3 工作量估计结果对比表（PM）

Table 5.3 Effort estimation results contrast（PM）

	代码行估计的结果	用功能点分析的结果		实际结果
工作量	28.2	乐观	29.7	32
		正常	37.1	
		悲观	46.4	

从估计结果与实际结果的对比显示，估计与实际结果基本上是相同的，存在的差异可能主要是因为该工具主要是由 C 开发的，但有一部分的功能是用 VB 开发的，而在进行 COCOMO II 估计时选择的开发语言为 C 语言。而且分析结果得出 COCOMO II 和功能点分析有它的优越之处，它不需要更多的技术，只要从用户的角度考虑就可以了。

经过对大量软件项目的估计结果的分析，我们得出结论：COCOMO II 系统有较好的估计可行性和准确性，应该应用到对软件项目的成本估计上。

5.6 总结和展望

本文中对 COCOMO II 系统的分析设计和校正改善的目的是为了更好对软件项目的开发提供更加有效的可信的定量的决策和管理信息，从而使项目的开发成熟度到达一定的水平。因为现在只对此软件成本估计系统进行了总体分析和初步实现，所以当前主要的工作应放在对 COCOMO II 系统实现的完善和加强上，另外也应注意 COCOMO II 系统对以往项目的分析总结，建立完善的软件财富库，为将来的项目决策和管理提供有效合理的信息，从而使软件能力成熟度稳步提高。该 COCOMO II 系统的充实和提高也必须在可靠理论的指导下才能顺利地进行。而且软件成本估计的研究和实现是不断探索、总结和再实践的过程。本系统在公司实际的软件项目的成本估计中的应用取得了良好的效果。但它所使用的方法和技术仍需要进一步的完善和提高。随着软件成本估计知识和相应信息的不断积累，COCOMO II 系统将逐步利用收集的软件项目数据来验证估计的准确性和校正模型的参数，从而完善模型的估计能力，并细化和数量化模型公式中调整因素取值规则，同时我们还要加强对 COCOMO II 扩展模型的研究，早日实现其它的扩展模型功能。这样，COCOMO II 系统便可以为软件的策划、开发和决策等方面提供更多有用的信息。

第六章 结束语

本文针对如何搞好软件成本估计的 COCOMO II 模型提出了一些看法和观点。这里也提出了软件成本估计的实现是提高软件能力成熟度等级的必要条件。本文还通过 COCOMO II 系统的分析设计和具体实践论证了如何更好地利用 COCOMO II 估计模型进行软件成本估计。在对 COCOMO II 的具体实践过程中，我结合本公司的 CMM 和项目的实际情况，提出了一个实用的 COCOMO II 模型分析设计，并对 COCOMO II 模型进行了调整因素取值规则的细化，编写了使用 COCOMO II 来进行成本估计的规范，而且还对模型参数的校正方法进行了改进，并对几个校正方法进行了对比。

软件成本估计和 COCOMO II 模型的研究和实践，以及其系统分析和部分实现都是我在做毕业论文实习期间所负责的工作。在此期间，我还对 CMM，及其相关理论进行较深入的学习、研究和实践，从而加深了对软件工程和软件开发及其软件能力成熟度的认识。

由于时间、本人能力和水平的限制，本论文难免有笔误和错漏之处，希望大家多提意见，多加指正。

参考文献

1. T. Capers Jones. Estimating Software Costs[M]，New York：McGraw-Hill，1998，3-66
2. 张金奎. 线性模型参数估计及其改进[M]，长沙：国防科技大学出版社，1992， 7-65
3. 罗圣仪. 计算机软件质量保证的方法和实践[M]，北京：科学出版社，1999，95-138
4. 王式安. 数理统计方法及应用模型[M]，北京：北京科学技术出版社，1992，254-270
5. 中国软件行业协会上海分会. 软件质量及其评价技术[M]，北京：清华大学出版社，1990，58-76
6. 王永葆. 应用数理统计[M]，沈阳：东北大学出版社，1995，132-177
7. 薛万鹏. C++程序设计教程[M]，北京：机械工业出版社，2000，1-223
8. Barry W. Boehm. Software Cost Estimation with COCOMO II[M] ， New Jersey：Prentice Hall PTR，2000，1-493
9. 宛延凯. C++语言和面向对象程序设计[M]，北京：清华大学出版社，1999，1-238
10. 侯雪萍. C++深入浅出[M]，北京：学苑出版社，1994，488-501
11. 周升锋. Visual C++ Windows 实用编程技术[M]，北京：北京航空航天大学出版社，1996，1-383
12. 谭浩强. C 程序设计[M]，北京：清华大学出版社，1996，268-291
13. Sunita Chulani，Barry Boehm，Bert Steece. Calibrating Software Cost Models Using Bayesian Analysis，USC-CSE，1998
14. Barry Boehm. Cost Models for Future Software Life Cycle Process: COCOMO 2.0，USC-CSE，1995
15. Kemerer C. An Empirical Validation of Software Cost Estimation Models[J]，Communications of the ACM，1997(5)，416-429
16. Paulk M.，B. Curtis. Capability Maturity Model for Software, Version 1.1，Software Engineering Institute ，CMU-SEI-93-TR-24，1993
17. USC Center for Software Engineering. COCOMO II Model Definition

Manual，http://sunset.usc.edu/cocomo.html，1997

18. Sunita Devnani-Chulani. Calibrating the COCOMO II Post-Architecture Model，USC Center for Software Engineering，1997
19. Sunita Devnani-Chulani. Calibration Approach and Results of the COCOMO II Post-Architecture Model，USC Center for Software Engineering，1997
20. USC Center for Software Engineering. COCOMO II Cost Estimation Questionaire , Computer Science Department , http://sunset.usc.edu/cocomo.html，1997
21. Sunita Devnani-Chulani. A Bayesian Software Estimating Model Using a Generalized g-Prior Approach，USC Center for Software Engineering，1998
22. Barry Boehm , Chris Abts. Software Development Cost Estimation Approaches-A Survey，USC-CSE，1998
23. James O. Berger. Statistical Decision Thoery: Foundations, Concepts, and Mothods[M]，New York：Springer-Verlag，1980，89-169
24. Sir Ronald A. Fisher. Statistical Mothods and Scientific Inference[M]，London：Oliver and Boyd，1959，37-177
25. 周之英. 现代软件工程（中）：基本方法篇[M]，北京：科学出版社，2000，174-436
26. 杨一平. 软件能力成熟度模型 CMM 方法及其应用[M]，北京：人民邮电出版社，2001，18-273
27. Mark C. Paulk，C. V. Weber，S. Garcia，M. B. Chrissis，and M. Bush. Key Practices of the Capability Maturity Model(Ver. 1.1)，Software Engineering Institute，CMU/SEI-93-TR-25， 1993
28. Clark, B. The Effects of Process Maturity On Software Development Effort[D]，USA：University of Southern Califarnia，1997

致 谢

本论文是在我的导师程秋木老师的精心指导下完成的。程老师有着渊博的学识，严谨的治学态度，对科学技术发展趋势准确的预见能力和诲人不倦的学者风范。从课题的立项、方案设计到具体实现每一步都倾注了程老师大量的心血，本人有幸得到程老师的亲切关怀和高水平指导，在此谨向程老师表示衷心的感谢。

在实习和论文撰写的过程中，我得到何晓源、刘栋臣等老师的悉心指导和帮助。他们对本论文的组织形式、写作方式以及材料的选择等方面都提出了宝贵的意见，在此对他们表示深深的谢意。

本论文是在东软集团的东软股份公司完成的。公司的领导为本人创造了良好的条件，给予了热情的支持，部门各位同事也为我提供了许多帮助，使我能够顺利地完成课题的研究设计。在此谨向公司的领导和同事们表示衷心的感谢。

我还要感谢同我一起工作的同事、同学，他们为我的论文工作提供了很多的帮助，同时还有很多的朋友为我的论文工作提供了帮助，他们的名字不能一一尽述，在此一并表示感谢。

同时，我还要感谢我的家人，他们在我的学业中给予了我极大的生活上和精神上的关怀，使我顺利完成学业。

最后，再一次向所有帮助和关心我的各位老师、同学、同事和家人表示深深的谢意。