贝叶斯校正算法在软件估算模型 COCOMO Ⅱ 中的应用

丁岳伟　马亦舟
（上海理工大学计算机工程学院　上海 200093）

摘　要　软件估算模型 COCOMO Ⅱ 的研究目标在于建立起一个能够适用于当前的软件开发并且每年更新的模型，使之能够估算软件开发活动中的成本、工作量及时间安排。为了提高 COCOMO Ⅱ 的估算精度，采用归纳性的分析方法——贝叶斯校正算法对其进行校正，在逻辑一致的基础上根据先验信息和样本信息作出推论，得到的后验结果能够较大幅度提高估算精度，是对 COCOMO Ⅱ 模型的进一步完善。

关键词　软件成本估算　COCOMO II　贝叶斯校正算法　多元回归算法

IMPLEMENTATION OF BAYESIAN CALIBRATION IN THE SOFTWARE ESTIMATE MODEL COCOMO Ⅱ

Ding Yuewei　Ma Yizhou
(*College of Computing Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China*)

Abstract　The COCOMO Ⅱ research effort is concentrated on developing a model well-suited for the contemporary software development and then annually updating it for the forthcoming years, which allows one to estimate the cost, effort, and schedule when planning a new software development activity. In order to improve the estimate accuracy of COCOMO Ⅱ, Bayesian calibration, one generalized analysis method, is adopted to calibrate the model. It arrives, on the basis of both prior information and sample information in a logically consistent manner, at the posterior result which can makes the estimate more accurate in big degree and definitely improves the COCOMO Ⅱ Model.

Keywords　Software estimation　COCOMO Ⅱ　Bayesian calibration　Multiple regression

1　引　言

在过去的几十年间，出现了许多模型用于软件生命周期早期阶段的软件成本和进度估算。这些评估方法的目标就是为了保证在预算和质量要求的基础上，通过软件开发过程管理，及时交付软件产品。在少数几种公开发表的模型中，Barry Boehm 教授在其著作《软件工程经济学》中发表的早期的 COCOMO 模型被广泛地接受和使用。

1990 年后，软件项目管理和开发技术与工具发生了很大的变化，出现了快速应用开发模型、软件重用、再工程、CASE、面向对象方法以及软件过程成熟度模型等一系列软件工程方法和技术。早期的 COCOMO 模型已经不再适应新的软件成本估算和过程管理的需要，因此，Barry Boehm 教授根据未来软件市场的发展趋势，重新研究和调整原有模型，于 1994 年发表了 COCOMOⅡ。

2　COCOMO Ⅱ 模型原理与算法[1]

2.1　COCOMO Ⅱ 模型简介

COCOMO Ⅱ 包含三个子模型：应用构图模型、早期设计模型、后体系结构模型。后体系结构模型包含了软件产品的实际开发和维护过程。该模型在软件生命周期的进程中能更好地进行成本评估，可以验证系统的任务，运作概念和风险，并可从中建立产品的框架。它使用源代码行数（SLOC）和功能点作为项目大小的输入，并使用修正因子来表示重用和软件的“破损率”。该模型具有 17 个倍乘成本驱动因子和 5 个比例指数因子。

由于后体系结构模型是目前被接受和应用最广泛的模型，下文将围绕其展开讨论。以下凡提到 COCOMO Ⅱ 模型处，如无注明，均特指后体系结构子模型。

2.2　COCOMO Ⅱ 算法

在 COCOMO Ⅱ 模型中，以人月（PM）作为工作量的计量单位。工作量计算公式为：

$$Effort = A \times [Size]^{1.01+\sum_{i=1}^{5} SF_i} \times \left(\prod_{i=1}^{17} EM_i\right) \qquad (1)$$

其中，A——倍乘常数；

$Size$——软件项目大小，用千代码行（KSLOC）来计量；

SF_i（$i = 1, 2, 3, 4, 5$）代表了 5 个比例指数因子，分别为：PREC（Precendentedness，有先例性）；FLEX（Development Flexibility，开发灵活性）；RESL（Architecture and Risk Resolution，结构风险解决度）；TEAM（Team cohesion，团队合作度）；PMAT（Process Maturity，过程成熟度）。

EM_i（$i = 1, 2, \cdots, 17$）代表 17 个工作量乘数，而这 17 个 EM 又可归为 4 个因素集合，分别为：

收稿日期：2005 - 02 - 22。上海市教委发展基金资助项目（03GK16）。丁岳伟，教授，主研领域：网络应用与信息安全，软件工程与 CMM。

• **产品因素集合** RELY(Required Software Reliability,所需的软件可靠性);DATA(Data Base Size,数据库大小);CPLX(Product Complexity,产品复杂度);RUSE(Requried Reusability,所需的重用性);DOCU(Documentation match to life-cycle needs,生命周期需求匹配文档)。

• **平台因素集合** TIME(Execution Time Constraint,执行时间的限制);STOR(Main Storage Constraint,主存限制);PVOL(Platform Volatility,平台易变度)。

• **人员因素集合** ACAP(Analyst Capability,分析员能力);PCAP(Programmer Capability,程序员的能力);AEXP(Applications Experience,应用经验);PEXP(Platform Experience,平台使用经验);LTEX(Language and Tool Experience,编程语言和编程工具使用经验);PCON(Personnel Continuity,团队一致性)。

• **项目因素集合** TOOL(Use of Software Tools,软件工具的使用);SITE(Multi-Site Development,多点开发);SCED(Requried Development Schedule,所需的开发进度)。

每个 SF 和 EM 包含不同级别,从很低到特高,每个级别都有对应的衡量标准和取值。

值得注意的是,自 Barry Boehm 教授于 1994 年提出 COCOMO Ⅱ模型以来,5 个比例因子(SF)和 17 个工作量乘数(EM)的各级别的衡量标准和取值随着软件开发技术的进步一直不断做着修正[2]。如何实现高效的修正,来适应各个用户的具体情况,贝叶斯校正算法提供了一个可行的方法[3]。

3 贝叶斯校正算法

3.1 贝叶斯校正算法原理[4,5]

贝叶斯算法提供了一个标准过程,将专家决定作为先验信息与样本信息(数据)相结合,建立起一个可靠的后验模型:

$$f(\beta|Y)=\frac{f(Y|\beta)f(\beta)}{f(Y)} \tag{2}$$

其中 β 是我们所关心的参数向量,Y 是联合密度函数 $f(Y|\beta)$ 的样本观测向量 $f(\beta|Y)$ 是 β 的后验密度函数,其中综合了有关 β 的所有信息 $f(Y|\beta)$ 表示样本信息,在代数上等于 β 的可能性函数 $f(\beta)$ 是综合了关于 β 的专家决定信息的样本信息。后验分布均值 b^{**} 和方差 $Var(b^{**})$ 定义为:

$$b^{**}=\left[\frac{1}{s^2}X'X+H^*\right]^{-1}\times\left[\frac{1}{s^2}X'Xb+H^*b^*\right] \tag{3}$$

$$Var(b^{**})=\left[\frac{1}{s^2}X'X+H^*\right]^{-1} \tag{4}$$

其中,X——自变量矩阵;

s^2、b——样本数据的期望值和方差;

H^*、b^*——先验信息的精度(方差的倒数)和均值。

从式(3)、(4)可以看到为了确定贝叶斯后验均值和方差,我们首先需要确定先验信息和样本信息的均值和精度。

3.2 先验信息

为确定计算所用到的先验信息,贝叶斯算法的倡导者进行了一个 DELPHI 实验。在第一轮实验中,八位软件估算领域的专家各自单独给出 COCOMO Ⅱ模型的各项成本驱动因子(SF、EM)的取值。实验组织者将第一轮结果归纳总结后,送回给各位专家,进行第二轮的实验。让他们在结合其他专家的结论的基础上独立修改之前作出的判断,以取得更一致的结果。COCOMO Ⅱ模型的 22 项成本驱动因子的先验均值(b^*)和方差(H^*)就由第二轮实验的结果计算得到。表 1 展示了一个工作量乘数因子 RELY 的先验值集合。

表 1 RELY 先验值集合 COCOMO Ⅱ.1998

所需的软件可靠性(RELY)	效用范围	很低	低	中等	高	很高
定义	最大值/最小值	轻微不便	易于恢复的较低损失	易于恢复的中等损失	较高经济损失	威胁到人员生命安全
先验值(1998)	1.41/0.74=1.91	0.74	0.88	1.0	1.16	1.41

3.3 样本信息

样本信息的来源是开始于 1994 年的数据采集活动,收集的数据包括作为自变量的实际软件大小,用千源代码行(KSLOC)计量;作为因变量的实际工作量,用人月(PM)来度量。1998 年收集的数据库中 161 个数据点的工作量和软件大小的分布如图 1 所示:

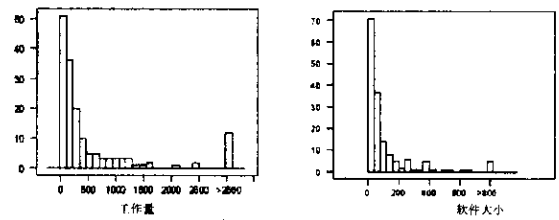

图 1 工作量和软件大小分布

数据收集完成后,核对两者的分布,以保证对其取对数的操作可行,并且服从正态分布,然后进行多元回归分析[6]。

多元回归模型可以表示为:

$$y_t=\beta_0+\beta_1x_{t1}+\cdots+\beta_kx_{tk}+\varepsilon_t \tag{5}$$

其中 $x_{t1}\cdots x_{tk}$——第 t 次观测的自变量值;

$\beta_0\cdots\beta_k$——待估系数;

ε_t——第 t 次观测的常规误差项;

y_t——第 t 次观测的因变量值。

对式(1)等号左右两边取自然对数,使其线性化,得到:

$$\ln(Effprt)=\beta_0+\beta_1\cdot 1.01\cdot\ln(Size)+\beta_2\cdot SF_1\cdot\ln(Size)+\cdots+\beta_6\cdot SF_5\cdot\ln(Size)+\beta_7\cdot\ln(EM_1)+\beta_8\cdot\ln(EM_2)+\cdots+\beta_{22}\cdot\ln(EM_{16})+\beta_{23}\cdot\ln(EM_{17}) \tag{6}$$

因此,β 向量的计算公式如下:

$$\vec{\beta}=(X^T\times X)^{-1}\times X^T\times X \tag{7}$$

其中$\vec{\beta}$是 β 参数值的列向量,也就是我们所关心的校正因子:

$$\vec{\beta}^T=(\beta_0\quad\beta_1\quad\beta_2\quad\cdots\quad\beta_{23})$$

Y 是矩阵 $Y^T=(\ln(Effort_1)\ \ln(Effort_2)\ \cdots\ \ln(Effort_i))$,$Effort_i$ 是第 i 个采样数据中的工作量值。

X 是矩阵:

$$X^T=\begin{pmatrix}1 & 1 & \cdots & 1\\ \ln(Size_1) & \ln(Size_2) & \cdots & \ln(Size_i)\\ SF_{1,1}\times\ln(Size_1) & SF_{1,2}\times\ln(Size_2) & \cdots & SF_{1,i}\times\ln(Size_i)\\ \vdots & & & \\ SF_{5,1}\times\ln(Size_1) & SF_{5,2}\times\ln(Size_2) & \cdots & SF_{5,i}\times\ln(Size_i)\\ \ln(EM_{1,1}) & \ln(EM_{1,2}) & \cdots & \ln(EM_{1,i})\\ \ln(EM_{2,1}) & \ln(EM_{2,2}) & \cdots & \ln(EM_{2,i})\\ \vdots & & & \\ \ln(EM_{17,1}) & \ln(EM_{17,2}) & \cdots & \ln(EM_{17,i})\end{pmatrix}$$

$Size_i$ 是第 i 个采样数据中的软件大小值,$SF_{j,i}$ 是第 i 个采样数据中的第 j 个 SF 的值,$EM_{j,i}$ 是第 i 个采样数据中的第 j 个 EM 的值。

样本数据的期望值(b)和方差(S^2)的计算公式如下:

$$RSS = (Y - X\beta)^T (Y - X\beta) \tag{8}$$

$$S^2 = \frac{RSS}{n - P'} \tag{9}$$

$$b = EM_i^{\beta} \text{ 或 } SF_i^{\beta} \tag{10}$$

其中 n——X 的行数,即采样数据数;

P'——自由度,当 $\beta_0 = 0$ 时,P'的取值为 X 的列数;当 $\beta_0 \neq 0$ 时,P'的取值为 X 的列数 + 1。

3.4　综合先验信息和样本信息[7]

式(3)和式(4)表明如果先验信息(H^*)的精度比样本数据高,那么后验分布值将更接近于先验分布值。这种情况发生在收集到的数据分布很散,噪音很大,如图 2 所示。

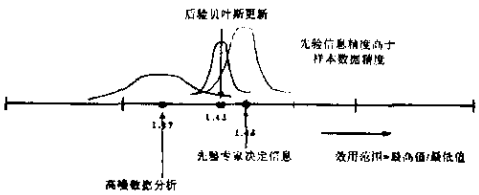

图 2　高噪数据后验贝叶斯更新(LTEX)

图 2 显示先验信息的置信度高于样本数据的置信度。于是,被赋于更高权重的先验信息使得 后验分布均值更接近于先验分布均值。相反,如果样本信息的精确度高于先验信息,那么,样本信息将获得较高的权重,引起后验分布均值更接近于样本数据。最后得到的后验分布精度一定比先验分布和样本数据的精度都要高。从表 2 可以清楚地看到采用贝叶斯算法对个体校正后,COCOMO Ⅱ. 1997(83 个数据点)和 COCOMO Ⅱ. 1998(161 个数据点)模型的估算精度有了较大的提高。

表 2　COCOMO Ⅱ. 1997 和 COCOMO Ⅱ. 1998 贝叶斯校正前后预算精度比较

COCOMO Ⅱ	预测精确度	个体校正前	个体校正后
1997	误差在 20% 以内	46%	49%
	误差在 25% 以内	49%	55%
	误差在 30% 以内	52%	64%
1998	误差在 20 以内	56%	66%
	误差在 25% 以内	65%	74%
	误差在 30% 以内	71%	76%

4　相关实验及结果

根据 COCOMO Ⅱ 模型及贝叶斯校正算法,开发了一个简单的软件估算演示程序,对该算法的有效性进行验证。

本演示程序使用 VB. NET 及 SQL Server2000 编写,实现的主要功能包括:

- 使用 COCOMO Ⅱ 模型,按源代码行数估算软件开发成本;
- 更新比例因子(SF)和倍乘因子(EM)各级别的取值;
- 更新数据库中的项目案例;
- 使用贝叶斯校正算法对估算模型进行校正。

数据流图及程序界面由图 3、图 4 所示。

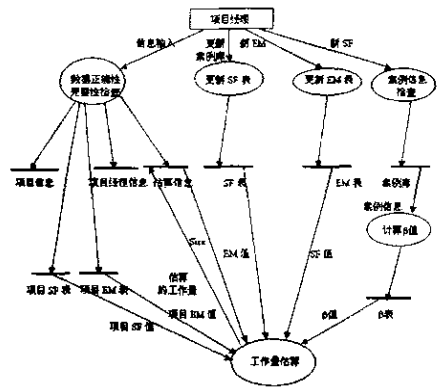

图 3　数据流图

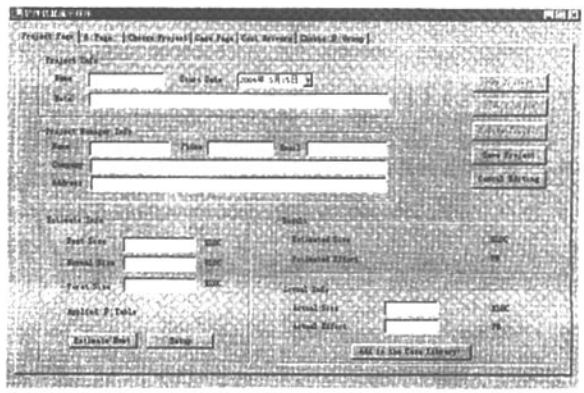

图 4　程序界面

使用该演示程序,对从十个企业收集到的实际数据进行处理,结果发现,采用贝叶斯校正算法能够有效地提高估算精度。工作量估算误差在 30% 以内的项目占各个企业总项目数的百分比在校正前后的分布如图 5 所示,平均提高了 12.9%,说明该算法确实能够比较有效地提高 COCOMO Ⅱ 模型的估算精度。

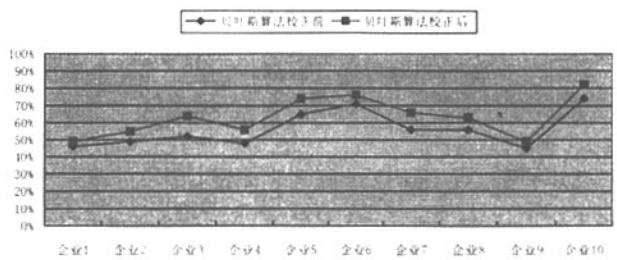

图 5　采用贝叶斯校正算法前后估算精度对比

5　结　论

面对软件工程界最大的问题之一——使用数量较少、不够完整的数据进行软件估算,贝叶斯校正算法采用有机组合基于经验的专家决定信息和样本数据信息来保证逻辑一致的方法较好地解决了这个问题,是对 COCOMO Ⅱ 软件估算模型的进一步完善。

参 考 文 献

[1] Barry W. Boehm, Chris Abts, A. Winsor Brown. Software cost estimation with COCOMO Ⅱ, Prentice-Hall, July 2000.

(下转第 159 页)

```
Do
  For i = 1 to PopulationSize
    If f(x_i) < f(p_i) then
      p_i = x_i ; p̄_i = x̄_i ;
    end if
    p_g = min {p_i} ; p̄_g = min {p̄_i} ;
    For d = 1 to Dimension
      v_id = ωv_id + c_1 r_1 (p̄_id − x̄_id) + c_2 r_2 (p̄_gd − x̄_id) ;
      x̄_id = x̄_id + v_id ;
      If sign(x̄_id) ≤ 0.5 then x_id = 0 else x_id = 1 ;
    Next d
Next i
Until termination criterion is met ;
Return( p_g , f(p_g))
```

在算法中，Initialize population 表示随机产生初始大小为 PopulationSize 的种群，Dimension 为解空间的维数；c_1 和 c_2 是加速正常数，其作用是：c_1 调节粒子飞向自身最好位置方向的步长，而 c_2 调节粒子飞向全局最好位置方向的步长。此外，$v_{ij} \in [-5,5]$，$\bar{x}_{ij}, \bar{p}_{ij}, \bar{p}_{gj} \in [-5k, 5k]$，$k = 1.5$ 或 2，$x_{ij}, p_{ij} \in \{0,1\}$。

4　实验结果和结论

下面首先简要介绍著名的 SAT 问题[8]，然后以其为测试实例，对 DS_BPSO 算法和 BPSO 算法进行数值计算比较。

所谓 SAT(Satisfiability)问题，是指对于给定命题变元集 $M = \{p_1, p_2, \cdots, p_m\}$ 上的一个合式范式(CNF) $A = C_1 \wedge C_2 \wedge \cdots \wedge C_i \wedge \cdots \wedge C_n$，判断是否存在一个指派 $t = <t_1, t_2, \cdots, t_m>$ 使 A 是可满足的(即 $t(A) = 1$)，$t \in \{0,1\}$。其中，$C_i = p_{i1} \vee p_{i2} \vee \cdots \vee p_{ik} \vee \neg p_{i,k+1} \vee \neg p_{i,k+2} \vee \cdots \vee \neg p_{ir}$ 称为子句，$p_{ij} \in M$。对于 CNF A，如果每个子句 C_i 中最多只含有 3 个不同的变元，则称为 3-SAT 问题。3-SAT 问题是著名的 NP 难问题。注意到 $t(A) = 1$ 当且仅当 $C_i, t(C_i) = 1$。如果将 A 中的 p_i 替换为整型变量 x_i，$\neg p_i$ 替换为 $(1 - x_i)$，$x_i \in \{0,1\}$，将 $\wedge$ 替换成普通加法，$\vee$ 替换成普通乘法，那么 SAT 问题是可满足的等价于 $\{0,1\}^m$ 上的整型多项式函数 $f(X) = \prod_{i=1}^{n} f(C_i) = 1$；是不可满足的等价于 $f(X) = 0$。其中，$f(C_i) = x_{i1} + x_{i2} + \cdots + x_{ik} + (1 - x_{i,k+1}) + (1 - x_{i,k+2}) + \cdots + (1 - x_{i,r})$，$x_{ij}$ 对应于 p_{ij}，$X \in \{0,1\}^m$。又 $f(X) = 1$ 当且仅当 $g(X) = \sum_{i=1}^{n} = n$；$f(X) = 0$ 当且仅当 $0 \leqslant g(X) < n$ 因此有 $t(A) = 1 \Leftrightarrow g(X) = n \Leftrightarrow g(X)/n = 1$；$t(A) = 0 \Leftrightarrow 0 \leqslant g(X) < n \Leftrightarrow 0 \leqslant g(X)/n < 1$。于是，判断 SAT 问题是否可满足的等价于 $g(X)/n$ 是取得最大值 1。

例如，3-SAT 实例 $A = (\neg p_1 \vee p_2 \vee p_3) \wedge (p_1 \vee p_3) \wedge (p_1 \vee p_2 \vee \neg p_3)$ 是可满足的，当且仅当 $g(X)/3 = \{[(1 - x_1) + x_2 + x_3] + [x_1 + x_3] + [x_1 + x_2 + (1 - x_3)]\}/3 = (2 + x_1 + 2x_2 + x_3)/3$ 取得最大值 1。

为了体现 DS_BPSO 算法优势，采用随机产生不同规模的 3-SAT[8] 测试实例，在 DELL Pentium(R)4 -CPU1.69GHz 型微机上，利用 C++ 语言编程对 DS_BPSO 算法和 BPSO 算法从算法执行迭代次数的平均数、标准方差和求出正确解的成功率等三个方面进行对比。其中 3-SAT 测试实例的规模定义为 (m,n)，其中 m 是变元个数，n 为表示 3-SAT 问题的合取范式中所含子句个数[8]。每种规模的测试实例个数为 100，此外，算法的最大迭代次数设为 10000。实验结果见表 1。

表 1　BPSO 算法与 DS_BPSO 算法对 3-SAT 的实验结果比较

算法	BPSO 算法			DS_BPSO 算法		
3-SAT 规模	迭代平均数	标准方差	成功率	迭代平均数	标准方差	成功率
$m=30, n=90$	924.31	2957.49	59%	6.20	28.35	100%
$m=50, n=150$	3795.07	4707.37	21%	198.12	364.20	100%
$m=80, n=240$	8190.66	1560.28	6%	2307.75	547.37	93%
$m=100, n=300$	10000.00	0	0%	5491.04	880.92	85%
$m=150, n=450$	10000.00	0	0%	7928.39	1507.72	61%

由表 1 可以看出：随着 3-SAT 测试实例规模的递增，DS_BPSO 算法的成功率相比 BPSO 算法越来越好；并且由算法执行迭代次数的平均数和标准方差可以发现：DS_BPSO 算法的执行效率和全局搜索能力均远远超过 BPSO 算法，而且 DS_BPSO 算法还具有较好的鲁棒性。显然，在求解离散优化问题时，DS_BPSO 算法是一种比 BPSO 算法更优的算法，这主要在于 DS_BPSO 算法不仅保留了 PSO 算法的所有优点，而且也继承了 BPSO 算法在处理离散优化问题时的灵活特性，因此非常适用于求解最优解可表示为二进制向量的各种离散优化问题。

参　考　文　献

[1] Kennedy J. and Eberhart R. C., Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks (Perth). IEEE Service Center, Piscataway, NJ, IV. 1995, 1942 ~1948.

[2] Eberhart R. C. and Kennedy J., A new optimizer using particle swarm theory. The 6th Int1 Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995.

[3] 徐宗本，计算智能—模拟进化计算，北京：高等教育出版社，2005.

[4] Kennedy J. and Eberhart R. C., A discrete binary particle swarm optimization. Procceedings of 1997 Conference on System, Man, and Cybernetices. Piscataway, NJ, IEEEService Center, 1997, 4104 ~4109.

[5] Clerc M., Discrete particle swarm optimization illustrated by the Traveling Salesman Problem. http://www.mauriceclerc.net, 2000 -2 -29.

[6] Frans Van den Bergh. An analysis of particle swarm optimizers [PhD dissertation]. Pretoria: University of Pretoria, 2001.

[7] 许承德、王勇，概率论与数理统计，北京：科学出版社，2001.

[8] 张健，逻辑公式的可满足性判定——方法、工具及应用，北京：科学出版社，2000.

(上接第 153 页)

[2] Sunita Chulani, Brad Clark, Barry Boehm. Calibrating the COCOMO Ⅱ Post Architecture Model, 20th International Conference on Software Engineering, (April 1998).

[3] Sunita Chulani, Barry Boehm, Bert Steece. Calibrating Software Cost Models Using Bayesian Analysis. submitted for the IEEE Transactions on Software Engineering, Special Issue on Empirical Methods in Software Engineering, (Fall 1998).

[4] 张孝令、刘福升等，贝叶斯动态模型及其预测，山东科学技术出版社，1992.8.

[5] 张尧庭、陈汉峰，贝叶斯统计推断，北京：科学出版社，1991.3.

[6] 胡国定、张润楚，多元数据分析方法——纯代数处理，天津：南开大学出版社，1990.6.

[7] Sunita Chulani, Bert Steece. A Bayesian Software Estimating Model Using a Generalized g-Prior Approach.