第33卷 第6期 计 算 机 工 程 2007年3月
Vol.33 No.6 Computer Engineering March 2007
·博士论文· 文章编号：1000—3428(2007)06—0045—03 文献标识码：A 中图分类号：TP311.5

基于缺陷度量的软件质量成本分析模型

苑 畅，胡克瑾
(同济大学经济与管理学院，上海 200092)

摘 要： 针对软件项目的质量成本效益无法直接度量的这一难题，提出了建立组织级质量成本度量基线的解决方案。设计了基于缺陷度量的软件质量成本分析模型，实现了对软件质量成本及其效益的定量度量与分析，为评价软件质量保证活动绩效、指导质量投资决策提供了试用方法。

关键词： 软件度量；软件质量成本；软件缺陷

Cost of Software Quality Analysis Model Based on Defect Metrics

YUAN Chang, HU Kejin
(College of Economics and Management, Tongji University, Shanghai 200092)

【Abstract】 It is hard to directly measure the benefit of a software project's quality cost. The paper supplies a solution to set up an organizational metrics baseline of quality cost. Meanwhile, the paper presents a quality cost analysis model based on defect metrics, which realizes the quantitative analysis of quality cost and its benefit, and provides objective evidences and guidelines for software quality assurance's performance appraisal and quality investment decision-making.

【Key words】 Software metrics; Cost of software quality; Software defect

软件质量保证的目的是通过对预防和评估活动的投入来减少软件质量损失，因此软件质量成本效益分析的关键在于分析质量投入与质量成本损失的变化关系。

目前，ISO9001-2000、SW-CMM 和 CMMI 中没有提供软件质量成本分析方法。S.T.Knox 等研究了软件质量成本的细分和 CMM 认证企业的质量成本数据，得到了各 CMM 等级中质量成本的比例[1]，但没有提出进行软件质量成本分析的可操作方法。A.R.Hevner 提出了基于生命周期阶段的软件度量模型[2]，但没有与质量成本分析结合起来。软件质量成本效益分析的难度在于质量投入是直观的，而带来的价值，即质量损失的降低($\nabla NCoSQ$)是隐性的、难于度量的。

本文提出了可以将软件项目质量成本与组织级质量成本基线相比较，从而获得该项目的$\nabla NCoSQ$数据，实现了$\nabla NCoSQ$的可度量。并设计了基于缺陷度量的质量成本分析模型，为指导质量成本分析、质量投资决策提供了便捷的方法。

1 软件质量成本分析

1.1 软件质量成本结构

质量成本分两部分：一是为确保质量而发生的费用称为符合性成本(ACoSQ)，也称为质量投入；二是由于没有达到质量要求所造成的损失称为不符合性成本(NCoSQ)，也称为质量损失。根据软件项目的特点，可将软件质量成本分解为如表 1 所示的结构。

如图 1[3]所示，在达到最低的质量总成本点 A 之前，质量投入的增加能带来更多的质量损失的降低，即$\frac{\nabla|NCoSQ|}{\nabla|ACoSQ|}>1$，并使总质量成本持续减少。因此，企业在(B,A)区间的质量投入都是经济合理的。而图 2[1]说明了随着软件企业能力成熟度的提高，软件质量损失和质量总成本不断下降。由此可知软件项目的质量总成本是处在图 1 的(B,A)区间内，即软件企业加大质量投入是必要和经济的。

表 1 软件质量成本结构

软件质量总成本(TCoSQ)	分 类	质 量 投 入
软件质量符合性成本 (Achievement Costs of Software Quality , ACoSQ)	鉴定成本	评审(各阶段的评审)
		走查
		测试(首次)
		独立验证与确认(首次)
	预防成本	与质量保证活动相关的培训
		质量方针、规程和方法的研究与制定
		质量保证工具
		质量策划及质量改进方案
		质量数据的收集及分析
		缺陷的原因分析
		质量报告的编制
软件质量不符合性成本 (Nonconformance Costs of Software Quality, NCoSQ)	内部及外部不符合成本	修正缺陷
		文档返工
		修改代码
		再评审
		再测试
		再测试的实验室成本
		变更控制委员会(CCB)相关的变更工作量
		外部失败及赔偿
		客户支持

作者简介： 苑 畅(1971－)，女，博士生，主研方向：软件质量管理；胡克瑾，教授、博导

收稿日期： 2006-03-25 **E-mail：** yueshun-consult@163.com

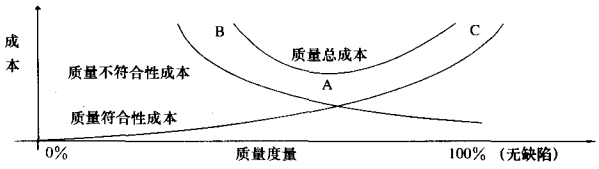

图 1 质量成本模型

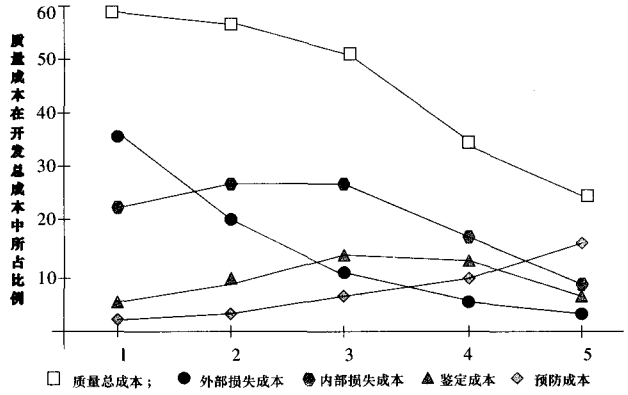

图 2 软件质量成本与 CMM/CMMI 等级

1.2 软件质量成本的度量

对软件质量成本的度量是实施质量成本分析的前提[4]。如表 1 所示，通过对各项质量保证活动工作量的度量，可按人员成本折算出项目的鉴定成本和不符合性成本；而预防成本包括了各项目的预防成本和在企业范围内为保证质量所提供的培训、工具和过程改进等花费的成本。后者可按比例分摊到各项目中。因此，软件项目的总成本、鉴定成本、预防成本和不符合性成本都是可度量的。进行质量成本效益分析，需要度量 $\nabla ACoSQ$ 和 $\nabla NCoSQ$。但对于一个项目而言，$\nabla ACoSQ$ 和 $\nabla NCoSQ$ 是无法直接度量的，因为项目是不可逆的，所以无法度量 ACoSQ 的变化会产生如何不同的 NCoSQ。

借鉴 CMM/CMMI 理念，本文认为可建立软件企业范围内的质量成本度量体系，在对企业内各类项目度量的基础上建立企业的质量成本基线，包括 $\overline{ACoSQ}$ 和 $\overline{NCoSQ}$，当与某一具体项目的质量成本数据进行比较，即可计算出该项目的 $\nabla ACoSQ$ 和 $\nabla NCoSQ$。

2 基于缺陷度量的质量成本分析模型

2.1 基于阶段的缺陷度量

软件过程改进的理念是“过程化思维”，(见图 3[5])，即软件生命周期被分割为不同阶段或过程，质量控制的要点在于对进入每个阶段的“输入”，即上阶段完成的工作产品进行质量验证与确认，同时对本阶段的“输出”，即递交给下一阶段的工作产品进行质量验证与确认。采用阶段式质量控制的目的是将每阶段引入的缺陷尽量控制在本阶段内解决，避免缺陷的跨阶段传递和扩散。

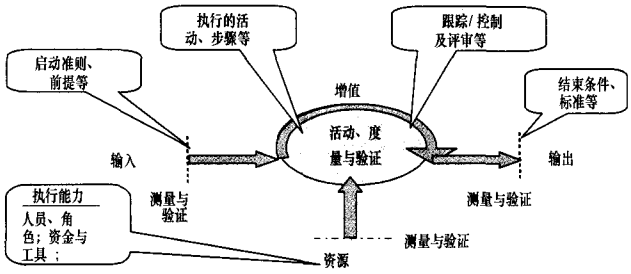

图 3 基于过程控制的软件活动

E.Weller 和 D.Paulish 的研究也表明：缺陷在其引入阶段内被发现和排除的成本最低，而当缺陷跨越阶段进行传递时，它的影响范围、排除成本逐级放大[6, 7]。因此，软件质量控制的关键在于使各阶段引入的缺陷最小化，并尽早地发现和排除各类缺陷。

实施基于阶段的缺陷度量的前提首先是在软件生命周期过程中具有系统化、规范化的质量保证制度和措施，如各阶段的评审、测试、走查等，包括针对需求规格说明书、设计说明书、代码、测试用例等关键工作产品的编写规范、评审标准、缺陷分析标准等。

其次，度量要与软件开发过程紧密结合，软件阶段的划分为度量提供了数据采集点和采集机制。本文参照通用的瀑布生命周期模型，将软件生命周期划分为 5 个阶段：(1)需求阶段；(2)设计阶段；(3)实现阶段(编码、单元测试)；(4)测试阶段(集成测试、系统测试、验收测试)；(5)运行阶段。

如果发现的缺陷是在本阶段引入的，定义为“阶段内缺陷”；如果是在其引入阶段之后发现的，则定义为“跨阶段缺陷”。基于阶段的缺陷度量矩阵如表 2 所示，该矩阵记录了整个生命周期各阶段发现的缺陷数量、缺陷来源、缺陷被发现阶段、占总缺陷的比率和缺陷传递率等信息。

表 2 基于阶段的缺陷度量矩阵

引入阶段 \ 发现阶段		1 需求	2 设计	3 实现	4 测试	5 运行	缺陷比率%	缺陷传递率%
1	需求	M_1	N_{12}	N_{13}	N_{14}	N_{15}	T_1/T	N_1/T_1
2	设计		M_2	N_{23}	N_{24}	N_{25}	T_2/T	N_2/T_2
3	实现			M_3	N_{34}	N_{35}	T_3/T	N_3/T_3
4	测试				M_4	N_{45}	T_4/T	N_4/T_4
5	运行					M_5	T_5/T	

设 M_i 为第 i 阶段引入并在本阶段排除的缺陷数；N_i 为由第 i 阶段传递出的缺陷总和；T_i 为第 i 阶段引入的缺陷总和，

则 $N_i=\sum_{j=i+1}^{5} N_{ij}$；$T_i = M + N_i$ (1)

$N=\sum_{i=1}^{4} Ni$；$M=\sum_{i=1}^{5} M_i$；$T=\sum_{i=1}^{5} Ti$ (2)

2.2 基于缺陷度量的质量成本分析模型

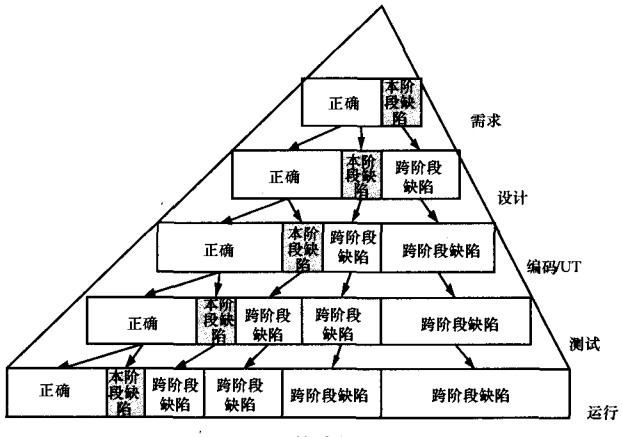

图 4 缺陷传递

如表 1 所示，质量不符合性成本主要包括由软件缺陷引起的各类返工，因此软件缺陷造成的返工能够代表软件的不符合性成本(如发生外部赔偿，则需要增加赔偿成本)。返工工作量不但便于度量，而且可直接折算为成本。因此，本文以返工工作量作为度量缺陷成本的基础度量元。为简便起见，暂不考虑缺陷严重等级不同所造成的对返工工作量的影响，以平均缺陷返工工作量作为度量基准。

根据图 4，随着项目开发工作的进行，缺陷在阶段间是不断传递、放大和新增的。

设：Di 为第 i 阶段存在的缺陷总和；

C_{ij} 为第 i 阶段缺陷传递到第 j 阶段的缺陷放大倍数；

则

$D_1= M_1$

$D_2= C_{12}D_1+ M_2 = C_{12}M_1+ M_2$

$D_3= C_{23}D_2+ M_3 = C_{12}C_{23}M_1+ C_{23}M_2+ M_3= C_{13}M_1+ C_{23}M_2+ M_3$

$D_4= C_{34}D_3+ M_4 = C_{12}C_{23}C_{34}M_1+ C_{23}C_{34}M_2+ C_{34}M_3+ M_4= C_{14}M_1+ C_{24}M_2+ C_{34}M_3+ M_4$

$D_5= C_{45}D_4+ M_5 = C_{45}C_{12}C_{23}C_{34}M_1+ C_{23}C_{34}C_{45}M_2+ C_{34}C_{45}M_3+ C_{45}M_4+ M_5= C_{15}M_1+ C_{25}M_2+ C_{35}M_3+ C_{45}M_4+ M_5$

由此得到缺陷传递矩阵 D：

$$\boldsymbol{D} = (D_1, D_2, D_3, D_4, D_5) = (M_1,M_2,M_3,M_4,M_5) \times \begin{pmatrix} 1 & C_{12} & C_{13} & C_{14} & C_{15} \\ 0 & 1 & C_{23} & C_{24} & C_{25} \\ 0 & 0 & 1 & C_{34} & C_{35} \\ 0 & 0 & 0 & 1 & C_{45} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

其中：$C_{i(j+1)}= \prod_{k=i}^{j} C_{k(k+1)}$，I= 1…4；j=1…4 (4)

表 3 缺陷质量成本分析模型

引入阶段 \ 发现阶段		1 需求	2 设计	3 实现	4 测试	5 运行
1	需求	1	C_{12}	C_{13}	C_{14}	C_{15}
2	设计		1	C_{23}	C_{24}	C_{25}
3	实现			1	C_{34}	C_{35}
4	测试				1	C_{45}
5	运行					1

以排除一个阶段内缺陷的工作量为基准，即 C_{ii}=1，则缺陷传递矩阵 D 可转化为缺陷质量成本分析模型(如表 3 所示)。

设：C_{Ni} 代表第 I 阶段引入的缺陷传递到后续阶段所造成的质量成本倍数；C_N 为整个生命周期中，跨阶段缺陷引起的质量成本倍数；则

$$C_{Ni} = \sum_{j=1}^{4} (\prod_{k=i}^{j} C_{k(k+1)}) N_{i(j+1)}, \quad I= 1…4; \quad (5)$$

$$C_N = \sum_{i=1}^{4} C_{Ni} = \sum_{i=1}^{4} [\sum_{j=1}^{4} (\prod_{k=i}^{j} C_{k(k+1)}) N_{i(j+1)}] \quad (6)$$

2.3 基于缺陷度量的质量成本分析模型的应用

2.3.1 缺陷原因分析

缺陷原因分析的目的是通过缺陷的表现形式，发现缺陷产生的根本原因。在运用基于阶段的度量数据分析缺陷原因时，可以按照不同阶段工作产品的特点，对每阶段各类工作产品中可能存在的缺陷进行识别和分类，形成“缺陷原因列表”。以“缺陷原因列表”为指导有利于快速定位缺陷的来源、分析缺陷产生的原因、为制定缺陷排除方案提供依据，有利于提高软件开发质量和效率。

2.3.2 项目决策分析

软件缺陷会带来返工，而返工是造成项目延期、成本超支的最主要原因。而项目团队往往迫于进度紧张、人员紧缺等原因而放弃了各阶段的质量保证工作。跟踪返工工作量，分析对项目成本、进度的不良影响，并在组织范围内进行横向比较，可以得出质量成本投入与质量成本损失间此削彼长的关系；并能客观评价软件质量保证活动的效率和效果、指导项目团队权衡项目期限、成本、质量与项目风险间的约束关系，为制定明智的项目决策提供依据。

2.3.3 质量成本分析和缺陷跟踪分析

将缺陷状态划分为“打开”、“处理”和“关闭”状态，并对其进行动态地跟踪和更新、度量缺陷发现率、缺陷修复时间、未修复缺陷数等指标。将项目的具体质量成本数据与组织的质量成本基线进行比较，计算出项目的 $\nabla ACoSQ$、$\nabla NCoSQ$ 以及 $\frac{\nabla|NCoSQ|}{\nabla|ACoSQ|}$，从而客观地评价质量投入和质量保证工作的效果。

3 实例分析

该实例为某软件企业的一个开发周期为 6 个月，计划工作量为 28 人月的开发项目。该项目采用瀑布模型，生命周期划分为需求、设计、实现、测试和运行维护五个阶段，运行维护期限为半年。

表 4 缺陷质量成本分析矩阵

引入阶段 \ 发现阶段		1 需求	2 设计	3 实现	4 测试	5 运行
1	需求	0.5	1.5	3	6	22.5
2	设计		0.5	2	4	12
3	实现			0.5	2	6
4	测试				0.5	3
5	运行					0.5

根据该企业的质量成本度量基线和此项目的实际度量数据，对各阶段缺陷排除成本和 C_{ij} 值进行了估算和赋值：设发现和排除一个本阶段缺陷的质量成本为 0.5 人日的工作量；取 C_{12}=3；C_{23}=4；C_{34}=4；C_{45}=6；则该项目的缺陷质量成本分析矩阵如表 4 所示。将该项目的各阶段缺陷数据和缺陷质量成本汇总在表 5 中。

表 5 项目缺陷及缺陷成本度量表

缺陷数量及质量成本		发现阶段 需求	设计	实现	测试	运行	小计	缺陷比例%	缺陷传递率%
引入阶段	需求	4	1	2	1	0	T1=8	3.68	50
	缺陷成本	2	1.5	6	6	0	C1=15.5	4.36	
	设计		6	5	1	0	T2=12	5.53	50
	缺陷成本		3	10	5	0	C2=18	4.62	
	实现			54	116	15	T3=185	85.25	70.8
	缺陷成本			27	232	90	C3=349	89.49	
	测试				2	0	T4=2	0.92	0
	缺陷成本				1	0	C4=1	0.26	
	运行					10	T5=10	4.61	
	缺陷成本					5	C5=5	1.26	
总计：							T= 217；C=388.5(人日)		

3.1 项目分析

返工工作量为 388.5 人日(8 小时工作制)，折算为 17 人月，占项目总工作量的 38%。采取加班措施使实际进度延期为 1 个月。从该项目中可发现以下问题：(1)项目返工成本过高，达到 38%。企业处在 CMM2 级水平[1]；(2)项目进度延期低于返工工作量，说明原项目计划中人员工作任务不饱满，而在发生返工时又造成大量加班。

3.2 缺陷原因分析

(1)实现阶段的缺陷数目和返工工作量过高，说明有必要加强开发技术培训、制定明确的编程规范；

(2)缺陷传递率过高，说明项目各阶段的评审和测试工作没有严格执行，或者执行效果不理想。

3.3 质量成本分析

该公司的质量成本基线与该项目的实际质量成本度量数据汇总在表 6 中。由表 6 计算出该项目的 $\nabla ACoSQ$ 和 $\nabla NCoSQ$ 分别为-6.7%和-8.5%，$\frac{\nabla|NCoSQ|}{\nabla|ACoSQ|}$ =1.27，说明

(下转第 68 页)

4 实验及算法分析

为测试算法性能，用 VC++6.0 实现了算法，在 Celeron 2.4GHz CPU、512MB 内存、Windows XP 系统的 PC 机上进行了测试。测试数据是 IBM Almaden 研究中心的 KDD 研究小组提供的数据生成器生成的合成数据(Http://www.almaden.ibm.com/cs/quest/syndata.html)。参数设置规则同文献[1]。

通过把一维 CFP 树挖掘算法与 PrefixSpan 算法进行比较，发现 CFP 树挖掘算法在处理大规模数据时具有更好的性能。在实验数据是 C10-T2.5-S4-I1.25、最小支持度阈值是 0.25%的情况下，处理时间和数据库大小之间的关系显示 CFP 树挖掘算法在数据库规模变大时具有更明显的优势。在实验数据是 C10-T2.5-S4-I1.25-D50 的情况下，处理时间和最小支持度阈值之间的关系显示 CFP 树算法在最小支持度阈值在 0.25%~1.5%之间都有明显优势。

PrefixSpan[2] 算法基于频繁模式增长的思想，采用 divide-and-conquer 策略以减少每次搜寻可能的频繁模式的搜索空间。但中间过程增加了建立大量后缀投影数据库的额外开销。在处理大规模数据时，不能有效地避免投影开销的迅速增加。CFP 树挖掘算法使用渐进的前缀序列搜索机制能最大可能地减少可能频繁模式的搜索空间，并且中间过程只需存储必要的后缀树的头结点信息。

此外，CFP 树是一种紧凑的序列数据存储结构。由于相同子序列前缀的存在，CFP 树的规模的增长速度低于数据量的增加速度。在处理大规模数据时，数据的紧凑存储使得可能频繁模式的搜索开销并不会随着数据量的增加而急剧增长。这是 CFP 树挖掘算法比 PrefixSpan 算法在处理大规模数据时具有优势的原因。但是，当数据规模较小时，建立 CFP 树的开销会在一定程度上影响算法的执行效率。

5 结束语

本文提出了一个能从大规模数据库中高效地挖掘出频繁序列模式的新算法，该算法同时适用于一维和多维序列数据。提出项目关系标识符的概念，为一维和多维序列数据建立了统一的简化线性形式，并进一步利用编码机制，建立了统一的紧凑存储结构 CFP 树。利用 ID-queue 链接和定位码机制，能方便地找出每棵后缀树的下一个频繁头结点 ID，避免了整棵树的遍历，更不用递归地重建新的子树。中间过程只需存储后缀树的头结点信息，减少了大量内存和 CPU 开销。因此，这种渐进的前缀序列搜索机制能高效地挖掘频繁序列模式。实验证明，与 PrefixSpan 算法相比，CFP 树挖掘算法在处理大规模数据时更高效。以后的工作主要是进行 CFP 树的增量挖掘算法。

参考文献

1 Agrawal R, Srikant R. Mining Sequential Patterns[C]//Proceedings of the International Conference on Data Engineering, Taipei, Taiwan. 1995: 3-14.

2 Pei J, Han J, Mortazavi A B. Mining Sequential Patterns by Pattern-growth: the Prefixspan Approach[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(11).

3 Yu C, Chen Y. Mining Sequential Patterns from Multidimensional Sequence Data[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(1): 136-140.

4 Pei J, Han J, Mortazavi A B. Mining Access Patterns Efficiently from Web Logs[C]//Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Kyoto, Japan. 2000.

5 Ezeife C I, Lu Y. Mining Web Log Sequential Patterns with Position Coded Pre-order Linked WAP-tree[J]. Data Mining and Knowledge Discovery, 2005, 10(1): 5-38.

(上接第 47 页)

由于质量保证活动执行得不充分，使项目造成了 1.27 倍的质量成本损失。因此，项目组要加强质量保证活动的执行力度。

表 6 质量成本基线及项目质量成本数据

工作量占项目总工作量的比率(%)	TCoSQ	ACoSQ	NCoSQ
组织的质量成本基线	50%	15%	35%
E－broker 项目的质量成本度量数据	52%	14%	38%

4 结论和展望

本文提出了解决质量投入价值无法直接度量的解决方案，并提出了基于缺陷度量的质量成本分析模型，为软件质量成本分析提供了可操作的方法。最后，以一个项目实例说明了质量成本分析模型的应用。该实例项目的规模不大，各阶段的缺陷度量数据有限，尤其是运行阶段的缺陷数据较少，在今后的研究中应当以大规模软件项目为实例来验证和改进组织的质量成本度量基线。

参考文献

1 Krasner H. Using the Cost of Quality Approach for Software[J]. Journal of Defense Software Engineering, 1998, 41(11): 6-11.

2 Hevner A R. Phase Containment Metrics for Software Quality Improvement[J]. Information and Software Technology, 1997, 39(5): 867-877.

3 Gryna F M. Quality Costs, Juran's Quality Control Handbook[M]. 4th Ed. New York: McGraw-Hill, 1998.

4 Fenton N E, Pfleeger S L. Software Metrics——A Rigorous and Practical Approach[M]. 2nd Ed. 北京: 机械工业出版社, 2004.

5 郑人杰. 基于软件成熟度模型(CMM)的软件过程改进: 方法与实施[M]. 北京: 清华大学出版社, 2003.

6 Weller E. Using Metrics to Manage Software Projects[J]. IEEE Computer, 1994, 27(9): 23.

7 Paulish D. Case Studies of Software-process Improvement Measurement[J]. IEEE Computer, 1994, 27(9): 11.