ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.4, April 2007, pp.775−795
DOI: 10.1360/jos180775

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

软件成本估算方法及应用*

李明树[1,2+], 何 梅[1,3], 杨 达[1,3], 舒风笛[1], 王 青[1]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

[2](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100080)

[3](中国科学院 研究生院,北京 100049)

Software Cost Estimation Method and Application

LI Ming-Shu[1,2+], HE Mei[1,3], YANG Da[1,3], SHU Feng-Di[1], WANG Qing[1]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[3](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: Phn: +86-10-82621122, Fax: +86-10-62562533, E-mail: mingshu@iscas.ac.cn, http://www.iscas.ac.cn

Li MS, He M, Yang D, Shu FD, Wang Q. Software cost estimation method and application. *Journal of Software*, 2007,18(4):775−795. http://www.jos.org.cn/1000-9825/18/775.htm

Abstract: Software cost estimation has played an important role in software development since its emergence in 1960's. Based on a classification of algorithmic model based methods, non- algorithmic model based methods and composite methods, the typical software cost estimation methods in history are overall reviewed. The issue of software sizing, which is closely related to software cost estimation, is also discussed in this paper. Then a three phases' evaluation criterion of software cost estimation methods is proposed and a case study on cost estimation of government sponsored projects in China is analyzed. At last, six possible trends from estimation models, estimation evolutions, estimation applications, estimation contents, supporting tools and human factors, are presented as a primary conclusion in the paper while viewing the future development for software cost estimation.

Key words: estimation; software cost estimation; algorithmic model; software sizing; evaluation; application

摘 要: 软件成本估算从 20 世纪 60 年代发展至今,在软件开发过程中一直扮演着重要角色.按照基于算法模型的方法、非基于算法模型的方法以及组合方法的分类方式,全面回顾、分析了软件成本估算的各种代表性方法,也归纳讨论了与成本估算强相关的软件规模度量问题.在此基础上,进一步研究了软件成本估算方法的评价标准,并给出了一个应用实例及其分析.最后,从估算模型、估算演进、估算应用、估算内容、工具支持和人为因素 6 个方面,指出了软件成本估算方法下一步的主要发展趋势.

关键词: 估算;软件成本估算;算法模型;软件规模度量;评价;应用

中图法分类号: TP311 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60573082, 60473060 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z182 (国家高技术研究发展计划(863)); the National Key Technologies R&D Program of China under Grant No.2005BA113A01 (国家科技攻关计划)

Received 2006-11-28; Accepted 2007-02-16

随着软件系统规模的不断扩大和复杂程度的日益加大,从 20 世纪 60 年代末期开始,出现了以大量软件项目进度延期、预算超支和质量缺陷为典型特征的软件危机,至今仍频繁发生.根据 Standish 组织在 1995 年公布的 CHAOS 报告显示,在来自 350 个组织的 8 000 个项目中,只有 16.2%是“成功的(succeeded)”,即能在预算和限期内完成;31.1%是“失败的(failed)”,即未能完成或者取消;其余 52.7%被称为“被质疑的(challenged)”,虽然完成但平均预算超支 89%[1].2004 年,该组织的统计项目数累计达到 50 000 多个,结果显示,成功项目的比例提升到 29%,而被质疑的项目比例仍有 53%[2](如图 1 所示).虽然有些研究认为,CHAOS 报告中关于预算超支 89%的数据被夸大了,实际情况应该平均在 30%~40%[3].但有一点却能够取得共识:人们经常对软件成本估算不足.它与需求不稳定并列,是造成软件项目失控最普遍的两个原因[4].

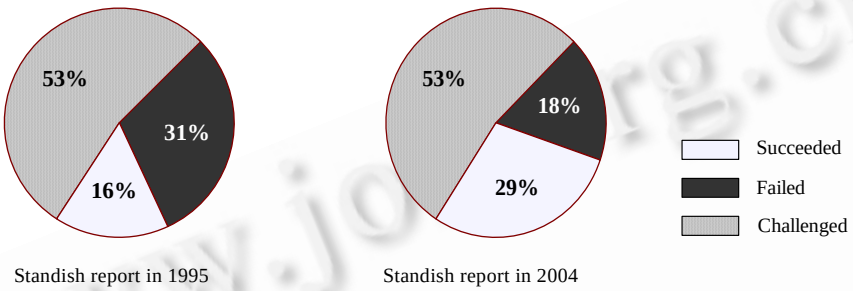


Fig.1 Statistical data on software projects from the Standish Group International, Inc.[1,2]

图 1 Standish 咨询公司对软件项目完成情况的统计数据[1,2]

无论是产业界还是学术界,越来越多的人认识到,做好软件成本估算是减少软件项目预算超支问题的首要措施之一,不但直接有助于作出合理的投资、外包、竞标等商业决定,也有助于确定一些预算或进度方面的参考里程碑,使软件组织或管理者对软件开发过程进行监督,从而更合理地控制和管理软件质量、人员生产率和产品进度.正如美国 Southern California 大学的 Boehm 教授所指出的,“理解并控制软件成本带给我们的不仅仅是更多的软件,而会是更好的软件”[5].

估算,即确定项目开发时间和开发成本的过程.值得一提的是,在谈及估算的时候,常会联想到众多量值,例如成本、工作量、资源、进度、规模、风险等.而“成本估算”与“工作量估算”这两个术语出现得最为频繁.由于当前人员成本通常占到整个软件项目成本的绝大部分,“成本估算”与“工作量估算”在很多情况下可交替使用,因此本文也不加以特别区分.而且,本文所指的“成本估算”主要是指“开发成本估算”.

最早的软件成本估算研究可以追溯到 20 世纪 60 年代的 SDC(system development corporation)线性模型,已历经了 40 年的发展[6].BESTweb (BEST=better estimation of software development tasks)论文数据库中收集了近几十年来主要国际期刊、会议发表的与软件成本估算相关的大部分论文[7].软件成本估算方法有很多种分类形式,本文根据是否采用算法模型(algorithmic model)分为 3 大类:基于算法模型的方法、非基于算法模型的方法以及组合方法(如图 2 所示).

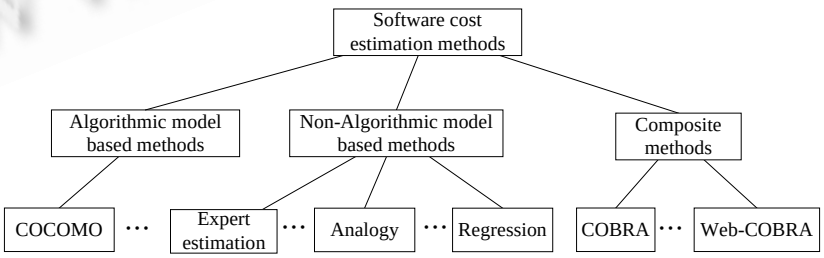


Fig.2 Classification of software cost estimation methods

图 2 软件成本估算方法的分类

本文首先从上述分类角度综述软件成本估算方法中最有代表性的一些方法和技术,然后概要介绍对软件成本估算起强相关作用的软件规模度量方法,再从应用的角度对软件开发成本估算方法的评价标准进行讨论,并给出一个应用实例及其分析.最后在估算模型、估算演进、估算应用、估算内容、工具支持和人为因素等 6 个方面给出我们对软件成本估算方法未来发展趋势的一些展望.

1 基于算法模型的软件成本估算方法

基于算法模型的软件成本估算方法,提供了一个或多个算法形式,如线性模型、乘法模型、分析模型、表格模型以及复合模型等,将软件成本估算为一系列主要成本驱动因子变量的函数[8].该方法通过成本估算关系(cost estimating relationship)把系统特征与工作量、进度的估算值联系起来.所谓算法就是从参数得到成本估算的一系列规则、公式.不同的算法模型不仅会在成本因子关系的表达式上有所区别,而且在因子的选取上也各不相同[9].

基于算法模型的方法的基本思想是[6]:找到软件工作量的各种成本影响因子,并判定它对工作量所产生影响的程度是可加的、乘数的还是指数的,以期得到最佳的模型算法表达形式.当某个因子只影响系统的局部时,我们一般说它是可加性的.例如,如果我们给系统增加源指令、功能点实体、模块、接口等,大多只会对系统产生局部的可加性的影响.当某个因子对整个系统具有全局性的影响时,我们则说它是乘数的或指数性的,例如,增加服务需求的等级或者不兼容的客户等.

基于算法模型的方法的主要优缺点是[8,10]:一方面,它们比较客观、高效、可重复,而且能够利用以前的项目经验进行校准,可以很好地支持项目预算、权衡分析、规划控制和投资决策等;另一方面,它们难以用在没有前例的场合,不能处理异常情况,也不能弥补不准确的规模输入和成本驱动因子级别的问题.

基于算法模型的方法,大体上可以分为 3 个主要发展阶段[6]:(1) 早期阶段(1965 年~1985 年):寻找好的模型方式和因子关系;(2) 中期阶段(1985 年~1995 年):活动求精、风险分析和规模定量;(3) 后期阶段(1995 年~2005 年):针对新软件开发风格的扩展.其中 COCOMO(constructive cost model,构造性成本模型)贯穿 3 个阶段不断发展和改进,是最为典型的算法模型,几乎无一例外地出现在对估算方法进行比较研究或用来作参照的文献中[11−15].同时,基于它或其扩展所实现的工具也一直在软件估算工具的主流之列[16].

在 COCOMO 软件成本模型的发展中,无论是最初的 COCOMO 81 模型[8],还是 20 世纪 90 年代中期提出的逐步成熟完善的 COCOMO II[17],所解决的问题都具有当时软件工程实践的代表性.作为目前应用最广泛、得到学术界与工业界普遍认可的软件估算模型之一,已经发展到了一组模型套件(包含了软件成本模型、软件扩展与其他独立估算模型 3 大类),形成了 COCOMO 模型系列,也给基于算法模型的方法提供了一个通用的形式[18]:

$$PM = A \times (\sum Size)^{\sum B} \times \prod (EM) \tag{1}$$

其中,PM 为工作量,通常表示为人月(person months);A 为校准因子(calibration factor);$Size$ 为对工作量呈可加性影响的软件模块的功能尺寸的度量;B 为对工作量呈指数或非线性影响的比例因子(scale factor);EM 为影响软件开发工作量的工作量乘数(effort multiplicative).

1.1 COCOMO 81

COCOMO 81 有 3 个等级的模型[8]:(1) 基本(basic)模型,在项目相关信息极少的情况下使用;(2) 中等(intermediate)模型,在需求确定以后使用;(3) 详细(detailed)模型,在设计完成后使用.

COCOMO 81 这 3 个等级的模型也都满足类似公式(1)的通式,即

$$Effort = a \times (\mathrm{KDSI})^b \times F \tag{2}$$

其中,$Effort$ 为工作量,表示为人月;a 和 b 为系数,具体的值取决于建模等级(即基本、中等或详细)以及项目的模式(即组织型、半独立型或嵌入型).这个系数的取值先由专家意见来决定,然后用 COCOMO 81 数据库的 63 个项目数据来对专家给出的取值再进一步求精.KDSI 为软件项目开发中交付的源指令(delivered source instruction,简称 DSI)千行数,也有用代码行(line of codes,简称 LOC)表示,代表着软件规模.F 是调整因子,基本

COCOMO 81 模型中,F=1,后两个模型中,F 为 15 个成本因子对应的工作量乘数的乘积.

在开发的起始阶段,项目的相关信息很少,只要确定软件项目的模式与可能的规模,就可以用基本 COCOMO 81 模型作一个工作量初始估算,见例 1.

例 1:要开发一个估计规模为 30KDSI 的银行系统应用程序项目,其功能以数据处理为主,属于组织型软件模式,根据专家意见和项目数据校准,系数 a=2.4,b=1.05;调整因子 F=1,则工作量 Effort 估算为

$$Effort=2.4\times(30)^{1.05}=85.3 \text{ 人月}.$$

随着项目的进展和需求的确定,可以使用中等 COCOMO 81 模型进行估算.中等 COCOMO 81 模型定义了 15 个成本因子(见表 1),按照对应的项目描述,可将各个成本因子归为不同等级.例如,当软件失效造成的影响只是稍有不便时,要求的软件可靠性因子(required software reliability,简称 RELY)等级为“很低”;当软件失效会造成很高的财务损失时,RELY 等级为“高”;当造成的影响危及人的生命时,RELY 等级为“很高”.不同等级的成本因子会对工作量(也即开发成本)产生不同的影响.例如,当一个项目的可靠性要求“很高”时,其可靠性因子 RELY 取值为 1.40,就是说,该项目相对于一个其他属性相同但可靠性要求为标称(即取值为 1.00)的项目来说,要多出 40%的工作量.

Table 1 The list of cost factors in intermediate COCOMO 81[8]

表 1 中等 COCOMO 81 模型的成本驱动因子及等级列表[8]

Cost drivers	Rating levels						Remark
	Very low	Low	Nominal	High	Very high	Extra high	
Product attributes							
RELY	0.75	0.88	1.00	1.15	1.40		Required software reliability
DATA		0.94	1.00	1.08	1.16		Database size
CPLX	0.70	0.85	1.00	1.15	1.30	1.65	Product complexity
Computer attributes							
TIME			1.00	1.11	1.30	1.66	Execution time constraints
STOR			1.00	1.06	1.21	1.56	Main storage constraints
VIRT		0.87	1.00	1.15	1.30		Virtual machine volatility
TURN		0.87	1.00	1.07	1.15		Computer turnaround time
Personnel attributes							
ACAP	1.46	1.19	1.00	0.86	0.71		Analyst capability
AEXP	1.29	1.13	1.00	0.91	0.82		Applications experience
PCAP	1.42	1.17	1.00	0.86	0.70		Programming capability
VEXP	1.21	1.10	1.00	0.90			Virtual machine experience
LEXP	1.14	1.07	1.00	0.95			Language experience
Process attributes							
MODP	1.24	1.10	1.00	0.91	0.82		Use of modern programming practices
TOOL	1.24	1.10	1.00	0.91	0.83		Use of software tools
SCED	1.23	1.08	1.00	1.04	1.10		Required development schedule

每个成本驱动因子按照不同等级对项目成本的影响程度,得到不同的工作量乘数,即调整因子 F.

按照中等 COCOMO 81 模型的工作量公式,确定软件项目的模式和因子取值,即可估算工作量,见例 2.

例 2:对于例 1 的系统,若随着项目进展,可以确定其 15 个成本因子的情况,除了以下要求的软件可靠性因子 RELY、计算机周转时间因子 TURN(computer TURNaround time)、要求的开发进度因子 SCED(required development schedule)等 3 个特殊说明外,其余因子均为标称取值 1.00,系数 a=3.2,b=1.05,则其工作量估算为

$$Effort=3.2\times(30)^{1.05}\times(1.15\times0.87\times1.08)=113.8\times(1.15\times0.87\times1.08)=123.0 \text{ 人月}.$$

Cost drivers	Descriptors	Rating levels	Effort multipliers
RELY	High financial loss	High	1.15
TURN	Interactive	Low	0.87
SCED	86% of nominal	Low	1.08

一旦软件的各个模块都已确定,估算者就可以使用详细 COCOMO 81 模型.详细 COCOMO 81 模型的工作量计算公式与中等 COCOMO 81 模型相同,其主要区别在于:(1) 将待估算的软件项目分解为模块、子系统、系统 3 个等级.最低为模块级别,由模块中交付的源指令数(DSI)和 CPLX,PCAP,VEXP,LEXP 这 4 个成本因子来描

述;其次为子系统级别,由其余 11 个成本驱动因子来描述;最高为系统级别,用于采纳主要的整体项目关系,如标称工作量和采用标称项目工作量按阶段分解等.(2) 增加了与开发阶段相关的工作量乘数,它可以准确反映成本驱动因子对工作量阶段分布的影响.例如,“低”级别的应用经验因子(applications experience,简称 AEXP)会增加早期开发的额外工作量,但到后期待团队熟悉应用后,就不会再耗费那么多工作量去做了解背景知识之类的工作了.对每个成本驱动属性,详细 COCOMO 81 模型都有一组表格,每个因子级别都有一个独立的工作量乘数来说明它对各主要开发阶段的影响(以应用经验因子 AEXP 为例,见表 2).表格中对应的 4 个阶段分别为:需求与产品设计 RPD(requirements & product design)、详细设计 DD(detailed design)、代码与单元测试 CUT(code & unit test)和集成与测试 IT(integration & test).

Table 2 An example of different effort multipliers by phase in detailed COCOMO 81[8]

表 2 详细 COCOMO 81 工作量乘数的阶段差异性示例[8]

Cost drivers	Development phase	Rating levels					
		Very low	Low	Nominal	High	Very high	Extra high
AEXP	RPD (requirement and product design)	1.40	1.20	1.00	0.87	0.75	--
	DD (detailed design)	1.30	1.15	1.00	0.90	0.80	--
	CUT (code and unit test)	1.25	1.10	1.00	0.92	0.85	--
	IT (integration and test)	1.25	1.10	1.00	0.92	0.85	--

总而言之,详细 COCOMO 81 模型通过更细粒度的因子影响分析、考虑阶段的区别,使我们能更加细致地理解和掌控项目,有助于更好地控制预算和项目管理.

1.2 COCOMO II

COCOMO 81 以及后来的专用 Ada COCOMO[19],虽然较好地适应了它们所建模的一类软件项目,但是随着软件工程技术的发展,新模型和新方法不断涌现,不但没有好的软件成本和进度估算模型相匹配,甚至因为产品模型、过程模型、属性模型和商业模型等之间发生的模型冲突(model clash)等问题,不断导致项目的超支与失败,COCOMO 81 也显得越来越不够灵活和准确.针对这些问题,Boehm 教授与他的同事们在改进和发展 COCOMO 81 的基础上,于 1995 年提出了 COCOMO II[17].

COCOMO II模型由 3 个子模型组成[17]:(1) 应用组合(application composition)模型,基于对象点(object point)对采用集成计算机辅助软件工程工具快速应用开发的软件项目工作量和进度进行估算,用于项目规划阶段;(2) 早期设计(early design)模型,基于功能点(function point,简称 FP)或可用代码行以及 5 个规模指数因子、7 个工作量乘数因子,选择软件体系结构和操作,用于信息还不足以支持详细的细粒度估算阶段;(3) 后体系结构(post-architecture)模型,顾名思义,发生在软件体系结构完好定义和建立之后,基于源代码行和/或功能点以及 5 个规模指数因子、17 个工作量乘数因子,用于完成顶层设计和获取详细项目信息阶段.

COCOMO II模型所需输入依次为产品规模估算值,产品、过程、平台和人力 4 类项目属性(即成本因子),关于复用、维护与增量开发的参数,组织历史项目数据;模型可以得到的输出有开发和维护的成本与进度估算,以及按阶段、活动、增量开发分布的成本和进度;同时,模型针对不同的组织,使用时可用本组织历史数据进行系统参数校准.在 COCOMO II模型自身参数校准方面,第一次是在 1997 年,对预先的专家决定的模型参数进行了 10%加权平均的多重回归调整,数据集来自商业、航空、政府以及非盈利组织等领域的 83 个实际项目[20].1998 年,又利用贝叶斯分析(Bayesian analysis)来调整专家判定的模型参数,使工作量估算在实际值 30%范围变动的时候从 1997 年的 52%提高到 71%[21,22];2000 年,项目数增加到 161 个,工作量估算在实际值 30%范围变动的时候也进一步增加到 75%[23].贝叶斯分析方法成功地把专家判定技术和回归分析技术结合起来,从而可以使用逻辑上一致的样本数据和专家判定的经验数据,合理地确定模型参数的分布.

与 COCOMO 81 相比,COCOMO II主要作了如下改进[23]:

第一,COCOMO II规模度量在不同开发阶段,可以分别用对象点、功能点或代码行表示.

在应用组合模型估算阶段,通常先解决项目中风险较高的问题,例如用户界面、软件与系统的交互、技术

成熟度等.这一阶段,对于产品最终规模的可用信息过少,COCOMO II采用对象点的方法来估算规模.

在早期设计模型估算阶段,设计者要寻求可选的体系结构,虽然已有信息不足以作细粒度的工作量估算,但是也比之前可以做更多工作.这一阶段,COCOMO II用功能点估算规模,通过体现需求的具体功能,比对象点能够更丰富地描述系统.

在后体系结构模型估算阶段,开发工作已经展开,它可以用功能点或者代码行数来度量规模,并且有更多可用信息用于确定更多成本因子的赋值并进一步估算.

此外,COCOMO II还支持对象点、功能点到代码行的转换.

第二,COCOMO II充分考虑了复用与再工程.

这一点,从规模的计算上就可以看出,COCOMO II工作量公式中 Size 的估算公式如下:

$$Size=\left(1+\frac{REVL}{100}\right)\times(\text{新编}KSLOC+\text{等价}KSLOC) \tag{3}$$

其中,需求演化和变更因子 REVL(requirements evolution and volatility)表达了需求变更的百分比,等价 KSLOC(thousands of source lines of code)是将复用代码和改编代码的有效规模调整后所得的新代码行.

第三,COCOMO II进一步调整和改进了成本因子.

在早期设计/后体系结构的工作量计算等式中,规模的比例驱动因子已经不再是单一的变量,而是由 5 个属性决定:先例性 PREC(precedentedness)、开发灵活性 FLEX(development flexibility)、早期体系结构/风险化解 RESL(architecture/risk resolution)、团队凝聚力 TEAM(team cohesion)、过程成熟度 PMAT(process maturity).每个比例因子的取值原理与工作量乘数相同,先划分等级再对应取值.

COCOMO II中除了对成本因子种类有所增减外,即使名称相同的因子也有不同的等级量表和不同的工作量乘数取值.不过,计算工作量乘数的原理与 COCOMO 81 还是相同的,因此这里不再详细举例.

1.3 COCOMO模型扩展及其系列

随着使用范围的扩大及估算需求的增加,Boehm 教授及其在美国 Southeast California 大学软件工程中心(Center for Software Engineering,简称 CSE)的同事、研究生除了进行上述改进之外,还增加了不少扩展模型以解决其他问题,形成了 COCOMO 模型系列[18],包括:用于支持增量开发中成本估算的 COINCOMO(constructive incremental COCOMO)、基于数据库实现并支持灵活数据分析的 DBA COCOMO(database (access) doing business as COCOMO II)、用于估算软件产品的遗留缺陷并体现质量方面投资回报的 COQUALMO(constructive quality model)、用于估算并跟踪软件依赖性方面投资回报的 iDAVE(information dependability attributed value estimation)、支持对软件产品线的成本估算及投资回报分析的 COPLIMO(constructive product line investment model)、提供在增量快速开发中的工作量按阶段分布的 COPSEMO(constructive phased schedule and effort model)、针对快速应用开发的 CORADMO(constructive rapid application development model)、通过预测新技术中最成本有效(most cost effective)的资源分配来提高生产率的 COPROMO(constructive productivity improvement model)、针对集成 COTS 软件产品所花费工作量估算的 COCOTS(constructive commercial-off-the-shelf cost model)、估算整个系统生命周期中系统工程所花费工作量的 COSYSMO(constructive systems engineering cost model)、用于估算主要系统集成人员在定义和集成软件密集型 SoS(system-of-system)组件中所花费工作量的 COSOSIMO(constructive system of systems integration cost model)等.

这些模型无一例外地继续沿用了 COCOMO II 的“七步法”[17]:(1) 分析历史信息;(2) 执行行为分析;(3) 确定模型形式及标识参数特性;(4) 执行专家判断或者 Delphi 评估;(5) 收集项目数据;(6) 决定 Bayesian 后验更新;(7) 收集更多的数据,细化模型.

因为 COCOMO 模型应用的日益广泛,其他研究者也纷纷提出有针对性的改进或者扩展方案,不断丰富和完善基于算法模型的估算方法.比如,中国科学院软件研究所也提出了带有不确定的估算模型 COCOMO-U[24].

2 非基于算法模型的软件成本估算方法

非基于算法模型的软件成本估算方法是相对于基于算法模型的方法而言的,采用的是除数学算法以外的方法,进行软件成本的估算.比较典型的非基于算法模型的方法有专家估算、类比估算和回归分析.

2.1 专家估算

专家估算,包括从毫无辅助的直觉到有历史数据、过程指引、清单等支持的专家判断[25].这是一个比较宽泛的定义,其主要判断标准是,估算工作由一个被认为是该任务专家的人来控制,并且估算过程的很大一部分是基于不清晰、不可重复的推理过程,也就是“直觉(intuition)”.

对于某一个专家自己所用的估算方法而言,经常使用工作分解结构 WBS(work breakdown structure),通过将项目元素放置到一定的等级划分中来简化预算估计与控制的相关工作.WBS 包括两个层次的分解:一个表示软件产品本身的划分,把软件系统分解为各个功能组件以及其下的各个子模块;另一个表示开发软件所需活动的划分,工作活动分解为需求、设计、编码、测试、文档等大块以及其下的更具体的细分,例如系统设计、数据库设计、详细设计等.WBS 方法帮助估算者确定究竟哪些成本是所要估算的.并且,如果对每个元素的成本设定一个相应的概率,就可以对整个开发的费用得到一个自底向上的全面期望值[26].

由于专家作为个体,存在很多可能的个人偏好,因此通常人们会更信赖多个专家一起得出的结果,并为达成小组一致,引入了 Delphi 方法:首先,每个专家在不与其他人讨论的前提下,先对某个问题给出自己的初步匿名评定.第 1 轮评定的结果收集、整理之后,返回给每个专家进行第 2 轮评定.这次专家们仍面对同一评定对象,所不同的是他们会知道第 1 轮总的匿名评定情况.第 2 轮的结果通常可以把评定结论缩小到一个小范围,得到一个合理的中间范围取值.Delphi 方法在 COCOMO 成本估算中也是被广泛使用的[17].

当仅有的可用信息只能依赖专家意见而非确切的经验数据时,专家方法无疑是解决成本估算问题的最直接的选择.并且,专家可以根据自己的经验对实际项目与经验项目的差异作更细致的发掘,甚至可以洞察未来新技术可能带来的影响.但是,其缺点也很明显,就是专家的个人偏好、经验差异与专业局限性都可能为估算的准确性带来风险.

2.2 类比估算

使用类比(analogy)的方法进行估算是 CBR(case-based reasoning,基于实例推理)的一种形式[27],即通过对一个或多个已完成的项目与新的类似项目的对比来预测当前项目的成本与进度[28].在软件成本估算中,当把当前问题抽象为待估算的项目时,每个实例即指已完成的软件项目.

类比估算要解决的主要问题是:(1) 如何描述实例特征,即如何从相关项目特征中抽取出最具代表性的特征;(2) 通过选取合适的相似度/相异度的表达式,评价相似程度;(3) 如何用相似的项目数据得到最终估算值.特征量的选取是一个决定哪些信息可用的实际问题,通常会征求专家意见以找出那些可以帮助我们确认出最相似实例的特征.当选取的特征不够全面时,所用的解决方法也是使用专家意见[29].

对于度量相似度,目前的研究中常有两种求值方式来度量差距,即不加权的欧式距离(unweighted Euclidean distance)[27]和加权的欧式距离(weighted Euclidean distance)[30].对于相似度函数的定义,有一些不同的形式,但本质上是一致的[31]:

$$distance(P_i,P_j)=\sqrt{\frac{\sum_{k=1}^{n}\delta(P_{ik},P_{jk})}{n}} \tag{4}$$

$$\delta(P_{ik},P_{jk})=\begin{cases}\left(\dfrac{|P_{ik},P_{jk}|}{\max_k-\min_k}\right)^2, & \text{if } k \text{ is continuous}\\ 0, & \text{if } k \text{ is categorical AND } P_{ik}=P_{jk}\\ 1, & \text{if } k \text{ is categorical AND } P_{ik}\neq P_{jk}\end{cases} \tag{5}$$

例 3:一个待估算工作量的项目 P_0 和已完成的项目 P_1,P_2.

Project	Project type	Programming language	Team size	Project size	Effort
P_0	MIS	C	9	180	
P_1	MIS	C	11	200	1 000
P_2	Real time	C	10	175	900

项目间的相似度计算如下:

P_0 vs. P_1	P_0 vs. P_2
$\delta(P_{01},P_{11})=\delta$(MIS,MIS)=0	$\delta(P_{01},P_{21})=\delta$(MIS,Real Time)=1
$\delta(P_{02},P_{12})=\delta$(C,C)=0	$\delta(P_{02},P_{22})=\delta$(C,C)=0
$\delta(P_{03},P_{13})=\delta$(9,11) =[(9−11)/(9−11)]^2=1	$\delta(P_{03},P_{23})=\delta$(9,10) =[(9−10)/(9−11)]^2=0.25
$\delta(P_{04},P_{14})=\delta$(180,200) =[(180−200)/(200−175)]^2=0.64	$\delta(P_{04},P_{24})=\delta$(180,175) =[(180−175)/(200−175)]^2=0.04
distance(P_0,P_1)=(1.64/4)$^{0.5}$=0.64	*distance*(P_0,P_1)=0.57

对于工作量最后估算值的确定,有不同的方法:可以直接取最相似的项目的工作量(对应 P_0 工作量取 1000);也可以取比较相似的几个项目的工作量平均值(对应 P_0 工作量取 1900/2=950);还可以采用某种调整策略,例如用项目的规模作调整参考,对应到例子中,可采用如下调整:*Size*(P_0)/*Size*(P_1)=*Effort*(P_0)/*Effort*(P_1),得到 P_0 工作量为 1000×180/200=900.

类比估算最主要的优点是比较直观,而且能够基于过去实际的项目经验来确定与新的类似项目的具体差异以及可能对成本产生的影响.其主要不足,一是不能适用于早期规模等数据都不确定的情况;二是应用一般集中于已有经验的狭窄领域,不能跨领域应用;三是难以适应新的项目中约束条件、技术、人员等发生重大变化的情况.

2.3 回归分析

在对软件项目进行估算时,通常情况下能得到相关软件组织或软件产品的某些历史数据.充分利用这些历史数据对预测与估算未来状况是很有帮助的.回归分析,就是这样一种相当常用与有效的数据驱动方法[31],包括 CART(分类回归树,classification and regression trees),OSR(最优子集回归,optimized set reduction),Stepwise ANOVA(逐步方差分析,stepwise analysis of variance),OLS(普通最小二乘回归,ordinary least squares regression),RR(稳健回归,robust regression)等.其中,OLS 回归是最传统的回归方法,假定了将一个依赖变量与一个/多个独立变量相关联的一个函数形式[32].

对于 OLS 回归,我们首先要指定一个模型(以表现依赖变量与独立变量之间的关联形式),然后将数据与这个指定的模型相配合,试图使得方差的总和最小.通常,一个使用 OLS 方法的回归函数可表示为[33]

$$y_i=\beta_1+\beta_2x_{i2}+...+\beta_kx_{ik}+u_i \tag{6}$$

这里, $x_{i2}...x_{ik}$ 是对第 i 次观测值的回归变量, $\beta_2...\beta_k$ 是响应系数,β_1 是截距参数,y_i 是对第 i 次观测值的响应变量,u_i 是随机误差.

令 r_i 表示对于第 i 次观测值的实际观测结果 y_i' 与预计结果 y_i 的差值,则 r_i^2 就是平方误差.OLS 方法要做的就是估算出响应系数和截距参数,使得平方误差的总和达到最小化,即 $\min\left(\sum_{i=1}^{n}(y_i'-y_i)^2\right)$.其中,$n$ 表示观测值的总数.

在成本估算中,我们经常遇到需要回归得到的参数存在于非线性等式中,于是,我们可以先将函数转化为线性形式.

例 4:某个数据集保存了 10 个项目的工作量 *Effort* 与规模 *Size* 的信息(见下表第 1~3 列),假设它们之间存在函数关系 $Effort=a\times Size^b$,则可用回归方法得到 a 与 b 的值.

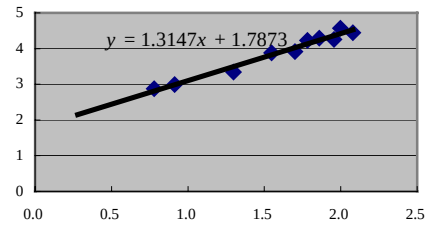
将等式两边取对数,转化为

$$\log(Effort)=\log(a\times Size^b)=\log(a)+b\times\log(size),$$

$$y=A+b\times x,\text{其中}A=\log(a),x=\log(size),y=\log(effort).$$

线性回归得到 $A=\log(a)=1.7873$,$a=10^{1.7873}\approx 61.3$;$b\approx 1.3$,则 $Effort=61.3\times Size^{1.3}$.

Project	Size (kloc)	Effort (Man-hour)	$x=\log(Size)$	$y=\log(Effort)$
P_1	8.20	980.00	0.913 814	2.991 226
P_2	50.10	8 320.00	1.699 838	3.920 123
P_3	90.50	18 000.00	1.956 649	4.255 273
P_4	6.00	748.00	0.778 151	2.873 902
P_5	60.60	17 000.00	1.782 473	4.230 449
P_6	72.30	19 400.00	1.859 138	4.287 802
P_7	35.20	7 530.00	1.546 543	3.876 795
P_8	99.20	36 800.00	1.996 512	4.565 848
P_9	120.00	28 000.00	2.079 181	4.447 158
P_{10}	19.90	2 200.00	1.298 853	3.342 423



OLS 方法简单、易用,Minitab,Splus,SPSS 等多种商业统计工具包都支持该方法[10].其主要缺点在于:(1) 由于每一个观测值对于模型公式有同等的影响,因此,哪怕只有一个差异过大的极端观测值,也会对模型产生不可预计的影响.(2) 由于所需的历史数据依赖于回归模型中的参数个数,当模型中回归变量增多时,需要较多数量的历史数据.通常,回归模型所需的历史数据数必须至少是模型中参数个数的 5 倍.(3) 需要满足对于软件工程数据来说比较严格的假设条件,即回归变量之间不能存在很强的相关性,回归误差的方差恒定.

3 软件成本估算的组合方法

所谓软件成本估算的组合方法,就是在估算技术上明显地综合运用了多种技术与分析方法,这是目前软件成本估算的趋势,也是中和各种估算方法利弊、适应不同估算场合与要求的更好选择.事实上,前面介绍的 COCOMO 模型及其扩展即在算法模型使用过程中结合了专家判断[17];同样,有人认为类比估算因为征求专家意见也是一种组合方法[31].

这里,我们将介绍一种典型的组合方法 COBRA[34]及其后来针对 Web 应用的扩展 Web-COBRA[35].

3.1 COBRA

COBRA(cost estimation, benchmarking, and risk assessment)是一种将算法方式与经验方式相结合的混合估算方法[34],其核心是建立一个由两组件构成的生产率估算模型,如图 3 所示.

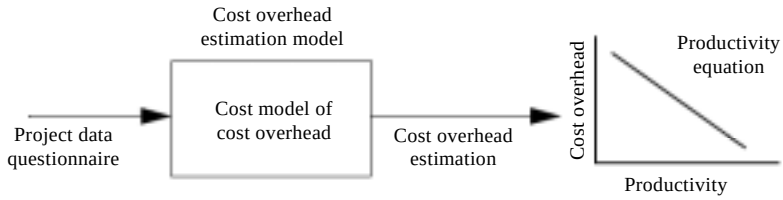


Fig.3 Overview of productivity estimation model[34]

图 3 COBRA 方法的基本模型组合框架[34]

首先,建立因果关系模型(causal model),用来进行成本超支(cost overhead)的估算.

成本超支估算模型的输入是一组描述具体项目特征的数据,从项目相关的非最佳情况来得到成本超支的预测.实际项目往往会与标称项目(即一种假设的理想状态)有很大偏差,成本超支即用于捕获这种偏差的程度.

建立成本超支因果关系模型并估算成本超支的步骤如下:① 确定最重要的成本驱动因子.② 建立定性的

因果关系模型.③ 提出项目数据问卷.④ 量化关系.对各个成本因子的量化就是反映它们在非正常项目中对成本影响的百分比程度,该值称为成本超支乘数.例如,当专家给某个成本因子赋 20%的超额值时,那就意味着,项目成本会达到正常费用的 120%.COBRA 中采用三角形分布来对专家给出的成本超支乘数的最小、最大、最可能的值进行建模.⑤ 建立成本超支估算模型,该模型由三角形分布的总和来表示.⑥ 估算成本超支,用 Monte Carlo 仿真来对三角形分布取样,仿真的结果是成本超支的一个分布,可以选取分布的中值作为项目成本超支的一个估算值.

然后,建立生产率等式(productivity equation),得到对应于一个成本超支值的资金数额或者工作量.

成本超支估算与生产率是强相关的,用过去相似项目的数据可以确定成本超支与生产率之间的关系.它们的关系可以具体表示为

$$P=\beta_0-(\beta_1\times CO) \tag{7}$$

P 为生产率,CO 为成本超支.β_0 是标称项目的生产率,β_1 是 CO 与 P 的斜率.由于所建模的关系是简单的双变量关系,我们可以由很少的历史数据就确定两个β参数的值.

同时,模型假设工作量与规模之间是线性关系,并表示为

$$Effort=\alpha\times Size,\text{其中 }\alpha=\frac{1}{\beta_0-(\beta_1\times CO)} \tag{8}$$

$Size$ 和α的值都由具体项目来确定.依照等式(8),我们可以由系统规模与成本超支得到一个实际项目的工作量.

COBRA 的优点在于:(1) 在仅有少量项目数据时可选用该方法;(2) 在项目估算方面,对可重用的专家知识进行了清晰的建模;(3) 模型是以组织自身经验为基础的,因此更容易被实践者所接受.它的不足主要在于:(1) 需要专家接受面访;(2) 知识抽取比较困难,需要更多培训与经验.

3.2 Web-COBRA

随着 Web 应用的逐渐普及,对 Web 应用程序开发工作量的度量变得也越来越实际,但其相关研究并不多.Ruhe 等人在相关研究的查找中只找到一个针对 Web 应用的估算工作量的模型例子[35],即 Reifer 在 COCOMO II基础上提出的 Web 开发成本估算模型 WebMo[36].Ruhe,Jeffery 和 Wieczorek 试图将 CORBA 应用到 Web 开发的成本估算中,针对 Web 应用的特点,对原始 COBRA 方法进行了改进[35]:(1) 关键成本因子的确定:因 Web 应用的领域特殊,有别于传统的软件开发,不能使用传统的成本影响因子列表,而要用开放的问题对组织内专家进行面访.(2) 定性因果模型的建立:原来的交叉关系难以被理解,现只考虑成本因子与成本之间的直接关系,从而可以得到独立成本因子的最小集合.(3) 因果模型中关系的量化:要保证初始的乘数根据极端值得到修正,将采用个人面访和专家组内讨论的方式,充分理解因子和乘数.(4) 对专家给出的多种答案的结合:因所采样的组织内专家经验水平非常相似,对各个专家意见要平等考虑,在通过专家组内讨论排除极端数据后,取平均值作为某个因子乘数的取值.(5) 规模的估算:因其他规模度量方法并非为 Web 应用特别设计,不够理想,故采用 Web 对象方式,针对 Web 应用进行特定剪裁.

4 软件规模度量

在软件成本估算的研究中,软件规模度量(software sizing)一直都是最受关注的基础话题之一[37−39].要作好软件成本估算,通常首先要明确项目的规模,而估算结果的可信度很大程度上也依赖于规模估算的质量.Boehm 教授还表明,“COCOMO II 模型最重要的输入就是规模”[23].

这里,我们也可以通过两个实际情况看出规模度量与成本估算之间的密切关系.

第一,在算法模型发展中,很多研究结论不约而同地沿用了本文引用的通用公式(2)或者另外一种表达形式[40]:$Effort=A+B\times(Size)^C$,这都表明了软件工作量 PM 或者 $Effort$ 与规模 $Size$ 直接的紧密联系.随着时间的发展,函数中的参数值发生了变化,规模度量形式也发生了改变,但是这种强相关的形式却一直被沿用着.

第二,目前在软件估算方法的研究中又出现了一批更加直接使用规模度量值的方法,例如,用项目规模除以

交付效率来得到工作量估计值[41].其中,交付效率又可由一些总结的数据经验值直接得到,像 Reifer 在 2004 年公布的基于 600 个项目的代码行生产率数据[42],以及 David 咨询组织基于 2000 年~2004 年完成的 8 740 多个项目的功能点生产率数据统计[43].由此可见,当前的成本估算研究仍旧没有脱离与规模的强相关.

目前,主要的软件规模度量方式有代码行、功能点及其扩展方式、对象点以及用例点.

4.1 代码行

代码行(lines of code,简称 LOC)是在软件规模度量中最早使用也是最简单的方法[37],已形成很详尽的样板[44].在用代码行度量规模时,常会被描述为源代码行(source lines of code,简称 SLOC)或者交付源指令(delivered source instruction,简称 DSI),目前成本估算模型通常采用非注释的源代码行[9].

虽然代码行仍是目前普遍采用的一种方式,但它也存在许多问题:

(1) 对代码行没有公认的可接受的标准定义[45].例如,最常见的计算代码时的分歧有空代码行、注释代码行、数据声明、复用的代码,以及包含多条指令的代码行等.在 Jones 的研究中发现,对同一个产品进行代码行计算,不同的计算方式可以带来 5 倍之大的差异[46].

(2) 代码行数量依赖于所用的编程语言和个人的编程风格[39].因此,计算的差异也会影响用多种语言编写的程序规模,进而也很难对不同语言开发的项目的生产率进行直接比较.

(3) 在项目早期,需求不稳定、设计不成熟、实现不确定的情况下很难准确地估算代码量.

(4) 代码行强调编码的工作量,只是项目实现阶段的一部分[40].

4.2 功能点

为克服代码行度量面临的种种问题,有关研究者提出了另一种按照功能特征来度量软件规模的方法[47],即从需求得到系统所要实现功能的功能点,功能点的数量即系统规模.从功能点可以映射到代码行,从而用于成本和进度估算模型,这个转换随着开发语言的不同也会发生变化.

功能点的计算分两步进行[9,48]:第 1 步,按照 5 种基本类型(外部输入 EI、外部输出 EO、内部逻辑文件 ILF、外部接口文件 EIF、外部查询 EQ)归类,得到初始功能点数,分别乘以复杂性权重(根据每个功能类型所含数据元素和引用文件的数量分别归为“简单”、“一般”、“复杂”3 个复杂度等级,并对应不同的复杂性权重,见表 3),5 个加权后的数字相加即得到“未调整功能点”UFP(unadjusted function points)数;第 2 步,根据 14 个基本系统特征(general system characteristic,简称 GSC)确定调整因子 VAF(value adjustment factor),把调整因子应用到未调整功能点,即得到调整的功能点.

Table 3 Weights for UFP with different complexity

表3 未调整功能点复杂度权重对应表

Type of function	Weights for different complexity		
	Simple	Average	Complex
External inputs	3	4	6
External outputs	4	5	7
External queries	3	4	6
Internal logical files	7	10	15
External interface files	5	7	10

国际功能点用户组(International Function Point User Group,简称 IFPUG)[49]提出了 5 种基本类型功能点的统计规则,提供对功能点计算人员的培训,并且根据引入的新开发语言,不断更新转换表.Michigan 大学的工程与计算机科学学院还开发了一个功能点自动计算器[50].

功能性度量方法在软件规模度量中很受重视,因为它们独立于编程语言,并可在早期根据明确功能需求来对最终产品的规模进行估算.在对软件开发环境校准以后,功能性度量的结果为评估开发工作量和软件产品的成本提供了很好的指标[51].但是功能点分析方法在考虑复杂度问题的客观性上也遭到质疑[37].它并不能应用于所有软件类型,比如对逻辑较复杂的实时系统、科学软件,它们的规模度量需要更多信息,如算法、状态转换数目等[52].

4.3 功能点扩展

4.3.1 特征点

1986 年,软件生产率研究所 SPR(Software Productivity Research)的 Jones 对功能点提出了改进,命名为特征点(feature point)[53,54].Jones 认为,实时软件具有很高的算法复杂度而输入和输出则较少,鉴于 FP 对此考虑不足,SPR 的特征点在已有的 5 种功能点参数上增加了一个算法参数,并给出了一系列规则来帮助我们决定哪些算法是可度量的,是有意义的.根据计算步骤的数量和所操作的数据元素的数量,Jones 提出了对算法加权的方法,并设置默认权值为 3.同时,对原来 5 种参数的权值也作了修改.

但是,Whitmire 指出,特征点对算法的定义和所提出的规则不足以帮助我们完成算法的度量,长久以来,算法的度量问题在数学上也一直没有得到有效的解决.在实时软件中,存在大量的内部处理,但是由于受时间的约束,所执行的算法无须很复杂,因此,把算法的复杂度作为影响实时软件功能规模的重要因素也缺乏一定的说服力[52].

4.3.2 3D 功能点

Whitmire 进一步提出了“3D 功能点(3D function points)”[37,52].他认为,软件规模的度量在原有的数据维度基础上,应分为 3 个维度:数据、功能和控制.其中,数据维度由输入、输出、查询、内部数据结构和外部逻辑文件来度量;功能维度则由内部操作的数量和复杂度来度量,这些内部操作将输入数据转换为输出数据,因此也可称为“转换(transformations)”;最后,对控制维度,选取了两个特征值来对它进行度量——状态(state)与状态转化(transition).

3D 功能点为数据操作的复杂性量化引入了一些有意义的指标,但对于确定这些指标仍缺乏详细的定义[37].

4.3.3 全功能点

COSMIC(the Common Software Metrics International Consortium)是 1997 年发起的一个国际学术组织,它集结了一批软件度量方面的学者,力图在现有的软件规模度量研究的基础上,博采众长,找到一种可以更通用、更合理的解决方法.

COSMIC 提出了全功能点(full function point,简称 FFP)[55]概念,是为了把功能性的规模度量推广到实时系统,对 5 个原始的功能点类型增加了一些考虑,例如,通信处理的复杂度和同步性.整个需度量的应用系统被分为管理过程与控制过程.管理过程的目的是支持用户对信息进行管理,尤其是业务和行政管理信息;控制过程是指对另一应用系统或机械装置的行为进行直接或间接控制.FFP 在分析阶段,基于软件需求详述,提出了一个软件模型.根据这个模型,软件被理解为一套依次进行的功能步骤,并由两种类型的功能子过程(数据移动与数据操作)来实现,软件规模即由数据移动的数目来度量.

4.4 对象点

随着越来越多的项目开始使用应用组合方法来进行开发,规模度量方法中又出现了新的一类,即对象点(object point)的度量[56].该技术可以应用于所有类型的软件开发,而不局限于面向对象的开发.利用功能点的基本原理,对象点方法需要考虑那些需投入大工作量的方面来获取规模,例如,服务器数据表的数量、客户数据表的数量、报表(report)和屏幕(screen)中可重用的百分比等等.

要计算对象点,分析人员首先要计算应用中屏幕(screen)、报表(report)和第三代语言组件(3GL component)数量的最可能值.在这个步骤中,对象点的计算与功能点计算类似.然后,依照表 4,每个对象被分级为简单、中等或者困难.表的最底部包含了用于最终计算的权值.

就像功能点一样,将这些加权后的数字相加即得到最后的数值.虽然没有一系列调整因子,但是对象点仍然考虑了重用.如果分析人员可以确定有百分之 r 的构件可以由以前的开发中重用,则新的对象点会相应减少:

$$\text{根据重用调整后的对象点}=\frac{(\text{估算的对象点})\times(100-r)}{100} \tag{9}$$

Table 4 Calculation of object point[23]

表 4 对象点的计算[23]

		Number and source of data tables		
		Total<4(srvr*<2;clnt*<2)	Total 4~8(srvr 2~3;clnt 3~5)	Total >8(srvr>3;clnt>5)
For screens: Number of views contained	<3	Simple	Simple	Medium
	3~7	Simple	Medium	Difficult
	≥8	Medium	Difficult	Difficult
For reports: Number of views contained	0 or 1	Simple	Simple	Medium
	2 or 3	Simple	Medium	Difficult
	≥4	Medium	Difficult	Difficult
*srvr: Number of server data tables;*clnt: Number of client data tables				
Different element types correspond to different complexity-weight		Simple	Medium	Difficult
	Screen	1	2	3
	Report	2	5	8
	3GL component	—	—	10

仅当需求和设计足够详尽并可以得到可用的估算输入时,对象点才有利用价值.有分析认为,完成一个对象点估算的平均时间仅为功能点估算相应时间的 47%[57].然而,这个技术相对较新,因此,关于对象点的长期而成功应用的记录还较为缺乏.

4.5 用例点

随着软件系统更多地采用统一建模语言(unified modeling language,简称 UML)进行开发,继代码行、功能点、对象点之后,出现了基于 UML 的规模度量方法,而基于用例(use case)的估算,即用例点(use case point,简称 UCP)估算方法则是其中具有代表性的一种[58].

总体来说,UCP 方法通过分析用例角色、场景和不同的技术与环境因子,把它们抽象到一个等式中[59].该等式由多个变量组成:未调整用例点(unadjusted use case point,简称 UUCP)、技术复杂度因子(technical complexity factor,简称 TCF)和环境复杂度因子(environment complexity factor,简称 ECF).

UCP的计算等式表示为

$$UCP=UUCP\times TCF\times ECF \tag{10}$$

最终工作量计算等式为

$$Effort=UCP\times ProductivityFactor \tag{11}$$

UCP 方法从 1993 年提出至今,有许多研究者在此基础上作了进一步的应用研究.2001 年,Nageswaran 把测试和项目管理的系数加入 UCP 等式,在其项目的实际应用中得到了更准确的估算值[60].2005 年,Anda 对 UCP 方法步骤进行的修改结合了 COCOMO II 中针对改编软件的计算公式,使得它更适于增量开发的大型软件项目[61].2005 年,Carroll 在其研究中对 UCP 等式加入了一个风险系数,最后估算结果是,95%的项目估算误差在 9%以内[62].

5 软件成本估算方法的评价与应用

软件成本估算一直在软件开发过程中扮演着重要角色,影响着软件工程和软件产业的方方面面.为了方便使用,人们还开发了各种支持工具.比如,基于 COCOMO 模型的 Costar[63],CostXpert[64],True COCOMO[65]等商业软件成本估算软件都拥有广大的用户群.然而,软件成本估算方法、技术和工具种类很多,如何客观地评价和比较,以指导实际应用,一直是人们关心的一个问题.

针对软件成本模型的效用,Boehm 等人曾给出了 10 条评价标准[6]:(1) 定义(definition),即模型是否清楚定义了它所估算的成本以及它所排除的费用;(2) 保真度(fidelity),估算是否接近项目的实际费用;(3) 范围(scope),模型是否覆盖了要估算的软件项目的所有类别;(4) 客观性(objectivity),模型是否避免了将大多数软件成本变量分配给难以校正的主观因素(比如复杂性);(5) 建设性(constructiveness),模型是否告诉用户为何给出这样的估算、是否帮助用户理解了所做的软件工作;(6) 详尽(detail),模型是否容易适用于对多子系统和单元组成的软

件系统的估算、是否给出了准确的阶段与活动的细分结构;(7) 稳定性(stability),输入的微小变化是否产生估算结果输出的微小差异;(8) 易用性(easy of use),模型的输入和选项是否容易理解和说明;(9) 可预测性(prospectiveness),模型是否避免了使用那些直到项目完成才会知道的信息;(10) 精简性(parsimony),模型是否避免了使用非常多余或是对于结果没有明显贡献的因子.

Briand 等人则从更广义和细致的角度,给出了 3 组软件估算评价标准[31]:(1) 模型与估算的标准(model and estimate criteria),包括模型与估算的质量、所需输入变量、估算的完整性、估算类型、校准、可解释性等;(2) 估算方法的标准(estimation method criteria),包括假设、可重复性、复杂度、建模自动化、透明性等;(3) 运用的标准(application criteria),包括运用范围、通用性、全面性、估算的可行性、方法使用的自动化等.

上述方法以及其他类似的评价方法,也都有利有弊并有各自的适用范围.这里,我们从应用的角度,另外给出一种软件开发成本估算方法的三段式评价标准,由 3 个部分、9 项指标组成,如图 4 所示.

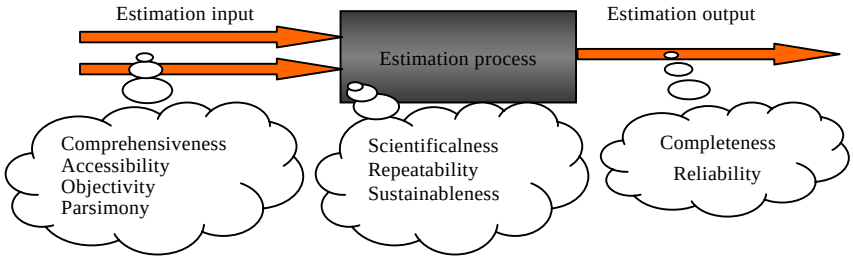


Fig.4 A three phases' evaluation criterion of software cost estimation methods

图 4 软件开发成本估算方法三段式评价标准

- 估算输入

(1) 易理解性(comprehensiveness):用户是否能够清楚了解估算方法所需各项输入的定义与要求.

(2) 信息可得性(accessibility):所需输入的信息是否能在估算阶段实际得到,而不是直到项目完成才能给出.

(3) 客观性(objectivity):能否尽量减少需用户主观判断的信息.

(4) 精简性(parsimony):是否排除了不必要信息输入以及对结果影响不大的因素.

- 估算过程

(5) 科学性(scientificalness):估算所使用的分析方法或数据处理方法是否有理有据,是否对基础假设条件没有异议.

(6) 可重复性(repeatability):估算过程的描述是否足够详尽和清晰,是否可避免在应用时产生主观上的理解误差.

(7) 可持续性(sustainableness):能否通过对相关参数的调整和组织内的校准,使得估算方法能够应对变更性增强、时效性延长等问题.

- 估算输出

(8) 信息全面性(completeness):提供的输出是否可满足用户所需信息范围的要求.

(9) 结果可信度(reliability):得到的估算结果是否能够可信以及如何得到验证.

在实际应用过程中,往往采取的是在成熟工作基础上的组合集成及有针对性的改进方法.下面介绍一个应用实例,对某"政府支持项目申报管理系统"进行规模及成本估算.其中,规模估算方法采用国际功能点用户组 IFPUG 提出的功能点分析方法[49],成本估算则是基于 COCOMO II[17],借鉴了国外软件采购方面的方法,并且基于国内政府数据、国内外工业界数据和国际基准数据等数据,所建立的适合中国政府软件合同定价的方法.

5.1 系统规模估算

根据"政府支持项目申报管理系统"的需求确认书、系统架构图和系统数据表,我们利用所开发的功能点估

算工具进行了规模估算.如图 5 所示为该系统的规模估算界面.

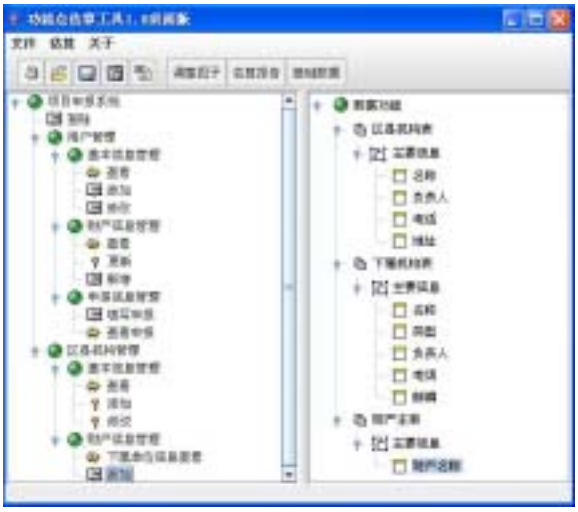


Fig.5 Software sizing in “Declaration and Management System for Government Sponsored Projects”

图 5 “政府支持项目申报管理系统”规模估算

经过规模估算,该系统共有 138 个未调整功能点.估算明细见表 5.

Table 5 Size estimation list of “Declaration and Management System for Government Sponsored Projects”

表 5 “政府支持项目申报管理系统”规模估算明细

Development type	New development			
Estimation phase	Design phase			
Estimation basis	Requirement specification, system architecture & system data table			
Size estimation specification				
Type of function	Complexity	Weights	Number	Amount of FP
External input	Simple	*3	7	21
	Average	*4	0	0
	Complex	*6	1	6
Subtotal				27
External outputs	Simple	*4	1	4
	Average	*5	0	0
	Complex	*7	0	0
Subtotal				4
External queries	Simple	*3	18	54
	Average	*4	1	4
	Complex	*6	0	0
Subtotal				58
Internal logical files	Simple	*7	7	49
	Average	*10	0	0
	Complex	*15	0	0
Subtotal				49
External interface files	Simple	*5	0	0
	Average	*7	0	0
	Complex	*10	0	0
Subtotal				0
Total unadjusted FP	138 UFPs			

5.2 系统成本估算

接下来,我们利用所开发的软件成本估算与预算评估系统对“政府支持项目申报管理系统”进行了成本估算.如图 6 所示为该系统的成本估算界面.该系统包括一个基准库.该基准库中共有 500 多个项目,这些项目来自于 ISBSG[66]、本实验室示范应用企业以及项目委托单位的历史项目.系统基于基准库进行改进的 COCOMO II

参数训练,从而计算工作量和成本.

Fig.6 Cost estimation in "Declaration and Management System for Government Sponsored Projects"

图 6 "政府支持项目申报管理系统"成本估算

计算过程是,首先根据系统规模信息,估算"政府支持项目申报管理系统"所需的工作量,同时根据美国软件生产率研究所的标杆数据[54],确定各类型软件开发人员在软件开发总工作量中所完成的比重.然后根据国内各类型软件开发人员的参考工资率,再估算该系统的人力成本.表 6 列出了工作量估算的明细,表 7 列出了成本估算的明细.

Table 6 Effort estimation list of "Declaration and Management System for Government Sponsored Projects"

表 **6** "政府支持项目申报管理系统"工作量估算明细

Project size information			
Project size	138 UFPs		
Project characteristic			
Platform	Non-Mainframe	Architecture	BS
Project type	MIS	Language	Java
Development flexibility	Nominal	Required reliability	Nominal
Database size	Nominal	Product complexity	Nominal
Developed for reusability	Nominal	Execution time constraint	Nominal
Main storage constraint	Nominal	Platform volatility	Nominal
Effort and schedule estimation result (MM: Man-Month)			
Total effort	3.7 MM	Total schedule	5.4 Months
Requirement effort	0.1 MM	Requirement schedule	0.2 Months
Design effort	0.3 MM	Design schedule	0.4 Months
Coding effort	0.7 MM	Coding schedule	1.0 Months
CM effort	0.0 MM	CM schedule	0.1 Months
Documenting effort	0.1 MM	Documenting schedule	0.2 Months
Test effort	2.0 MM	Test schedule	2.8 Months
PM effort	0.4 MM	PM schedule	0.6 Months

经过估算,"政府支持项目申报管理系统"的规模有 138 个未调整功能点,总工作量为 3.7 人月,总进度为 5.4 月,最可能的总人力成本为 2.83 万元人民币.

由于该项目目前尚处于项目计划阶段,系统所有需求尚未全部确定,因此我们的估算只能基于目前可以确定的需求来进行,并未考虑今后可能会发生的需求变更.根据经验,由于需求变更以及现有需求的进一步细化,一般在后续阶段,如系统设计阶段,未调整功能点数会有 1~2 倍的增加.

Table 7 Cost estimation list of "Declaration and Management System for Government Sponsored Projects"

表 7 "政府支持项目申报管理系统"成本估算明细

Project size information			
Estimated effort	3.7 Man-Month		
Project type	MIS		
Risk information			
Risk issue	Probability	Influence	
Requirement alteration	High	Very high	
Technique	High	High	
Quality	Average	Average	
External affairs	Average	Average	
Labor rate information (Unit: KRMB/Month)			
Requirement	10	Design	10.5
Coding	8	Configuration management	8
Test	8.5	Document editor	3.5
Project manager	11.5	Quality management	8
Cost estimation result (Unit: 10KRMB)			
	Maximum	Most likely value	Minimum
Total cost	10.28	6.99	4.75
Collaboration cost	1.46	0.67	0.00
Tenancy cost	0.72	0.31	0.00
Labor cost	3.75	2.83	1.91
Energy and material cost	0.65	0.28	0.00
Printing cost	0.52	0.25	0.00
Investigation cost	1.05	0.52	0.00
Traveling & communication cost	1.06	0.46	0.00
Equipment cost	1.98	1.03	0.08
Others	1.14	0.63	0.11

6 结 论

有效的软件估算,特别是软件成本估算,一直是软件工程和软件项目管理中最具挑战、最为重要的问题之一.Glass 在其《软件工程中的真相与假象》一书中,总结了 55 个真相和 10 个假象,其中直接与估算有关的就有 7 个真相、1 个假象[4]:

真相 8:造成软件项目失控最普遍的两个原因之一就是软件估算不足.

真相 9:往往在错误的时间,甚至在没有定义需求也即没有理解问题之前,就进行软件估算.

真相 10:往往由错误的人员,即不是由软件开发人员或者项目经理,而是由高层管理或者市场营销,进行软件估算.

真相 11:由错误的人员、在错误的时间做出的软件估算不但往往是错误的,也很少随着项目的进行做必要的调整.

真相 12:虽然软件估算会发生如此错误,但是人们不去关心产生错误的原因,却还在努力按照错误估算的时间进度执行.

真相 13:管理人员和技术人员对软件估算的认识是完全隔绝的,因此也难以达到一致的目标.

真相 14:很少有真正的可行性研究.

假象 6:要估算软件项目开发时间和开发成本,必须首先估算软件项目的代码行数.

无论从这些作者认为应该属于常识的"真相",还是可能造成问题的"假象"来看,都可以轻而易举地得出两个结论:软件估算的价值和意义正越来越被大家所认识到;但是同时,软件估算的一些基础性问题一直也没有得到彻底解决,有的甚至还进一步恶化.尤其是,现有数百种软件估算方法、工具和模型中的多数仅仅能够覆盖软件生命周期中的设计、编码和测试阶段,而且都存在着重大缺陷[41].不过,这也并不意味着人们就束手无策和无所作为.只要坚持按照这些估算方法去做,还是可以获得比以往更好的结果的[41].

近年来,Boehm 等人试图用"价值(value)"的理念,把包括软件成本估算在内的影响软件开发的诸多要素统一在"基于价值的软件工程(value-based software engineering)"框架下,从而获得更多的回报[67].然而,估算因为要

预言未来而带来 3 个本质特征[68],即省略、不确定和变化.因此,很多情况下,在准备进行估算的时候,往往不但不了解产品需求、设计细节和过程活动,甚至连谁来执行这些活动都不知道.这就使得软件成本估算变得格外困难,在不断得到改进的同时,又不断受到新的挑战.

关于软件成本估算方法的未来发展趋势,我们认为在以下几个方面特别值得关注:

(1) 估算模型:不断会有新的软件成本估算模型和估算方法被提出,既会是对以往模型的有益补充,也可能带来新的不适应.实际上,没有一种模型或方法明显优于其他模型或方法.如何根据具体应用背景与条件,在软件开发过程的不同阶段,评价和选择适合的软件成本估算模型与方法,特别是这些模型与方法的适当组合应用及兼容性问题,将一直是软件开发过程管理的首要问题.

(2) 估算演进:期望一次估算就非常准确是不现实的.由于影响成本的软件规模、项目风险和技术复杂度等信息,随着软件生命周期的演进而逐渐确定或始终动态变化,成本估算模型需要能适应并体现软件开发动态演进的特征,特别需要在软件开发早期大量信息不确定的情况下进行合理估算并评估其可靠程度.同时,软件开发技术的进步、软件开发工具的普及、软件应用场合的拓宽以及以互联网为代表的软硬件环境的发展、以数据仓库为代表的数据库技术的升级和以开源社区为代表的软件开发模式的变化,也会给未来的软件成本估算方法带来巨大的挑战.

(3) 估算应用:在使用软件成本估算模型进行估算之前,需要基于历史数据和开发环境对模型进行校准.在组织拥有足够的历史数据时,基于本组织历史数据的估算是相对准确的估算.但是,当组织缺乏历史数据而又要选择和比较已有模型方法时,目前缺乏公认的公用数据集和评价标准的现状,使得组织很难作出正确的抉择.如何充分利用所有可能的已有信息、简单而准确地预见未来信息,无疑会对软件成本估算方法的下一步发展方向产生重要影响.

(4) 估算内容:软件成本估算内容不断扩展.随着“软件变服务”的发展趋势,从目前大家更多关注的直接开发成本,会逐步扩展到包括安装、培训、服务、设备、人员、维护等多种费用形式,甚至在某些场合下直接开发成本可以忽略不计.因此,如何把握整体成本,需要更系统化的估算视角.估算内容、规模度量方式以及规模与软件成本的关联等也会发生相应的变化.

(5) 工具支持:目前的估算模型相对还都比较复杂,尤其是在模型校准部分,影响了估算方法的有效使用,而软件成本估算的工具支持还有待于改进.此外,估算结果会影响到项目计划、过程控制等各个方面,如何与其他成熟软件过程工具(集)有效集成,以及如何保障用户的投资回报,更是一个需要考虑的重要方面.

(6) 人为因素:随着软件项目规模、复杂度的增加,人为因素在软件成本估算中的影响会越来越大.一方面,很多软件成本估算方法的输入值通常都是基于主观评估的;另一方面,这些方法也都需要大量人为输入以覆盖不同类型项目的情况.如何合理地把这一影响因素综合考虑进来,同时发挥专家意见和项目数据的积极调节作用,可能既涉及到与其他学科的交叉,也不可避免地需要进一步加强人为因素的管理与控制.

References:

[1] The Standish Group. CHAOS Report, 1995. http://www.standishgroup.com

[2] The Standish Group. 2004 the 3rd Quarter Research Report, 2004. http://www.standishgroup.com

[3] Moløkken K, Jørgensen M. A review of software surveys on software effort estimation. In: Andrews AKA, ed. Proc. of the Int'l Symp. on Empirical Software Engineering. Rome: IEEE Computer Society, 2003. 223–230.

[4] Glass RL. Facts and Fallacies of Software Engineering. Boston: Addison-Wesley, 2003.

[5] Boehm BW. Understanding and controlling software costs. IEEE Trans. on Software Engineering, 1988,14(10):1462–1477.

[6] Boehm BW, Valerdi R. Achievements and challenges in software resource estimation. Technical Report, No.USC-CSE-2005-513, 2005. http://sunset.usc.edu/publications/TECHRPTS/2005/usccse2005-513/usccse2005-513.pdf

[7] BESTWeb System. 2004. http://www.simula.no/BESTweb

[8] Boehm BW. Software Engineering Economics. Englewood Cliffs: Prentice Hall, 1981.

[9] Pfleeger SL. Software Cost Estimation and Sizing Methods: Issues, and Guidelines. Santa Monica: Rand Corp., 2005.

[10] Boehm BW, Abts C, Chulani S. Software development cost estimation approaches—A survey. Annals of Software Engineering, 2000,10(1-4):177−205.

[11] Kitchenham BA, Taylor NR. Software project development cost estimation. The Journal of Systems and Software, 1985,5(4):267−278.

[12] Kemerer CF. An empirical validation of software cost estimation models. Communications of the ACM, 1987,30(5):417−429.

[13] Navlakha JK. Choosing a software cost estimation model for your organization: A case study. Information and Management, 1990, 18(5):255−261.

[14] Srinivasan K, Fisher D. Machine learning approaches to estimating software development effort. IEEE Trans. on Software Engineering, 1995,21(2):126−137.

[15] Shepperd M, Cartwright M. Predicting with sparse data. IEEE Trans. on Software Engineering, 2001,27(11):987−998.

[16] McGibbon T. Modern cost estimation tools and modern empirical cost and schedule estimation tools. 1997. http://www.dacs.dtic.mil/techs/estimation/title.shtml

[17] Boehm BW, Clark B, Horowitz E, Westland C. Cost models for future software life cycle processes: COCOMO 2.0. Annals of Software Engineering, 1995,1:57−94.

[18] Boehm BW, Valerdi R, Lane J, Brown A. COCOMO suite methodology and evolution. CrossTalk: The Journal of Defense Software Engineering, 2005,18(4):20−25.

[19] Boehm BW, Royce W. Ada COCOMO and the Ada process model. In: Proc. of the 5th Int'l COCOMO User's Group Meeting. Pittsburgh, 1989.

[20] Boehm BW, Clark B, Chulani S. Calibration results of COCOMO II 1997. In: Proc. of the 22nd Software Engineering Workshop. 1997.

[21] Chulani S, Boehm BW, Steece B. Calibrating software cost models using Bayesian analysis. Technical Report, USC-CSE-98-508, 1998.

[22] Chulani S, Boehm BW, Steece B. Bayesian Analysis of empirical software engineering cost models. IEEE Trans. on Software Engineering, 1999,25(4):573−583.

[23] Boehm BW, Abts C, Brown AW, Chulani S, Clark BK, Horowitz E, Madachy R, Reifer D, Steece B. Software Cost Estimation with COCOMO II. New York: Prentice Hall, 2000.

[24] Yang D, Wan Y, Tang Z, Wu S, He M, Li M. COCOMO-U: An extension of COCOMO II for cost estimation with uncertainty. In: Wang Q, *et al*., eds. Software Process Change, SPW/ProSim 2006. LNCS 3966, Berlin, Heidelberg: Springer-Verlag, 2006. 132−141.

[25] Jørgensen M. A review of studies on expert estimation of software development effort. Journal of Systems & Software, 2004, 70(1-2):37−60.

[26] Baird B. Managerial Decisions Under Uncertainty. New York: John Wiley & Sons, 1989.

[27] Shepperd M, Schofield C. Estimating software project effort using analogies. IEEE Trans. on Software Engineering, 1997,23(12): 736−743.

[28] Delany SJ, Cunningham P, Wilke W. The limits of CBR in software project estimation. In: Gierl L, Lenz M, eds. Proc. of the 6th German Workshop on Case-Based-Reasoning. Berlin: Springer-Verlag, 1998.

[29] Shepperd M, Schofield C, Kitchenham B. Effort estimation using analogy. In: Proc.of the 18th Int'l Conf. on Software Engineering. IEEE CS Press, 1996. 170−178.

[30] Cost S, Salzberg S. A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning, 1993,10(1): 57−78.

[31] Briand LC, Wieczorek I. Resource estimation in software engineering. In: Marcinak JJ, ed. Encyclopedia of Software Engineering. New York: John Wiley & Sons, 2002. 1160−1196.

[32] Berry WD, Feldman S. Multiple Regression in Practice. Newbury Park: Sage Publications, 1985.

[33] Gujarati DN. Basic Econometrics. 3rd ed., New York: McGraw Hill, 1995.

[34] Briand LC, Emam K, Bomarius F. COBRA: A hybrid method for software cost estimation, benchmarking and risk assessment. In: Proc. of the 20th Int'l Conf. on Software Engineering. IEEE CS Press, 1998. 390−399.

[35] Ruhe M, Jeffery R, Wieczorek I. Cost estimation for Web application. In: Proc. of the 25th Int'l Conf. on Software Engineering. IEEE CS Press, 2003. 270−279.

[36] Reifer D. Web-Development: Estimating quick-time-to-market software. IEEE Software, 2000,17(8):57−64.

[37] Tran-Cao D. Measuring software functional size: towards an effective measurement of complexity. In: Proc. of Int. Conf. on Software Maintenance. 2002. 370−376.

[38] Bielak J. Improving size estimates using historical data. IEEE Software, 2000,17(6):27−35.

[39] Kemerer CF. Improving the reliability of function point measurement: An empirical study. IEEE Trans. on Software Engineering, 1992,18(11):1011−1024.

[40] Matson JE, Barrett BE, Mellichamp JM. Software development cost estimation using function points. IEEE Trans. on Software Engineering, 1994,20(4):275−287.

[41] Laird LM, Brennan MC. Software Measurement and Estimation: A Practical Approach. Wiley-IEEE Computer Society Press, 2006. 79−88.

[42] Reifer D. Industry software cost, quality, and productivity benchmarks. The DOD Software Tech. News, 2004. http://www.software technews.com/stn7-2/reifer.html

[43] David Consulting Group. 2005. http://www.davidconsultinggroup.com

[44] Park R. Software size measurement: A framework for counting source statements. Technical Report, CMU/SEI-92-TR-20, Pittsburgh: Software Engineering Institute, 1992.

[45] Fenton NE, Pfleeger SL. Software Metrics: A Rigorous & Practical Approach. 2nd ed., Boston: PSP Publishing Company, 1997.

[46] Jones TC. Programming Productivity. New York: McGraw-Hill, 1986.

[47] Albrecht AJ, Gaffney JE. Software function, source lines of code and development effort prediction: A software science validation. IEEE Trans. on Software Engineering, 1983,9(6):639−648.

[48] International Function Point Users Group. Function Point Counting Practices Manual, Release 4.1.1, Princeton Junction: International Function Point User's Group, 2001.

[49] http://www.ifpug.org

[50] http://www.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/artan/functionpoints.htm

[51] Low CG, Jeffery R. Function points in the estimation and evaluation of software process. IEEE Trans. on Software Engineering, 1990,16(1):64−71.

[52] Abran A, Desharnais JM, Maya M, St-Pierre D, Bourque P. Design of a functional size measurement for real-time software. Research Report, No. 13-23, UQAM, 1998.

[53] Jones C. Applied Software Measurement—Assuring Productivity and Quality. New York: McGraw-Hill Inc., 1991.

[54] http://www.spr.com

[55] Vogelezang F. COSMIC full function points the nest generation of functional sizing. In: Software Measurement European Forum—SMEF 2005. 2005.

[56] Banker RD, Kauffman RJ, Kumar R. An empirical test of object-based output measurement metrics in a computer aided software engineering environment. Journal of Management Information Systems, 1991-1992,8(3):127−150.

[57] Kauffman RJ, Kumar R. Modeling estimation expertise in object-based case environments. Stern School of Business Report, New York: New York University, 1993.

[58] Karner G. Resource estimation for objectory projects. Objective Systems SF AB, 1993.

[59] Clemmons RK. Project estimation with use case points. CrossTalk: The Journal of Defense Software Engineering, 2006,(2):18−22.

[60] Nageswaran S. Test effort estimation using use case points. In: Proc. of the 14th Int'l Internet & Software Quality Week 2001. San Francisco, 2001. http://www.cognizant. com/html/content/cogcommunity/test_effort_estimation.pdf

[61] Mohagheghi P, Anda B, Conradi R. Effort estimation of use cases for incremental large-scale software development. In: Proc of the 27th Int'l Conf. on Software Engineering. St Louis, 2005. 303−311.

[62] Carroll ER. Estimating software based on use case points. In: Proc. of the Conf. on Object Oriented Programming Systems Languages and Applications. New York: ACM Press, 2005. 257−265.

[63] http://www.softstarsystems.com

[64] http://www.costxpert.com

[65] http://www.pricesystems.com

[66] http://www.isbsg.org

[67] Biffl S, Aurum A, Boehm B, Erdogmus H, Grunbacher P. Value-Based Software Engineeringl. New York: Springer-Verlag, 2006.

[68] Stutzke RD. Estimating Software-Intensive Systems: Project, Products, and Processes. Boston: Addison-Wesley, 2005.

李明树(1966－),男,吉林德惠人,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件工程方法学,软件过程技术,需求工程,软件工程经济学.

何梅(1982－),女,博士生,主要研究领域为软件工程经济学,软件过程技术.

杨达(1979－),男,博士生,主要研究领域为软件工程经济学,软件过程技术,需求工程.

舒风笛(1976－),女,博士,助理研究员,主要研究领域为需求工程,软件估算.

王青(1964－),女,博士,研究员,博士生导师,主要研究领域为软件过程技术.

第 24 届全国数据库学术会议(NDBC 2007)

征文通知

2007 年 10 月 19–21 日，海南 海口
http://www.hainu.edu.cn/ndbc2007/

第 24 届全国数据库学术会议（NDBC 2007）将于 2007 年 10 月 19 日–21 日在海南省海口市举行。会议由中国计算机学会数据库专业委员会主办，海南大学承办，海南省计算机学会、成都理工大学、海南师范大学、琼州大学、海南软件学院协办。

重要日期: 提交截止时间: 2007 年 4 月 20 日
录用通知时间: 2007 年 6 月上旬
排版稿件截止时间: 2007 年 6 月下旬

会议网站: http://www.hainu.edu.cn/ndbc200

会务组联系方式: 电子邮件 ndbc2007@hainu.edu.cn

电话: 0898-66288382 (雷老师) 0898-66279141 (钟老师) 13006002229 (靳老师)

传真: 0898-66288382

通讯地址: 海南省海口市海南大学信息科学技术学院(邮编：570228)