

软件开发项目工作量估算技术的比较研究

王求真

(浙江大学 管理科学与工程系, 浙江 杭州 310027)

[摘 要] 软件开发项目的工作量估算技术一般分为三类: 基于专家判断的技术、基于算法模型的技术和面向学习的技术。不同的估算技术各有自己的优点和局限性, 没有一种估算技术能适用于所有开发环境, 并且软件开发方法和技术的更新速度也对所有这些估算技术提出了挑战。软件组织应根据具体的项目特征和可获得的信息来选择合适的估算技术, 并针对当前项目情况对使用的估算模型加以调整, 依据不同技术的特点组合不同的估算技术进行估算, 以提高估算准确性。在估算项目工作量时要充分考虑到项目前期阶段的工作量, 并建立本组织的软件项目库。

[关键词] 软件开发项目; 规模估算; 工作量估算; 基于专家判断的技术; 基于算法模型的技术; 面向学习的技术

[中图分类号] C931 **[文献标志码]** A **[文章编号]** 1008 - 942X(2005)04 - 0090 - 08

软件开发项目超支、延期和不能满足用户的需求, 已成为软件开发组织普遍存在的问题。根据 Standish Group 最近对美国公司的一项调查, 只有 28% 的软件开发项目在预计的时间和预算内完成^[1]。造成软件项目超支和延期的原因很多, 不准确的估算就是其中一个重要原因。如果在做项目计划时, 对项目的规模、工作量、成本和进度的估算不准确、不现实, 那么即使项目管理、控制得再好, 也很难达到预期的目标。软件项目估算不准确会导致一系列问题, 如项目经理将工作量和成本估算得太低, 可能会导致开发后期资源不足, 进而影响软件产品的质量; 如项目经理将工作量和成本估算得太高, 则会影响资源妥善分配, 甚至使管理层放弃开发有潜在收益的系统, 丧失潜在的获利机会。因此, 无论低估还是高估都会产生不利的影响。软件项目与其他工程项目相比, 有其特殊性, 主要表现在软件开发是处在动态开发环境下的需求更多变和人力密集型、知识密集型这样一类更具灵活性和不确定性的活动, 因此, 要在项目早期做到精确地估算工作量是不可能的。但是, 与单凭经验猜测相比, 能够考虑影响工作量估计的各种因素如系统规模和复杂性、开发人员的能力和硬件的限制、软件工具和方法的应用等, 采用恰当的估算技术可以大大提高估算的准确度。

目前, 用于软件开发项目的工作量和成本估算的方法有很多, 大多数估算工作量的方法需要估算软件的规模, 一般认为工作量估算包含两个阶段: 产品规模估算和基于规模的工作量估算。本文首先分析比较了不同的软件规模估算方法及在面向对象技术下的应用, 然后综合分析了现有主要的工作量估算技术, 对各种估算技术的优点和局限性进行了比较研究, 以期软件开发组织如何选择和有效地应用估算技术以提高估算准确度提供借鉴。

[收稿日期] 2005 - 03 - 03

[本刊网址·在线杂志] <http://www.journals.zju.edu.cn/soc>

[项目基金] 国家自然科学基金资助项目(70472056)

[作者简介] 王求真(1971 -), 女, 浙江湖州人, 浙江大学管理学院管理科学与工程系教师, 博士研究生, 主要从事软件项目管理等方面的研究。

一、软件规模的估算方法

(一)传统的软件规模估算方法

常用的软件规模的度量是用代码行(LOC)和功能点(FP)来测量。LOC是所有软件开发项目的“生成品”,很容易进行计算;但 LOC 度量依赖于程序设计语言,并且在项目开发初期由于信息不全,较难估算。功能点分析是一个两阶段过程:第一阶段,在系统输入、输出、查询、内部文件和外部文件数估计值的基础上计算未调整的功能点(UFP),这些输入、输出、查询、外部文件、内部文件按简单、平均或复杂来评价并分配给相应的权重;第二阶段,通过 14 个一般系统特征对 UFP 进行调整,以校正不同系统开发之间的技术复杂度差异^{[2](pp.283-284)}。FP 方法从用户的角度来测量系统功能,因此,能在系统开发生命周期的早期阶段预测项目的规模和工作量,并且它与程序设计语言无关,这是该方法的主要优点。但是功能点度量需要某种“人的技巧”,它们的计算基于主观的评价。另外,FP 最初的提出主要是应用于管理信息系统,因此,它的应用能力受到限制。针对这一不足,出现了一些 FP 的扩展模型,如 Feature Points 被设计用于 MIS 应用和其他类型的软件,如实时软件或嵌入式软件;Full Function Points(FFP)被提出用来测量实时系统的规模等。

(二)面向对象技术下的软件规模估算

面向对象(Object-Oriented,简称 OO)技术最先应用在程序设计,出现了各种面向对象程序设计语言(如 Smalltalk, C++ 等),后来 OO 思想应用在整个系统开发过程,包括应用的分析和设计。传统的结构化方法是面向功能的,分别为业务流程和数据建模,而 OO 方法将数据和过程集合在统一的对象中。OO 方法的基本思想是客观世界是由对象及其行为所构成的,因此,对现实世界建模的最好方法是用对象。随着 OO 方法逐渐成为软件开发的主流技术,一些研究人员对 OO 系统的估算技术进行了研究,目前,这些研究主要集中在软件规模的估算。现有的针对过程范式的软件规模度量(LOC 和 FP)被认为是不适合捕获特定的面向对象的特性,如类、继承性、封装性和消息传递等特性。尽管 FP 法被认为独立于开发方法,但有证据表明把 FP 法应用到面向对象系统时相当勉强。比如, Kermer 和 Porter 在如何改进 FP 度量的可靠性的实证研究中得出结论:“事件驱动的面向对象系统的出现……可能需要重新定义 FP 或开发识别系统规模的新的度量”^{[3](pp.71-72)}。

由于 FP 方法使用的普遍性,近几年提出的用于面向对象系统的规模估算基本上都是基于对传统的 FP 方法的修改。G.Caldiera 等提出了一种面向对象软件开发项目的规模估算方法,该方法的关键是将功能点概念映射到面向对象概念。在传统开发中,计算功能点用到的主要概念是逻辑文件和处理(输入、输出和查询)。在面向对象系统中,核心概念是“对象”。G.Caldiera 等提出把功能点映射到面向对象软件的主要类推是把逻辑文件和处理与类及方法联系起来,他们把逻辑文件映射为类,处理映射为方法^{[4](p.168)}。对象点是对 FP 的另一种调整,它被应用在 COCOMOII 模型中。对象点的计算与 FP 很类似,不同的是其计算过程的基础是对象,不过这些对象不是直接对应于 OO 方法中的“对象”,而是指屏幕、报告或 3GL 模块^{[5](p.55)}。Fetcke et al. 基于将 Jacobson OOSE 方法中的用例模型映射到功能点概念的规则提出一种 OO 系统的规模估算方法^{[6](p.55)}。还有其他一些基于 FP 的面向对象系统的规模度量方法,如预测对象点(POPs)和面向对象功能点(OFPs)^{[5](pp.54-55)}。

二、主要的软件开发项目工作量估算技术

目前用于估算软件开发项目的工作量的估算技术有许多,一般可以分为三类:基于专家判断的

估算技术,基于算法模型的估算技术和面向学习的估算技术。基于算法模型和基于专家判断的估算技术是发展较成熟且应用较普遍的估算技术,面向学习的估算技术是一种基于人工智能技术的估算方法,其应用普遍性和成熟性不如前两种估算技术,但该估算技术目前已成为软件项目估算研究领域比较关注的一个方面。

(一)基于专家判断的估算技术

基于专家判断的评估技术简称为专家评估技术,实际中被广泛使用。该类技术包含了许多估算方法,如专家磋商、直觉和经验等,其共性是“直觉”过程构成了估算的主要部分。专家评估技术在缺少量化的、历史数据的情况下非常有用。但是这种方法主要取决于评估人的经验,带有很大主观性,有时会带来较大的估算偏差,并且该方法导出估算值的机理不清晰,是不可重复的,因此软件组织很难清晰定义这种估算过程,不能获得组织学习的效应。Delphi 法是目前最流行的专家评估技术,该方法可以在一定程度上减轻由专家“专”的程度及对项目的理解程度所带来的偏差。该方法的具体步骤是:(1)协调人向各专家提供项目规格和估计表;(2)协调人召集小组会同各专家讨论与规模、工作量相关的因素;(3)各专家匿名填写迭代表格;(4)协调人整理出一个估计总结,以迭代表格的形式返回专家;(5)协调人召集小组会,讨论较大的估计差异;(6)专家复查估计总结并在迭代表格上提交另一个匿名估计;(7)重复(4)一(6),直到达到一个最低和最高估计的一致。

(二)基于算法模型的估算技术

这一类估算方法用一个公式来表达工作量(或成本)和一个或多个项目特征(也称成本驱动因子)之间的关系,是参数化的模型。算法模型包括经验估算模型(即有固定的公式)和回归模型。大多数算法模型使用系统规模作为估算过程的关键输入,因此,准确的规模估算非常重要。基于算法模型的估算方法有很多,如 SLIM、COCOMO、Checkpoint、SEER-SEM、Estimacs、PRICE-S 等,其中 COCOMO 是被广泛接受和使用的一种公开发表的估算模型。

Barry Boehm 在 1981 年提出 COCOMO(Constructive cost model)。COCOMO 模型按开发环境(项目规模、技术复杂度、开发经验等)把软件开发项目的类型分为组织型、嵌入型和半独立型三种。不同类型在应用 COCOMO 时,方程中的参数是不同的。COCOMO(81)是一个层次模型,按其估算的详细程度包括三个层次,即基本 COCOMO、中级 COCOMO 和高级 COCOMO。

(1)基本 COCOMO 的公式: $MM = a(KDSI)^b$,其中:MM 是用“人一月”表示的工作量,KDSI 是软件功能单元的代码行数(以千行为单位),a、b 是参数,它们因具体的项目类型而不同。

(2)中级 COCOMO 是在基本 COCOMO 的基础上,再用涉及产品、硬件、人员、项目等方面的影响因素调整工作量的估算。当估算人员不仅知道代码行的数目,而且对项目有较好的理解时,可采用该模型: $MM = a(KDSI)^b P(EM)$,其中:P(EM)是 15 个成本驱动因子的工作量乘数总效应。

(3)高级 COCOMO 包括了前两种模型的所有特征,不过它对项目开发的每个阶段(分析/设计,编码,测试)的每个成本驱动因子进行评价。因此,只有当有足够详细的每个项目阶段的信息时,可以采用该模型。

随着面向对象编程和 CASE 工具等新的软件开发方法的应用,1994 年,Boehm 重新调整和扩展原有模型,发表了 COCOMOII。COCOMOII 适用于面向对象开发和螺旋式生命周期,与早期版本相比,COCOMOII 的主要扩展是:增大的影响因素(即成本驱动因子)集合;用 5 个规模因子计算基本 COCOMO 公式中指数 b,代替了原来按项目类型的不同使用固定指数的方法;采用不同的方法来分析代码行;引入了功能点,功能点被转化为 SLOC,进而转化为工作量^{[7](p.590)}。

(三)面向学习的估算技术

面向学习的估算技术是建立在人工智能技术基础上的估算技术,主要的估算方法有人工神经网络(ANN,Artificial Netural Network)和基于案例推导(CBR,Case-based reasoning)等。

近几年,国外学者开始对 ANN 在软件开发中的应用进行了较多研究。神经网络中的处理单元叫做神经元,神经元之间的组织方式称为网络拓扑,目前最流行的拓扑是一种前向(feed-forward)网络结构(如图 1)。网络设计成输入层、输出层和中间若干隐蔽层,输入层有特定的输入点,如复杂度、实现语言、项目规模、技能水平等,输出层输出预测值如工作量,每个神经元的输出与下一层的所有神经元都建立连接。数据通过输入层流入网络,穿过一个或多个隐蔽层,最后通过输出层流出网络。ANN 采用一种学习方法导出一个预测模型,网络建立后,用一组历史项目数据(训练样本)训练网络,不断调整网络中各个输入有关的权重,直到预测值与实际值之间的误差在规定范围内。常用的训练方法是逆向传播方法(back propagation),其步骤如下:(1)建立一组训练样本;(2)数据通过输入层进入网络;(3)网络中的每个神经元处理输入数据,合成值稳定的经过网络各层次,直到输出层产生结果;(4)网络的输出与实际输出进行比较,根据两者的差异从输出层经过中间层到输入层逆向调整网络中所有连接的权重,直到产生正确的输出。重复(2)一(4),直到针对每一个输入案例网络都产生正确的输出时,训练过程就结束了,各层输入的权重确定,网络处于稳定状态可以被用来预测,通过代入具体项目的相应的输入值来预测一个新的软件项目的开发工作量^{[8](pp.912-913)}。

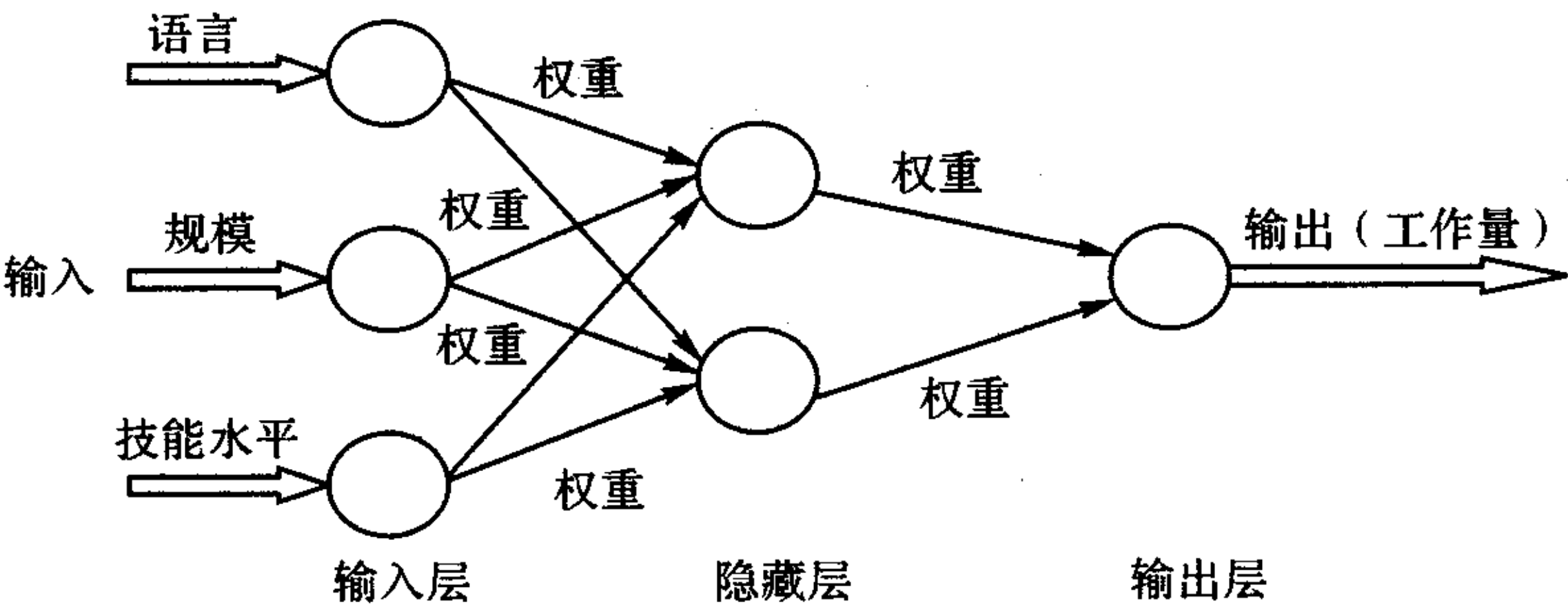


图 1 一个神经网络的结构

CBR 是一种问题解决技术,通过修改用来解决老问题的解决方案来解决新问题。CBR 的基本思想是识别出与新项目类似的案例(源项目),然后调整这些案例,使他们适合当前问题(新项目)的参数。调整是基于已存储的案例和当前问题之间的差异,如原始案例可能是关于由一个缺乏经验的编程队伍所开发的系统,而当前系统开发人员可能对开发环境非常熟悉。因此,开发应该加速,并且相应地调整估算值。CBR 的优点是它能在以前案例的基础上调整决策^{[2](p.282)}。

三、软件开发项目工作量估算技术的比较

(一)软件项目估算技术在国内外企业中的应用

国外的一些研究人员对不同软件项目估算技术在不同国家软件开发组织中的应用情况进行了调查,Kjetil Molkken 等(2004)对 1984 年至 2002 年期间进行的关于软件项目工作量估算的各种调查进行了概括总结,他们把软件项目估算技术分为三类:基于专家的估算方法、基于模型的估算方法和其他方法。研究指出,由于不同调查所采用的估算技术分类及被调查人对分类的解释不尽相同,

要严格地比较这些调查结果是困难的,不过从这些调查还是看到了一个明显的共同之处,即基于专家判断的估算技术是至今为止使用最普遍的软件项目估算技术^{[9](p.915)}。

由于缺乏历史数据的存储,目前国内大多数项目的软件工作量估算主要依靠 WBS(工作分解结构)法和经验判断。如国内一家中等规模的主要从事通讯行业软件开发的软件公司,其估计步骤是根据需求规格说明书的内容,让各开发人员独立估算各任务及具体模块的工作量,然后汇总,最多估计三次,总的工作量的计算公式是:工作量(人×天)=代码工作量+计划文档工作量+需求规模工作量+设计工作量+测试工作量。在整个估算过程中,假设参与评估人均有相同的此系统方面的工作经验。

(二)软件项目工作量估算技术的特点及局限性

从估算技术在国内外企业中的应用现状来看,基于专家判断的估算技术是使用最普遍的,这可能是因其所具有的易用性和灵活性,在缺少量化的、历史数据的情况下该技术非常有用。但是这种方法依赖专家的经验、背景和成熟度来猜测一项任务或整个项目需要多长时间完成或成本多少,因此,专家“专”的程度及对项目的理解程度是估算工作中的难点,也是这种估算技术的局限性,尤其当进行新产品开发,企业不具有开发类似项目的经验时,该方法很难实施。另外,该技术如何导出估算值的方法不清晰,是不可重复的,因此,软件开发组织很难清晰定义这种估算过程,不能获得组织学习的效应。

基于算法模型的估算技术与基于专家判断和面向学习的估算技术相比,其主要优点是让用户看到了一个模型如何结论的过程。虽然文献关于这一类估算技术的研究很多,但该技术在实际应用中使用得并不多,一方面是因为需要对估算模型理解透彻,另外也缺乏证据表明这类方法比基于专家的估算技术估算更准确。一些研究人员(如 Kemerer,1987;Ferens and Gurner,1992 等)对不同估算模型的精确度进行验证,发现这些模型的估算误差都很大,没有一个模型被证明能有效地、一致地预测软件开发工作量^{[10][11]}。基于模型的估算技术存在的主要问题是:(1)软件的规模估算问题。这些模型都假设能够在生命周期的早期估算出系统规模,而获得一致的规模估算和解释这些预测值需要高的技能水平和判断力;(2)历史数据问题。大多数研究人员是基于特定环境下的历史数据提出这些模型的,而这些模型不一定适用于现在的开发环境^{[12](p.244)}; (3)目前使用的许多算法模型是基于回归技术,软件开发工作量作为被解释变量,软件规模或其他项目特征作为解释变量。然而,一个好的回归模型需要相当大的有效数据集合。目前,很少有公司存储了以往项目的数据,就算有,目前项目开发采用新的技术和软件工具,而历史数据是与老的技术和特征直接有关,因此,这些历史数据的可用性也变低了。另外,采用回归方法所基于的假设(如成本驱动因素之间的相互独立性等)往往很难成立;(4)许多估算模型只为开发过程的后期阶段(实现阶段)估计时间或成本,而忽视了前期的分析和设计阶段^{[13](p.177)}。

面向学习的估算技术如人工神经网络(ANNS)等是最近几年开始用于软件项目估算的。该技术不需要任何函数形式的假设,能够建模复杂的非线性关系,并且具有学习功能,使预测值的精度越来越高。G.R. Finnie(1997)对软件工作量估算方法:人工神经网络(ANNS)、基于案例的推导(CBR)和回归模型进行了比较,结果表明,面向学习的估算技术能够提供更准确的估计。但是这类技术的效果很大程度上取决于用于训练或类推的大样本数据。因此,一个组织采用该技术进行估算需要保存好该组织所开发的所有软件项目的记录。这些记录不仅要包含所有与人员等级、技能、耗费的开发时间、开发环境和可用工具等有关的数据,还须包含所有的项目不同阶段的估算和对估算所做的调整的信息。由于人工神经网络等本身的复杂性及该技术对历史数据的需求,该估算技术目前在软件项目估算中很少被采用^{[2](pp.288-289)}。

从上述软件项目工作量估算技术的比较分析中可以看到,不同估算技术各有优点和局限性,没有一种估算技术能适用于所有类型的软件及所有开发环境,并且软件开发方法和技术的更新速度也对所有这些估算技术提出了挑战。因此,软件组织应根据具体的项目特征和可获得的信息来选择合适的估算技术,如当软件开发组织缺少量化的、历史数据的情况下,可以采用基于专家判断的估算技术,并且对于确定算法模型的调整因子来说,该技术也是一种有效的估算工具;当软件开发组织具有大规模的项目数据库并且对成本驱动因子有很好的理解时,可以采用面向学习的估算技术或基于模型的估算技术。另外,在文献中也出现许多复合模型的研究^{[14][15][16]},这些研究证实结合不同的估算技术,如把算法模型和专家判断或把算法模型和面向学习的估算技术结合起来使用,可以提高工作量估算的准确性。因此,软件组织在进行工作量估算时,可以考虑把多种不同的估算技术结合起来,以获得更准确的预测。

一些研究证明,基于算法模型的估算技术缺乏有效性,其主要原因是支持大多数估算模型的经验数据是来源于一个有限的项目样本库,并且这些经验模型的提出是基于当时的开发环境,软件组织在必要时应根据当前项目情况对估算模型加以调整,重新计算模型中的指数和系数;其次,准确的规模估算也是这些模型应用的主要问题,项目估算人员应根据项目的实际情况(如是否面向对象系统)选择合适的方法来度量规模;另外,一些研究证明,用软件开发组织本身的历史项目数据来构造一个算法模型是关键的,构建本土化模型比通用的模型更准确^{[17][18][19]}。同样,面向学习的估算技术的有效性很大程度上也取决于用于训练或用于类推的大样本数据,研究证明,软件组织基于自己的数据库比基于公用的数据库(行业数据)的预测会更准确^{[20](p.1016)}。因此,要获得合理、准确的估算,软件开发组织应建立本组织的软件项目库,保存好本组织所开发的所有软件项目的历史记录。

目前,许多估算方法多为开发过程的后期阶段估计时间或成本,忽视了前期阶段,而软件开发工作量的估算应是从软件计划、需求分析、设计、编码到测试,应以整个开发过程所花费的代价作为依据。工作量估算仅仅考虑软件的实现是不够的,实践证明,需求分析和设计的好坏会影响到软件的实现,最终会影响到软件产品的质量和交付,如果在需求和设计阶段投入相当的工作量,编程工作就变得相对简单。随着面向对象技术的应用,基于构件的开发模式使得系统实现更容易,而系统的分析和设计阶段更重要。因此,软件组织在进行工作量估算时,要充分考虑到项目前期阶段的工作量。

软件工程大师 Roger S. Pressman 指出,软件成本及工作量估算永远不会是一门精确的科学,太多的变化——人员、技术、环境、策略——影响了开发所需的工作量及软件项目的最终成本。但是,将完备的历史数据与系统化的技术结合起来,就能够提供具有可接受风险的估算,提高估算的准确度^{[21](p.102)}。

[参 考 文 献]

- [1] Standish Group International, Chaos: Project Failure and Success Research Report[R]. <http://www.standishgroup.com/chaos.html>, 2001 - 09 - 19/2005 - 02 - 18.
- [2] Finnie, G. R., Wittig, G. E. et al.. A Comparison of Software Effort Estimation Techniques: Using Function Points with Natural Networks, Case-Based Reasoning and Regression Models[J]. Systems Software, 1997, (39): 281 - 289.
- [3] Kemerer, C. F., Porter, B. S.. Improving the reliability of Function Point Measurement: An Empirical Study[J]. IEEE Transactions on Software Engineering, 1992, 18(11): 1011 - 1024.

- [4] Caldiera, G. , Antoniol, G. , Lokan, C. et al. . Definition and Experimental Evaluation of Function Points for Object-Oriented Systems[C] . Software Metrics Symposium, Proceedings. Fifth International, 1998: 167 – 178 .
- [5] Costagliola, G. , Tortora, G. . Class Point: An Approach for the Size Estimation of Object-Oriented Systems [J] . IEEE Transactions on software engineering, 2005, 31(1): 52 – 74 .
- [6] Fetcke, T. , Abran, A. , Nguyen, T. H. . Mapping the OO-Jacobson Approach into Function Point Analysis[C] . Proceedings of Technology of Object-Oriented Languages and Systems, 1997: 192 – 202 .
- [7] Kaczmarek, J. , Kucharski, M. . Size and effort estimation for applications written in Java[J] . Information and Software Technology, 2004, 46: 589 – 601 .
- [8] Heiat, A. . Comparison of artificial neural network and regression models for estimating software development effort[J] . Information and Software Technology, 2002, 44: 911 – 922 .
- [9] Molokken, K. , Jorgensen, M. . A Review of software surveys on Software Effort Estimation[C] . International Proceedings of Empirical Software Engineering, ISESE, 2003: 223 – 230 .
- [10] Kemerer, C. F. . An empirical validation of software cost estimation models[J] . Communications of the ACM, 1987, 30(5): 416 – 429 .
- [11] Ferens, D. V. , Gurner, R. B. . An Evaluation of Three Function Point Models of Software Effort [C] . IEEE National Aerospace and Electronics Conference, 1992, 2: 625 – 642 .
- [12] Vicinaza, S. S. , Mukhopadhyay, T. , Prietula M. J. . Software-Effort Estimation: An Exploratory Study of Expert Performance [J] . Information Systems Research, 1991, 2(4): 243 – 262 .
- [13] Chatzoglou, P. D. , Macaulay, L. A. . A review of existing models for project planning and estimation and the need for a new approach[J] . International Journal of Project Management, 1996, 14(3): 173 – 183 .
- [14] Myrtveit, I. , Stensrud, E. . A controlled experiment to assess the benefits of estimating with analogy and regression models [J] . IEEE Transaction of Software Engineering, 1999, 25: 510 – 525 .
- [15] Briand, L. C. , Emam, K. E. , Bomarius, F. . COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking and Risk Assessment[C] . Proceedings IEEE International Conference on Software Engineering, 1998, 390 – 399 .
- [16] Briand, L. C. , Wust, J. . Modeling Development effort in Object-Oriented Systems using design properties [J] . IEEE Transactions on Software Engineering, 2001, 27(11): 963 – 986 .
- [17] Kemerer, C. F. . An empirical validation of software cost models[J] . Communications of the ACM, 1987, 30(5): 416 – 429 .
- [18] Jack, R. , Mannion, M. . Improving the software cost estimation process [J] . Software Quality Management III, 1995, 1: 245 – 256 .
- [19] Cuelenaere, A. , Van Genuchten, M. , Heemstra, F. . Calibrating a software cost estimation model: why and how [J] . Information and Software Technology, 1987, 29(10): 558 – 567 .
- [20] Jeffery, R. , Ruhe, M. , Wiczorek, I. . A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data[J] . Information and Software Technology, 2000, 42: 1009 – 1016 .
- [21] Pressman, R. S. . 软件工程实践者的研究方法[M] . 北京: 机械工业出版社, 2002 .

[责任编辑 江予新]

A Comparison of Effort Estimation Techniques in Software Development Projects

WANG Qiu-zhen

(Department of Management Science and Engineering , Zhejiang University , Hangzhou 310027 , China)

Abstract: Estimating the effort to develop a software system is an important project management concern. There are a number of different effort estimation techniques currently in use in the software industry. They fall into

three general categories: expert judgment-based techniques, algorithmic model-based techniques and learning-oriented techniques. Expert judgment-based techniques are the techniques in most frequent use. They are useful in the absence of quantified, empirical data. However, "intuitive" processes constitute a major part of the expert estimation. It is difficult to articulate the estimation process and achieve organizational learning. Algorithmic model-based techniques attempt to represent the relationship between effort and one or more project characteristics. The existing algorithmic models, however, fail to produce accurate estimates of the development effort. There are four main disadvantages with these models. First, there is the problem of software sizing. Obtaining consistent size estimates and interpreting such estimates require high skill levels and good judgment. Secondly, there is the issue of historical data. Researchers have proposed models based on site-specific historical data, so these models can't be applied to local development circumstances. Thirdly, the dominant method of data analysis has been in linear regression. A good regression model needs a relatively large dataset which can be a problem in the software estimation field. Furthermore, the underlying assumptions of regression analysis are often questionable. Finally, many models estimate the effort for the later stages of the development process but almost ignore the early stages. Learning-oriented techniques are based on artificial intelligence techniques. Learning-oriented techniques don't require any function form assumption, as opposed to algorithmic models and they are good at modeling complex non-linear relationships. The performance of the learning-oriented techniques is to a large degree dependent on the large dataset for training or analogy.

Each of the existing techniques for effort estimation has its advantages and disadvantages. No single technique is best for all situations, and all techniques are challenged by the rapid pace of change in the software technology. In order to improve estimation accuracy, a software development organization should: (1) select proper estimation techniques according to specific project characteristics and information availability; (2) combine more estimation techniques to improve effort estimates; (3) adapt algorithmic models to local circumstances; (4) develop their own store of projects; (5) estimate the effort for the whole development process, considering the effort of the early stages sufficiently.

Key words: software development project; size estimation; effort estimation; expert judgment-based techniques; algorithmic model-based techniques; learning-oriented techniques

关于加入“中文科技期刊数据库”的声明

为适应我国信息化建设,扩大本刊及作者知识信息交流的渠道,《浙江大学学报(人文社会科学版)》自 2005 年起加入“中文科技期刊数据库”(重庆维普)。作者文章著作权使用费与稿酬一次付清,本刊不再另付其他报酬。如作者不同意文章被收录,请在来稿时向本刊声明,本刊将做适当处理。

浙江大学学报(人文社会科学版)