ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.6,No.34, December 2010, pp.9917-9919

E-mail: kfyj@cccc.net.cn
http://www.dnzs.net.cn
Tel:+86-551-5690963 5690964

改进的用例点估算方法

赵文杰[1],刘俊萍[2],南振岐[3]

(1.西北师范大学 数学与信息科学学院,甘肃 兰州 730070;2.郑州电力职业技术学院 信息工程系,河南 郑州 451450;3.兰州南特数码科技股份有限公司,甘肃 兰州 730010)

摘要:目前用例模型作为一种捕捉和分析软件功能性需求的方法已经被广泛采用。用例点估算方法正是在此基础上发展起来的一种很有潜力的软件估算方法。该文分析研究用例点方法的不足之处,并在此基础上提出了一种改进的用例点估算方法,从而进一步提高了该方法的客观性、准确性和易用性。

关键词:UCP;用例点;软件估算

中图分类号:TP312 **文献标识码**:A **文章编号**:1009-3044(2010)34-9917-03

Improved Use Case Points Estimation Method

ZHAO Wen-jie[1], LIU Jun-ping[2], NAN Zhen-qi[3]

(1.College of Mathematics and Informational Science, Northwest Normal University, Lanzhou 730070, China; 2.Department of Information and Project, Zhengzhou Electric Power Technology College, Zhengzhou 451450, China; 3. Lanzhou NETDIGITAL Ltd., Lanzhou 730010, China)

Abstract: Use case models are increasingly being used to capture and analyze the functional requirements of a software. It has been tested that use case points method based on use cases is a potential method for estimation software development. The paper analyses the shortcoming of the use case point estimation method. Then an improved method was proposed. This improved method is more objective, concise and precise than the original method.

Key words: UCP; use case points; software estimation

在软件开发前期,制定适合的软件开发计划对于整个软件项目的管理和项目的成功都至关重要。而制定正确的软件开发计划的前提是正确的软件估算,它可以为软件开发计划提供重要的数据支持。目前用例模型作为软件项目需求采集和分析的重要工具越来越得到广泛青睐,因此研究如何使用基于用例估算方法是很有意义的。

用例点方法[1](use case point method),是由 Gustav Karner 在 1993 年针对 FPA(function point access)方法而提出的一种改进方法[1]。大量研究和实例应用表明该方法是一种可靠、易于学习和使用的估算方法,其可靠性至少不输于重量级估算方法 COCOMOII[2]。

但目前该方法的一个显著不足是估算过程中有过多主观因素的存在。这势必大大影响了估算的可信度和准确性,致使不同的估算人员对同一项目进行单独估算时出来的估算结果可能会有较大差别。

对于上述问题本文提出了一种改进的用例点估算方法,可以在一定程度上降低个人主观对估算的影响,从而提高估算方法的稳定性和准确性。

1 用例点估算方法。

用例点法分析用例角色,场景和各种技术、环境因素,并抽象它们,得到一个计算公式。

计算公式包含下面三个参数组成:

1) 技术复杂度因子 (TCF)。

2) 环境复杂度因子 (ECF)。

3) 未调整用例点数 (UUCP)。

1.1 未调整用例点数 (UUCP)

UUCP 有下面两部分组成:

原始用例权重 (UUCW):基于用例场景中包含的活动数。

原始角色权重 (UAW):基于用例角色中多角色的组合复杂性。

UUCP=UUCW+UAW

1.1.1 原始用例权重 (UUCW)

单个用例可被分成简单、普通和复杂三种。如表 1 所示。

表 1

复杂度	说明	权重
simple	1 到 3 个用例事务[6]。它的实现涉及到 5 个类以内的用例。	5
Average	4 到 7 个用例事务。它的实现涉及到 5~10 个类的用例。	10
Complex	多于 7 个用例事务。它的实现涉及到 10 个类以上的用例。	15

收稿日期:2010-10-16

作者简介:赵文杰(1979-),男,河南安阳人,西北师范大学,硕士研究生。

通过计算各个种类用例的数目,乘上各种用例的权重,然后加和,就得到 UUCW。

$$UUCW = \sum_{i=1}^{3} n_i W_i$$

n_i 是相应复杂度用例的数目,W_i 是相应复杂度用例的权重。

1.1.2 原始角色权重 (UAW)

根据角色间的相互作用,角色分为简单、普通和复杂。如表 2 所示。

$$UAW = \sum_{i=1}^{3} n_i W_i$$

n_i 是相应角色类型在项目中的数目,W_i 是相应角色的权重。

1.2 技术复杂度因子 (TCF)

13 个标准技术因素对估算生产力存在影响，这些生产力可能涉及应用中的各种技术问题。每种因素都有与其相关影响相应的权重。如表 3 所示。

每个项目的技术因素 F_i 由开发团队来估算，根据相应的复杂度来设定从 0 到 5 之间的值。复杂度为 0 意味着技术因素与本项目无关;3 意味着一般;5 意味对项目有很强的影响。

TCF 的计算公式如下:

$$TCF = C_1 + C_2 \sum_{i=1}^{13} F_i W_i$$

其中常量 C_1 =0.6,C_2 =0.01。 F_i 是第 i 项的估值,W_i 是第 i 项的权重。

1.3 环境复杂度因子 (ECF)

环境复杂度因子估算各种环境因素对应用生产力的影响。如表 4 所示。

每种环境因素 F_i 都被评估,并根据其影响加权,权重值在 0 和 5 之间。

ECF 的计算公式如下:

$$EF = C_1 + C_2 \sum_{i=1}^{8} F_i W_i$$

其中常量 C_1=1.4,C_2=–0.03,F_i 是第 i 项的估值,W_i 是第 i 项的权重。

1.4 用例点(UCP)

以上三个参数每个参数都独立定义和计算,各个参数计算中,部分使用分析得到的值,部分使用常量。经过环境因子和技术因子对 UUCP 调整后得到 UCP 完整的公式为:

UCP=TCF×EF×UUCP

1.5 工作量估算

项目生产率(PF)是根据过去的项目得到的平均每个用例点所花的人时比率。如果没有历史数据可供参考,行业专家推荐使用 20–28 之间的一个数字,典型的,你可以使用 20。

研发工作量 DEV_t

DEV_t=PF×UCP

表 2

角色类型	说明	权重
Simple	角色通过良好定义的 API 与另一系统交互。	1
Average	角色通过某种协议(如：TCP/IP)与另一系统交互。	2
Complex	系统的最终用户。	3

表 3

技术因素	说明	权重
TF1	系统分步式程度	2
TF2	系统性能要求	1
TF3	最终用户使用效率要求	1
TF4	内部处理复杂度	1
TF5	复用程度	1
TF6	易于安装要求度	0.5
TF7	系统易于使用程度	0.5
TF8	可移植性	2
TF9	系统易于修改程度	1
TF10	并发性要求	1
TF11	特殊安全功能特性要求	1
TF12	为第三方系统提供直接系统访问	1
TF13	是否需要特殊的用户培训设施	1

表 4

环境因素	说明	权重
EF1	UML 精通程度	1.5
EF2	系统应用经验	0.5
EF3	面向对象经验	1
EF4	系统分析员能力	0.5
EF5	团队士气	1
EF6	需求稳定度	2
EF7	兼职人员比例高低	-1
EF8	编程语言难易程度	-1

2 改进的用例点估算方法

在 UCP 估算模型中并没有明确给出 8 个环境因素的度量规范。这样就带来一个问题 8 个环境因素的取值将完全由主观决定,这势必造成不同估算人员估算出的结果可能差别较大从而影响估算结果稳定性和准确性。

为解决该问题我们分析 UCP 方法中给出的 8 个环境因素后,发现除“需求稳定度”和“编程语言难易程度”两个因素外其它因素均反应的是开发团队的开发能力情况。就一个软件企业来看其开发团队一般来讲在一定时期内都是比较稳定的,也就是说这些因素值在一定时期内都是比较稳定的,不会有太大的变化。而开发语言通常对于一个开发团队来说通常也是固定的。随着企业过程改进的进行和人员素质的提高这些因素会逐步提升,但这方面所带来的环境因素的影响我们可以通过生产率 PF 的提高来反应。

综上所述我们最终决定在改进后的用例点估算方法中去除 EF 调整因子,而加入用来反应项目需求变化率的调整因子 RF。

RF=1+RV

RV 为需求变化率是反应组织在开发同类软件产品时平均需求变化情况，该项目数据可以通过分析项目团队的历史项目数据得到。

RV=变化的 UCP 值/原始的 UCP 值;

改进后的 UCP 计算机公式为:

UPC=RF×TCF×UUCP

公式中去除了具有过多主观因素决定的 EF 因子,使得算法更简洁更客观。

3 案例研究

3.1 实验

我们在某一软件开发组织内对两种估算方法做一实验。给定某一项目现由组织内的六名估算人员独立对该项目进行估算。我们要求 3 人使用原 UCP 估算法,另外 3 个人使用改进后的 UCP 估算法,结果如表 5。

表 5

	UCP 估算法	改进的 UCP 估算法
人员 1	2480	
人员 2	2680	
人员 3	3060	
人员 4		2940
人员 5		2700
人员 6		3160
均值	2740	2933
实际值	3346	3346
均值错误率	18%	12%
标准差	295	230

3.2 结果分析

数据结果表明由于改进后的算法去除了环境因子,减小了主观因素的影响,从而使估算值的稳定性比原方法有所提高,标准差从原方法的 295 下降到 230。同时方法的估算准确度也有所提高,从原方法的误差率 18%下降到 12%。分析原因主要是在使用原 UCP 估算方法时团队人员在对环境因子进行估值时常常过高地估计团队的能力所致。现在去除此因子,将环境因素的影响力分散到更加客观的“需求变化率因子”和“生产率”的变化上,从而在一定程度上提高了结果的稳定性和准确性。但该改进后的方法到底有多大的影响力,目前由于我们还没有足够多的项目数据进行统计分析所以无法精确给出,我们还需要在更多的项目上实验分析后得出进一步的结论。

4 结论

文中针对环境因子在估算时没有明确规范和标准, 会加入估算者的过多主观因素的问题提出了一种改进的 UCP 估算方法,改进后的方法在操作时更加简单、易用,并且提高了方法的稳定性和准确性。但该方法仅适用于人员和素质比较稳定的开发团队,且应注意不同的开发语言环境应该采用不同的生产率进行计算。在实战中为进一步提高估算的准确性建议多种方法结合使用,如可以和专家法结合,当两种估算方法结果偏差较大时可以分析原因后重新估算。也可采用多人独立估算,后取均值的方法,实验证明多人估算总是会比单人估算的准确性高。

参考文献:

[1] Karner G.Resource Estimation for Objectory Projects[M].Objective Systems SF AB,1993.

[2] Boehm B W.COCOMOII 模型方法[M].北京:机械工业出版社,2005.

[3] Cockburn A.Writing Effective Use Cases[M].Addison-Wesley,2001.

[4] Kirsten R.Estimating Object-Oriented Software Projects with Use Cases[D].Master of Science Thesis,Department of Informatics,University of Oslo,2001.

[5] Geri S, Winters J P.Applying Use Cases: A Practical Guide[M].Addison Wesley,1998.

[6] Boehm B.W.Software Engineering Economics[M].Prentice-Hall,1981.

[7] Collaris R,Dekker E.Software cost estimation using use case points: Getting use case transactions straight[J].The Rational Edge,2009(4).