基于用例点的软件项目工作量估算

朱 萍,任永昌
(渤海大学 信息科学与技术学院,辽宁 锦州 121013)

摘 要: 目前软件估算在软件开发过程中的应用相当普遍而且重要,而以用例模型为基础的用例点估算方法具有容易操作性、实用性、可靠性好等特点。同时对于使用用例估算,存在着不同的方法和途径来估计工作量。文中主要研究软件估算方法中的用例点估算方法,详细描述其在项目工作量估算过程中的流程和方法以及相对应的估算公式。最后通过一个简单的案例展示项目工作量的估算过程,能更好地说明在软件项目工作量估算中用例点估算方法的方便及其可行之处,从而为项目计划和管理提供更好的支持。

关键词: 软件估算; 用例点; 用例模型; UCP 方法; 软件项目估算
中图分类号: TP31 **文献标识码:** A **文章编号:** 1673-629X(2012)12-0071-04

Software Project Effort Estimation Based on Use Case Points

ZHU Ping, REN Yong-chang
(College of Information Science and Technology, Bohai University, Jinzhou 121013, China)

Abstract: Software estimation is quite common and important in the software development process. Use case point estimation method based on the use case model has characteristics of easy maneuverability, practicality, reliability. And there exists different ways and means to estimate the workload with use case estimation. It researches the use case point methods in software estimation methods and describes the processes and methods and the corresponding estimation formulas in the software project effort estimation in detail. Finally, through a simple case demonstrate the project effort estimation process that can better describe the convenience and feasibility with use case point estimation method in the software project effort estimation, providing the better support for the project planning and management.

Key words: software estimation; use case points; use case model; UCP method; software project estimation

0 引 言

软件估算是软件开发计划的前提,而制定一个合适的软件开发计划对于整个软件项目[1]管理及其成功是至关重要的。所以亦说软件工作量估算是成功的软件项目管理的重要的一部分。估算正确与否起着关键性的作用:过低的估算会导致该项目因为缺少工作量时间或资金而不能及时地在承诺的时间内完成;过高的估算则会加重成本,往往使得公司因要承担的成本过高而放弃该项目的竞标,使该项目失之交臂,而原本该项目是可以为公司带来更多机会和效益的。

一般来说,衡量一个软件项目的成功与否,就在于软件是否在不超支的情况下按合同的要求完成,并且产品达到或超过成本、进度和质量的预期目标。

收稿日期:2012-04-10;修回日期:2012-07-12
基金项目:国家自然科学基金资助项目(70871067);辽宁省自然科学基金资助项目(20072207)
作者简介:朱 萍(1987-),女,硕士研究生,研究方向为软件开发成本估算方法;任永昌,博士,教授,主要从事项目管理、成本估算、信息系统方面的研究。

据调查显示,大多数的软件项目都处于成本超支、时间超支状态中。

目前软件估算的方法很多,包括:代码行(LOC,Line of Code)、类比法(Analogy Approach)、自顶向下估算法(Top-Down)、自底向上法(Bottom-Up)、功能点分析法(FPA,Function Points Analysis)、参数化模型法(Parametric Models)、Putnam 方法、用例点法(UCP,Use Case Points)以及专家估算法(Expert Judgment)也叫 Delphi 法等等。而文中主要介绍 UCP 估算方法以及其估算过程。

1 用例模型

文中介绍的 UCP 估算方法必然要用到用例模型。下面先简单介绍一下用例模型。

用例模型(Use-Case Model)是系统既定功能及系统环境的模型,它可以作为客户和开发人员之间的契约。用例是贯穿整个系统开发的一条主线。同一个用例模型即为需求工作流程的结果,可当作分析设计工作流程以及测试工作流程的输入使用。图 1(a)是相关的一个用例模型举例。

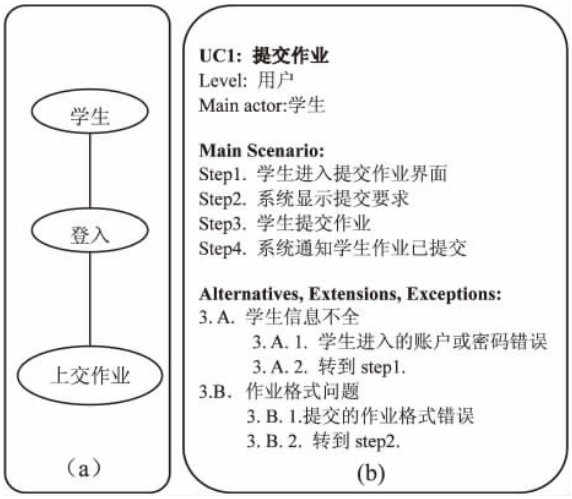

图1 用例模型示例

用例模型肯定关系到特定的用例。根据描写用例指南[2,3]，一个用例最重要的部分为：name/title 用来描述目的的；actors 是该用例的参与者；main scenario 是实现目的的最主要的步骤；extensions 是描述某些事件发生时的步骤。具体用例的例子[4]如图1(b)所示。而用例模型、用例、系统等主要概念相关性如图2所示。

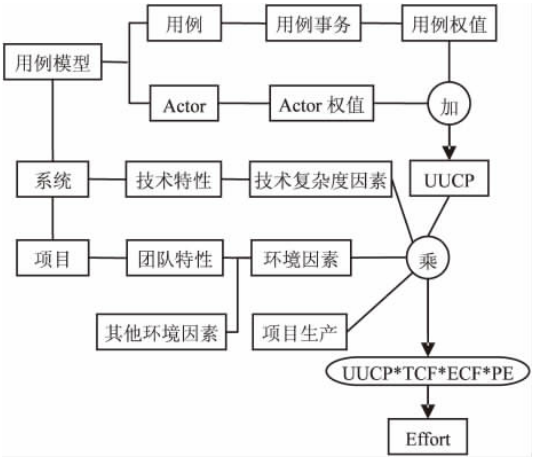

图2 相关概念的联系图

2 UCP 估算算法

UCP 估算算法是以用例模型为基础的。整个项目工作量估算过程：计算用例点，然后通过项目生产率的取值，计算用例点和工作量的换算，得到项目开发所需的以人小时数为单位的工作量[5]。UCP 算法是受到 Albrecht 的 FPA(Function Point Analysis)[6]和 MKII Function Points[7]的启发，并且是由 Gustav Karner 在1993提出，在对 Use Case 的分析的基础上进行加权调整得出的一种改进方法。

UCP 估算方法的基本步骤如下：

(1) 对每个角色 Actor 进行加权，计算未调整的角色的权值 UAW(Unadjusted Actor Weights)；

(2) 计算未调整的用例权值 UUCW(Unadjusted Use Case Weights)；

(3) 计算未调整的用例点 UUCP(Unadjusted Use Case Points)；

(4) 计算技术和环境因子 TEF(Technical and Environment Factor)；

(5) 计算调整的用例点 UCP(Use Case Points)；

(6) 根据规模—工时的转换因子来计算工作量(man-hours)。

2.1 估算 UAW

首先将软件需求用 Use Cases 方式表达，其次利用 Actor(参与者)的数量乘以相应的权值来计算 UAW。具体计算公式如下：

$$UAW = \sum_{c \in C} aWeight(c) \times aCardinality(c) \quad (1)$$

其中 $C = \{simple, average, complex\}$，aCardinality(c)是参与的角色数目。其权值如表1前半部分所示，Actor 的复杂度根据复杂度标准的定义分为3个不同的级别，而每个不同的级别又对应着不同的权值。

表1 Actor 权值定义

序号	复杂度级别	复杂度标准	权值	参与数	UAW_i
1	simple	角色通过 API 与系统交互	1	2	2
2	average	角色通过协议与系统交互	2	4	8
3	complex	用户通过 GUI 与系统交互	3	5	15

2.2 估算 UUCW

利用 Use Case(参与者)的数量乘以相应的权值来计算 UUCW。具体计算公式如下：

$$UUCW = \sum_{c \in C} uWeight(c) \times uCardinality(c) \quad (2)$$

其中 $C = \{simple, average, complex\}$，uCardinality(c)是参与的用例数目。其权值如表2前半部分所示，Use Case 的复杂度根据场景个数分为3个不同的级别，而每个级别又各自对应着相应的权值。

表2 Use Case 权值定义

序号	复杂度级别	事务/场景个数	权值	个数	$UUCW_i$
1	simple	1-3	5	5	25
2	average	4-7	10	2	20
3	complex	>7	15	3	45

2.3 估算 UUCP

估算未调整的用例点(UUCP)，如图2的联系图所示，是由在2.1和2.2的基础上计算得到的未调整的 Actor 权值和未调整的用例权值相加得到，即：

$$UUCP = UAW + UUCW \quad (3)$$

2.4 估算 TEF

UCP 估算方法中有21个适应性因子[8]，其中包括开发系统的技术复杂度和开发环境，即分为13个技术复杂因子和8个环境复杂度因子。换句话也就是说

TEF 包括技术复杂度因子(TCF)和环境复杂度因子(ECF)。

2.4.1　估算 TCF

TCF 的计算公式如下:

$$TCF = 0.6 + \left(0.01 \times \sum_{i=1}^{13} TF\_weight_i \times value_i\right) \quad (4)$$

其中,TCF_weight_i 的值见表 3 前半部分,$value_i$ 则根据该技术复杂度因子的影响等级 0 到 5 之间的值来确定[9]:0 表示该技术因子与本项目无关;3 表示该技术因子对本项目的影响一般;5 表示该技术因子对本项目有很强的影响。

表 3　技术复杂度因子的定义

序号	技术因子	说明	权值	Value 值	TCF_i
1	TCF1	分布式系统	2.0	3	6.0
2	TCF2	性能要求	1.0	5	5.0
3	TCF3	最终用户使用效率	1.0	3	3.0
4	TCF4	内部处理复杂度	1.0	5	5.0
5	TCF5	复用程度	1.0	0	0.0
6	TCF6	易于安装	0.5	3	1.5
7	TCF7	系统易于使用	0.5	5	2.5
8	TCF8	可移植性	2.0	3	6.0
9	TCF9	系统易于修改	1.0	5	5.0
10	TCF10	并发性	1.0	3	3.0
11	TCF11	安全功能特性	1.0	5	5.0
12	TCF12	为第三方系统提供直接系统访问	1.0	0	0.0
13	TCF13	特殊的用户培训设施	1.0	0	0.0

2.4.2　估算 ECF

ECF 的计算公式如下[10]:

$$ECF = 1.4 + \left(-0.03 \times \sum_{i=1}^{8} ECF\_weight_i \times value_i\right) \quad (5)$$

其中,ECF_weight_i 的值见表 4 前半部分,$value_i$ 的值同 TCF 中的值一样都是由开发团队根据该因子的影响等级来确定:0 表示项目组成员都不具备该因素;3 表示环境因子对本项目的影响程度中等;5 表示本项目组所有成员都具有该因素。

表 4　环境因子的定义

序号	环境因子	说明	权值	Value 值	ECF_i
1	ECF1	UML 精通程度	1.5	3	4.5
2	ECF2	系统应用经验	0.5	3	1.5
3	ECF3	面向对象经验	1.0	3	3.0
4	ECF4	系统分析员能力	0.5	5	2.5
5	ECF5	团队士气	1.0	3	3.0
6	ECF6	需求稳定度	2.0	3	6.0
7	ECF7	兼职人员比例高低	-1.0	0	0.0
8	ECF8	编程语言难易程度	-1.0	0	0.0

2.5　估算 UCP

以上 UUCP、TCF、ECF 三个参数每个参数都是独立定义和计算[11],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 2.4 环境因子和技术因子对 UUCP 调整后得到 UCP 完整的公式为:

$$UCP = UUCP \times TCF \times ECF \quad (6)$$

2.6　估算工作量

项目的工作量估算[12]也就是 UCP 的值乘以相对应的项目生产率。项目的工作量计算公式为:

$$Effort = UCP \times PF \quad (7)$$

其中,PF(Productivity Factor),即项目生产率。对于 PF 的取值,在没有历史数据可参考的情况下,一般取其默认值 20,该默认值是由 Karner 提出的。后期,Scheider 和 Winterstate 对关于 PF 的初始值如何取值提出了一个方法。计算 EF1 到 EF2 的值的和 t,如果 $t < 2$,则 PF 取值为 20;如果 $3 \leqslant t \leqslant 4$,则 PF 取值为 28;如果 $t > 4$,则 PF 取值为 36。

3　案例分析

通过对用例模型、用例、系统等主要概念相关性的了解和上述 UCP 算法的介绍以及其主要计算过程的基本步骤的分析和对各步骤公式的理解,下面就通过某一实际项目小案例,对该项目采用文中介绍的用例点方法来估算项目的工作量。

(1)首先根据实际项目所提供的数据以及据此计算得到的各复杂度级别所对应的值,如表 1、表 2 所示,通过式(1)~(3)计算 UAW、UUCW 和 UUCP;

$$UAW = \sum_{c \in C} aWeight(c) \times aCardinality(c) = 1 \times 2 + 2 \times 4 + 3 \times 5 = 2 + 8 + 15 = 25$$

$$UUCW = \sum_{c \in C} uWeight(c) \times uCardinality(c) = 5 \times 5 + 10 \times 2 + 15 \times 3 = 25 + 20 + 45 = 85$$

$$UUCP = UAW + UUCW = 25 + 85 = 110$$

(2)根据实际项目所提供的数据以及据此计算得到的各技术复杂度因子和环境复杂度因子对应的值如表 3、表 4 所示,通过式(4)和(5)计算 TCF、ECF;

$$TCF = 0.6 + \left(0.01 \times \sum_{i=1}^{13} TFweight_i \times value_i\right)$$
$$= 0.6 + 0.01 \times (2.0 \times 3 + 1.0 \times 5 + 1.0 \times 3 + 1.0 \times 5 + 1.0 \times 0 + 0.5 \times 3 + 0.5 \times 5 + 2.0 \times 3 + 1.0 \times 5 + 1.0 \times 3 + 1.0 \times 5 + 1.0 \times 0 + 1.0 \times 0)$$
$$= 1.02$$

$$ECF = 1.4 + \left(-0.03 \times \sum_{i=1}^{8} ECFweight_i \times value_i\right)$$
$$= 1.4 + [-0.03 \times (1.5 \times 3 + 0.5 \times 3 + 1.0 \times 3 + 0.5 \times 5 + 1.0 \times 3 + 2.0 \times 3 + -1.0 \times 0 + -1.0 \times 0)]$$
$$= 0.785$$

(3)由(1)、(2)可知,再应用公式(6)计算 UCP;

$$UCP = UUCP \times TCF \times ECF = 110 \times 1.02 \times 0.785 = 88$$

(4)根据该项目中的 PF 的取值,应用公式(7)计算项目的工作量;

$$Effort = UCP \times PF = 88 \times 20 = 1760h \quad (PF = 20)$$
$$Effort = UCP \times PF = 88 \times 28 = 2464h \quad (PF = 28)$$

Effort = UCP×PF = 88×36 = 3168h (PF = 36)

通过上述的计算可知该项目的工作量在 1760h 到 3168h 之间。假设每个工作人员一周工作 35h,且共有 8 个人员参与,则每周的工作量为 35×8 = 280h,由 1760 ÷280 = 6. 29 week,2464 ÷280 = 8. 8 和 3168 ÷280 = 11. 34 week 可知,完成该项目所需的时间在 7 周到 12 周之间。但一般估算的时候往往只计算 PF = 20 和 PF = 28 时的项目工作量,简而言之,完成该项目所需的时间在 7 周到 9 周之间。

4 结束语

现有的国内外的软件估算方法的确很多,但是在这众多的估算方法中,能将技术复杂度因素和环境复杂度因素充分考虑进去的并不是很多。而文中讨论的用例点估算方法就充分考虑这些主观因素,并且对这些主观因素的影响程度都用具体的表格表示出来,相关的计算也用对应的公式表示出来,方便计算。通过对 UCP 算法的具体介绍以及详细的估算过程步骤,最后又通过一个小的案例使人们进一步了解该方法的特色及其方便性,并且随着项目开发的进一步深入,将会发现使用用例点数得到的估算可靠性[13]。当然文中介绍的用例点算法可能还存在一定的缺陷有待进一步的改进。

参考文献:

[1] 任永昌,邢 涛,刘大成. 基于网络图的软件项目进度计划编制 [J]. 吉林大学学报,2011,29(2):128-134.

[2] Adolph S,Bramble P,Cockburn A,et al. Patterns for Effective Use Cases [R]. [s. l.]: Addison-Wesley,2002.

[3] Cockburn A. Writing Effective Use Cases [R]. Boston: Addison-Wesley,2001.

[4] Ochodek M,Nawrocki J,Kwarciak K. Simplifying effort estimation based on use case points [J]. Information and Software Technology,2011,53(3):200-213.

[5] 周 杨,吴海涛,张栋伟. 扩张的用例点估算 [J]. 计算机技术与发展,2006,16(12):64-66.

[6] Albrecht A. Measuring application development productivity [C]//Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium. [s. l.]: [s. n.],1979:83-92.

[7] Symons C R. Software Sizing and Estimating: Mk II FPA (Function Point Analysis) [R]. New York,NY,USA: John Wiley & Sons,Inc.,1991.

[8] 薛 丹,杨 宸,周 健. 一种基于区间值的模糊访问控制策略研究 [J]. 计算机技术与发展,2012,22(1):246-249.

[9] 余秋冬,徐 辉. 用例点估算方法在电信行业中的应用 [J]. 计算机工程,2009,35(24):276-277.

[10] Schneider G,Winters J. Applying Use Cases: A Practical Guide [R]. Boston,MA,USA: Addison-Wesley Longman Publishing Co.,Inc.,1998.

[11] 赵文杰,刘俊萍,南振岐. 改进的用例点估算方法 [J]. 电脑知识与技术: 学术交流,2010(12):9917-9919.

[12] 王 悠,罗燕京,易福华,等. 基于用例的软件复杂度估算及应用 [J]. 计算机技术与发展,2007,17(7):196-199.

[13] 陶 强,刘 莉,薛振清. 基于 UML 的软件规模估算服务模型研究 [J]. 信息技术与信息化,2010,7(2):70-73.

(上接第 70 页)

具有良好的性能,在四种测试类型中,ABX 测试结果提高了 10.88%,MOS 得分平均提高了 18.59%。

参考文献:

[1] Stylianou Y. Voice transformation: a survey [C]//International Conference on Acoustics,Speech and Signal Processing. [s. l.]: [s. n.],2009:3585-3588.

[2] 左国玉. 声音转换技术的研究与进展 [J]. 电子学报,2004,26(7):1165-1172.

[3] 李 波,王成友,蔡宣平,等. 语音转换及相关技术综述 [J]. 通信学报,2004,25(5):109-118.

[4] Nakamura K,Toda T,Saruwatari H,et al. Speaking-aid systems using GMM-based voice conversion for electrolaryngeal speech [J]. Speech Communication,2012,54(1):134-146.

[5] 陈 芝,张玲华. 基频轨迹转换算法及在语音转换系统中的应用研究 [J]. 南京邮电大学学报(自然科学版),2010,30(5):83-87.

[6] Laskar R H,Talukdar F A,Bhattacharjee R,et al. Voice conversion by mapping the spectral and prosodic features using support vector machine [J]. Applications of Soft Computing,2009,58:519-528.

[7] 尹 伟,易本顺. 一种基于正弦激励的线性预测模型的语音转换方法 [J]. 数据采集与处理,2010,25(2):218-222.

[8] Kunikoshi A,Qian Yao,Soong F,et al. Improve F0 modeling and generation in voice conversion [C]//IEEE International Conference on Acoustics,Speech and Signal Processing. [s. l.]: [s. n.],2011:4568-4571.

[9] 左国玉,刘文举,阮晓钢. 一种使用声调映射码本的汉语声音转换方法 [J]. 数据采集与处理,2005,20(2):144-149.

[10] 孙 俊. 基于激励源及其韵律特征的源-目标说话人声音转换研究 [D]. 合肥: 中国科学技术大学,2006.

[11] Rao K S. Voice conversion by mapping the speaker-specific features using pitch synchronous approach [J]. Computer Speech and Language,2010,24(3):474-494.

[12] 赵 力. 语音信号处理 [M]. 北京: 机械工业出版社,2008.

[13] 李燕萍,张玲华,丁 辉. 基于音素分类的汉语语声转换算法 [J]. 南京邮电大学学报: 自然科学版,2011,31(1):10-15.