World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:5, No:1, 2011

A Linear Use Case Based Software Cost Estimation Model

Hasan.O. Farahneh, Ayman A. Issa

***Abstract*—**Software development is moving towards agility with use cases and scenarios being used for requirements stories. Estimates of software costs are becoming even more important than before as effects of delays is much larger in successive short releases context of agile development. Thus, this paper reports on the development of new linear use case based software cost estimation model applicable in the very early stages of software development being based on simple metric. Evaluation showed that accuracy of estimates varies between 43% and 55% of actual effort of historical test projects. These results outperformed those of well-known models when applied in the same context. Further work is being carried out to improve the performance of the proposed model when considering the effect of non-functional requirements.

***Keywords*—**Metrics, Software Cost Estimation, Use Cases

I. INTRODUCTION

IDEALLY, software development projects should start with a feasibility study to estimate the effort and time required to deliver an operational system. Unfortunately, this is not the case in most software development projects for a number of reasons [7], and most importantly the unavailability of software cost estimation models that fit the different software development environments in the early stages of the software development life cycle.

The inherent problem with cost estimation is that small projects can be easily estimated, but the required accuracy may not be very important. On the other hand, large projects are very difficult to estimate, but the required accuracy is greater than what is normally achieved [6]. Different factors contribute to the inaccuracy of software cost estimation, such as, imprecise and drifting requirements, not enough information readily available on past projects, and algorithms that were developed and trained on specific data do not easily transfer to other environments [7].

Several requirements elicitation techniques have been used to model and specify user and system requirements. The linear use case modelling is one of these techniques and has been used observable result of value to a particular actor” [10].

Furthermore, use case models have become a common system model between all software systems stakeholders [10].

Thus, having requirements for a software system elicited, modelled, and initially specified using a use case model raises a number of questions that formed the main motivation behind this research:

1) Can a use case model be utilised as an appropriate platform to predict the size of an anticipated software system?
2) Consequently, can this predicted system size (based on its use case model) be utilised in developing use case model based software cost estimation models that can be used as a basis to estimate the software development effort?
3) To what extent can this predicted effort be accurate in the early stages of software development?
4) What external factors affect the accuracy of use case based estimates?

Section II. surveys the use case-based software cost estimation literature. The proposed linear use case-based estimation model is presented in section 3. Section 4 critically evaluates the accuracy of the proposed model. Finally, the conclusion and the outline of future work are presented in section 5.

Hasan .O.Farahneh , Electrical Engineering Department, Faculty of Engineering and Technology University of Jordan Amman Jordan h.farahneh@ju.edu.jo
Ayman A. Issa ,Software Engineering Department, Faculty of Information Technology, Philadelphia University P.O. Box 1, Amman. 19392, Jordan aissa@philadelphia.edu.jo

II. USE CASE MODEL BASED SOFTWARE COST ESTIMATION

The software cost estimation literature describes numerous use case model based estimation models and methods. Karner [9], as detailed in section 2.1, developed the Use Case Points (UCP) method that utilises the identified actors and use cases to size and estimate software development projects. Smith [8], as detailed in section 2.2, proposed another hierarchal approach to translate use cases into equivalent Lines Of Code (LOC) and used it as an input to a LOC dependent estimation model to predict the required effort and time. Issa’s approach of use case based software cost estimation is summarized in section 2.3.

A. Use Cases Points

UCP is a software sizing and estimation method based on use case model. The calculation of the final UCP count for a given application is accomplished in two steps. First, the Unadjusted UCP (UUCP) count is calculated based on the unadjusted weighted actors and use cases. Then, the Adjusted UCP (AUCP) count is calculated by adjusting the UUCP count using technical complexity and environment adjustment factors.

The UUCP represents the sum of the Unadjusted Actor Weights (UAW) and Unadjusted Use Case Weights (UUCW). The UAW depends on the actor types and complexity weights. Karner identified three actor types: simple, average, and complex. The simple actor represents another system with a defined application programming interface. The average actor represents another system interfacing through a protocol such as TCP/IP. The complex actor represents a person interacting through a graphical user interface or a web page. The UAW is calculated by multiplying the number of each actor type by its weight, as summarised in table I, and summing to give the total. The UUCW depends on the use case types and complexity weights. Karner identified three use case types: simple, average, and complex. The use case type is determined by the number of transactions within the use case scenarios. The use case types, number of transactions per use case type, and complexity weights are defined in table II. The UUCW is calculated by multiplying the number of each use case type by its weight and then summing to obtain the total.

The Technical Complexity Adjustment Factor (TCAF) and Environment Factor (EF) consist of 13 and 9 sub-factors, respectively. Each sub-factor has a weight according to how it affects productivity. Tables III and IV summarise the TCAF and EF sub-factors, respectively, and their weights. Each sub-factor is assigned a value between 0 (no influence) and 5 (strong influence) to represent its effect, then the effect of each sub-factor is multiplied by its weight and all the numbers are summed to form the TFactor and EFactor, respectively. TCAF is then calculated as: $TCAF = 0.6 + (0.01 \times TFactor)$, whereas EF is calculated as: $EF = 1.3 + (-0.3 \times EFactor)$. Finally, the AUCP is calculated as: $AUCP = UUCP \times TCAF \times EF$. For a project estimate, Karner [9] proposed 20 staff hours per UCP. Other field studies [3] showed that effort can range between 15-30 staff hours per UCP.

B. *Rationale's use case effort estimation*

Smith [8] assumed that each software solution has a structural hierarchy consisting of the following levels: system of systems, system(s), subsystem group, subsystem(s), and class(es). Typically, use cases exist in all levels except the class level and use cases at different levels have different complexities. The method determines typical adjacent factors between the architectural levels as detailed in table V.

Having the system size calculated in LOC, by propagating the adjacent factors of the architectural levels defined in table V, COCOMO and Putnam's models [4,6] can be used to estimate the required development effort and time

TABLE I UCP ACTOR TYPES AND COMPLEXITY WEIGHTS

Actor Type	Complexity Weighting Factor
Simple	1
Average	2
Complex	3

TABLE II UCP ACTOR TYPES AND COMPLEXITY WEIGHTS

Use Case Type	No. of Transactions	Complexity
Simple	<= 3	1
Average	4 to 7	2
Complex	>= 7	3

TABLE III UCP TECHNICAL COMPLEXITY FACTORS

Technical Factor	Weight
Distributed System	2
Response Objective	2
End User Efficiency	1
Complex Processing	1
Reusable Code	1
Easy to Install	0.5
Easy to Use	0.5
Portable	2
Easy to Change	1
Concurrent	1
Security Features	1
Access for Third Parties	1
Special Training Required	1

TABLE IV UCP ENVIRONMENT FACTORS

Environmental Factor	Weight
Familiar with RUP	1.5
Application Experience	0.5
Object Oriented Experience	1
Lead Analyst Capability	0.5
Motivation	1
Stable Requirements	2
Part Time Workers	-1
Difficult Programming Language	2

TABLE V SMITH'S ARCHITECTURAL LEVEL ADJACENT FACTORS

Architectural Level	Adjacent Factor
Operation Size	70 LOC
Number of operations per class	12
Number of classes per subsystem	8
Number of subsystems per subsystem group	8
Number of subsystem group per system	8
Number of subsystem group per system	8
Number of systems per system group	8

C. *Object points extraction using use case model method*

OP is one of the size metrics that fits the early prototyping stages of software development. Also, OP was developed to cope with the visual widgets of Fourth Generation Languages (4GLs) and Integrated Computer Aided Software Engineering (ICASE) environments [4]. OP is mainly concerned with producing a reliable early count of the application (object) points being the sum of the adjusted number of screens, reports, and Third Generation Language (3GL) modules that are expected to be developed to supplement the 4GL code. Moreover, OPs do not relate to object oriented concepts such as inheritance, encapsulation, etc. but are more closely dependent on the user interface of the system being developed.

A use case model of an anticipated system describes who will use the system, user-system interaction scenarios, and the interrelationship between them [10]. These user-system interaction scenarios represent the main input to create the system user interface prototypes. Several methods and techniques have been developed to derive, model, and design user interface from use case models such as object modelling and user interface design [5]. Thus, this method investigates the applicability of use case models to count the system OP [2]; and consequently, estimate the development effort at an early stage in the software development life cycle. The detailed rationale, workflow, and example of the proposed OP extraction using use case model method are presented in [2].

Although several studies [3] aimed at synthesising the above use case model-based estimation methods, more knowledge is needed about the contexts in which they can be applied and how it could be adapted to local environments to improve the estimation process. In addition, the effect of other use case model attributes (e.g. use case relationships, preconditions, and postconditions) on their estimate's accuracy needs to be investigated [4]. More importantly, the literature does not seem to have any trace of validation of their accuracy in the different phases of the diverse software development life cycles. Also, it is apparent that most of the proposed approaches cannot be applied in the early stages of software development when estimation is needed for feasibility studies purposes. Therefore, this research builds on the above software cost estimation methods to investigate the applicability and reliability of use case model, being a common system model between the different stakeholders in the most recent software development life cycles, as a basis for software cost estimation in the very early stages of software development.

III. The Proposed Linear Model

A multi-phase Function Points (FP) to Object Points (OP) conversion approach has been proposed to infer the OP information for 66 ISBSG multi-organizational historical projects [1]. Consequently, the reliability of the defined FP-OP relationship has been evaluated by assessing the resulting OP attributes against the actual ISBSG FP and effort attributes. The evaluation of the conversion approach empirical results showed high correlation, 88%, between the OP estimated and FP actual efforts that generally supports the reliability of the defined FP-OP elements mapping and consumption schemes. Furthermore, a high correlation, 87%, has been found between the calculated AOP and unadjusted/adjusted FP. Further work has been carried out to integrate the resulting FP-OP conversion models with earlier work [2] to build use case based historical projects database.

The resulted use case historical projects database is then utilized in an independent study devoted to investigate the relationship between the actual effort and the resulted number of Use Cases (UC) for each project. This has led the research team to discover a hidden relationship between them. It has been found that the ISBSG actual effort is highly correlated, ***94%***, to the number of UCs. Hence, the fitness, R^2, of several multiple regression linear and exponential models to represent the relationship between them has been investigated. It has been found that both types of models have similar fitness in the corresponding experiments data. Therefore, linear models of general form:

$$\text{Effort} = (\text{Cons}_1 * \text{NoOfUCs}) + (\text{Cons}_2 * Z_1) + (\text{Cons}_3 * Z_2) + (\text{Cons}_4 * Z_3) + (\text{Cons}_5 * Z_4) + \text{Cons}_6$$

have been used to represent the relationship between the ISBSG actual effort and number of UCs where Z_1, Z_2, Z_3, and Z_4 are project size and programming language generation indicator variables as summarized in table VI. Correspondingly, table VII summarizes the parameters of the resulting models in the different experiments data. Experiment performed on basis of three points estimates approach.

TABLEVI PROJECT SIZE AND PROGRAMMING LANGUAGE GENERATION INDICATORS

Indicator Variable	Value	Productivity Factor
Z_1	0 or 1	Small Size Project
Z_2	0 or 1	Medium Size Project
Z_3	0 or 1	Third Generation Language Project
Z_4	0 or 1	Fourth Generation Language Project

IV. Critical and comparative evaluation

The validation of the proposed linear use case based estimation model has been performed using the FP-OP-UC projects data. First, three points, worst, expected, and best estimates have been adopted to utilise the use case converted data to estimate the effort of the corresponding projects using the proposed model. Consequently, the generated estimates have been evaluated to assess the overall accuracy and reliability of the proposed model compared to the actual effort and established software cost estimation models [4,6], respectively.

The recommended algorithmic models to be used in the literature are summarized in table VIII based on the performed statistical studies reported in table VII. Correspondingly, the curve fitness of the recommended worst, best and, expected case models are depicted in figures 1, 2, and 3, respectively.

Table IX depicts the results of a sample experiment that evaluates the proposed model. It has been found that the range of this initial estimate varies from ***43%*** to ***55%*** of the actual effort. This error percentage at the early stages of software development is compared with other well-known models, e.g. COCOMO II, that tend to generate estimates with ± 400% error percentage in such early stages of the software development life cycle [4]. The low level performance investigation of the embodied size and generation of programming language showed that the higher the generation of programming language used, the more accurate the effort estimate.

TABLE VII ALGORITHMIC NUMBER OF USE CASES BASED SCE MODELS

		$Cons_1$	$Cons_2$	$Cons_3$	$Cons_4$	$Cons_5$	$Cons_6$	R	R^2	t
Experiment 1	Worst Case Estimate	1.47	26.44	18.34	-26.75	-28.54	7.72	0.93	0.87	9.64
	Best Case Estimate	1.51	34.5	18.99	-13.42	-21.06	-11.98	0.96	0.92	13.24
	Expected Case Estimate	1.56	33.93	20.61	-12.37	-18.54	-12.79	0.95	0.91	12.11
Experiment 2	Worst Case Estimate	1.48	27.83	19.44	-24.86	-26.59	5.45	0.93	0.87	9.43
	Best Case Estimate	1.51	34.93	19.23	-12.82	-20.55	-12.73	0.96	0.91	13.19
	Expected Case Estimate	1.58	35.11	21.44	-10.47	-16.78	-15.30	0.95	0.90	11.99
Expérimente 3	Worst Case Estimate	1.64	28.05	18.41	-31.18	-34.67	10.21	0.93	0.87	9.39
	Best Case Estimate	1.51	34.5	18.99	-13.42	-21.06	-11.98	0.96	0.92	13.24
	Expected Case Estimate	1.65	34.70	20.39	-14.67	-21.69	-11.34	0.95	0.90	11.97
Experiment 4	Worst Case Estimate	1.65	29.42	19.49	-29.37	-32.81	8.05	0.93	0.86	9.18
	Best Case Estimate	1.51	34.93	19.23	-12.82	-20.55	-12.73	0.96	0.91	13.19
	Expected Case Estimate	1.65	35.59	21.16	-13.27	-20.28	-13.10	0.95	0.90	11.85

TABLEVIII RECOMMENDED ALGORITHMIC NUMBER OF USE CASES BASED MODELS

Estimation Type	Algorithmic Model
Worst Case Estimate	Effort = (1.47 × NofUCs) + (26.44 × Z_1) + (18.34 × Z_2) + (-26.75 × Z_3) + (-28.54 × Z_4) + 7.72
Best Case Estimate	Effort = (1.51 × NofUCs) + (34.5 × Z_1) + (18.99 × Z_2) + (-13.42 × Z_3) + (-21.06 × Z_4) - 11.98
Expected Case Estimate	Effort = (1.56 × NofUCs) + (33.93 × Z_1) + (20.61 × Z_2) + (-12.37 × Z_3) + (-18.54 × Z_4) – 12.79

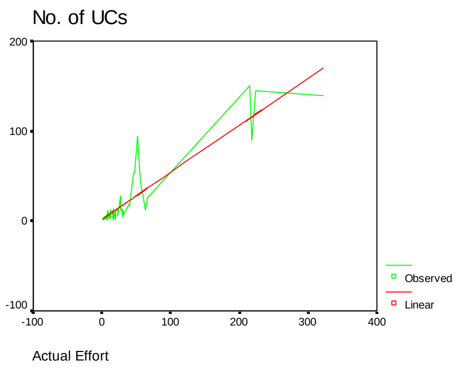

Fig.1 Worst Case Estimate Curve Fitness.

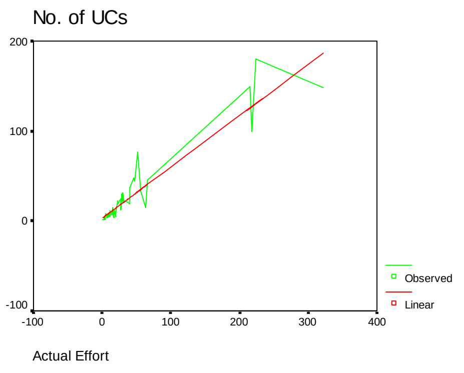

Fig.2 Best Case Estimate Curve Fitness.

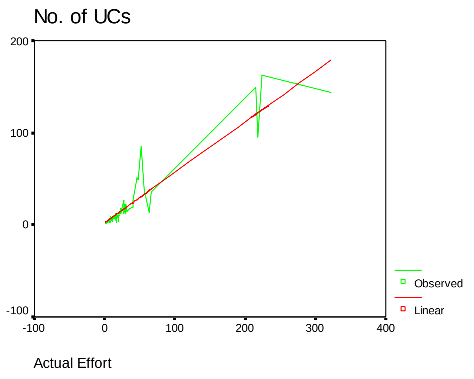

Fig. 3 Expected Case Estimate Curve Fitness.

V. Summary, conclusion, and future work

In an effort to investigate the appropriateness of use case models as a platform for accurate software cost estimation in the early stages of the software development life cycle, the number of use cases metric has been successfully used to develop new linear model as basis for providing an early yet rough estimation of the software development effort. In particular, the proposed approach has demonstrated the ability to generate early software cost estimates with an accuracy of approximately ± ***50%*** of the actual effort.

Hence, it may be concluded that the accuracy of approximately 50% of the actual effort in the proposed model is superior to that of the well-known software cost estimation models such as COCOMO II [4] and SLIM [6], that were reported to generate estimates of 400% and 771%, respectively, of the actual effort when applied in the early stages of software development.

Finally, future extensions of the proposed linear use case model based software cost estimation model are planned to consider the effect of the diverse non-functional requirements on the resulted effort estimates.

References

[1] A. Issa, M. Odeh, and D. Coward, “Can Function Points Be Mapped To Object Points?” International Arab Journal of Information Technology, (Accepted and to appear).

[2] A. Issa, M. Odeh, and D. Coward, “Using Use Case Models To Generate Object Points.” In Proceedings of the IASTED International Conference on Software Engineering Austria. ACTA Press, 2005, pp.468-473.

[3] B. Anda, E. Angelvik, and K. Ribu, “Improving Estimation Practices by Applying Use Case Models.” In Proceedings Product Focused Software Process Improvement 4th International Conference, PROFES 2002.,9-11 Dec Rovaniemi, Finland Springer-Verlag, 2002, pp. 383-397.

[4] Boehm, B., Horowitz, E., Madachy, R., Reifer, D., Clark, B., Steece, B., Brown, A., Chulani, S. and Abts, C., Software Cost Estimation With Cocomo II. Prentice Hall, 2000.

[5] C. Phillips, E. Kemp, and Sai Mei Kek, “Extending UML Use Case Modelling to Support Graphical User Interface Design.” In Proceedings of the 2001 Australian Software Engineering Conference, 27-28 Aug. 2001, Australia IEEE Comput. Soc, 2001, pp.48-57.

[6] F. Heemstra, “Software Cost Estimation.” Information and Software Technology, 34 (10), 1992, pp.627-639.

[7] J. Lewis, “Limits to Software Estimation”, Software Engineering Notes, 26 (4), 2001, p.54.

[8] J. Smith, “The Estimation of Effort Based on Use Cases” [online]. Rational Software, 2001, Available from: http://www.rational.com/products /whitepapers/finalTP171.jsp Accessed 1/15/2003.

[9] Karner, G., Resource Estimation for Objectory Projects. Objectory Systems, 1993.

[10] Kruchten, P., The Rational Unified Process : an Introduction. London: Addison-Wesley, 2002.