SOFTWARE EFFORT ESTIMATION MODEL BASED ON USE CASE SPECIFICATION

Xinguang Chen, Fengdi Shu
Lab for Internet Software Technologies, Institute of Software Chinese Academy of Sciences, Beijing, China

Ye Yang
Lab for Internet Software Technologies, Institute of Software Chinese Academy of Sciences, Beijing, China

Keywords: Use case, Software effort, Use case specification, Software estimation.

Abstract: Software effort estimation is essential for the project planning. Use case is widely used to capture and describe the requirements of customers and used as an index of software measurement and estimation. Based on the framework of traditional use case point estimation model, the paper presents UCSE, an effort estimation model based on use case specification. Firstly, the model abstracts factors influencing software effort from the use case specification and calculates the Use Case Weight, which is a kind of measurement of use cases size. Secondly, a function is constructed to translate the software size expressed by Use Case Weight to by software scale whose unit is kilo source line of code (KSLOC). Subsequently, effort estimation model COCOMO II is used to estimate the software effort according to the estimated software size measured by KSLOC. Compared with the traditional Use case point estimation model, UCSE model makes use of more relevant information and is more operable since it provides more concrete and objective references for the analysis and measurement of software effort factors in Use Case. What's more, the presented case study shows its results are more stable.

1 INTRODUCTION

How to select effective ways to improve accuracy of software cost estimation has been the key in the planning stage of software development. Long software development cycle, many influencing factors, emergencies and accidental events during the software process lead to that people usually make decision subjectively. That results in the objectivity and accuracy in software cost estimation. In 2004, Standish Organization statistics showed that among more than 50,000 software projects, the ratio of projects which could be completed is 29%, the ratio of projects which could be questioned is 53%, the ratio of projects which could be failed or cancelled is 18%, and the ratio of projects which could be finished but exceeded is 40% (Standish, 2004). It is widely believed that the main reason is that people are lack of software effort estimation.

From the 1860s, software effort estimation makes strong progress. At the beginning of research, researchers have constructed the software effort estimation model according to the characters of software development and simple algorithm, such as the SDC linear model (Boehm, 2005). Barry W. Boehm put forward the COCOMO 81 (Boehm, 1981) in 1981 and the COCOMO II (Boehm, 2000) in 2000. The COCOMO series establish the relationship between software effort and software scale, and adjust the function by a series of cost-driving factors, which are the popular model.

As more and more software projects use the unified modelling language (UML) to develop, use case model is more and more used to capture and describe the requirements of software. According to the research of Neill in 2003, there are about 50% projects adopting use cases or scenes to describe the functional requirement (Neill, 2003). In 1993, the first effort estimation method using use cases (Karner, 1993) was proposed by Dr. Karner, which was called use case point (UCP). Based on the UCP, there were many related studies, which could be generally classified to the 3 groups.

Chen X., Shu F. and Yang Y..
SOFTWARE EFFORT ESTIMATION MODEL BASED ON USE CASE SPECIFICATION.
DOI: 10.5220/0003463302260231
In *Proceedings of the 6th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2011), pages 226-231
ISBN: 978-989-8425-57-7

Firstly, many case studies were used to validate UCP. In 1999, John Smith from IBM proposed the estimation framework based on the use case (Smith, 1999). The framework consists of five-level structure which includes class, subsystem, subsystem group, system, and multisystem.

Secondly, many researches were focused on the improvements of UCP. In 2005, Carroll added a risk parameter to the UCP (Carroll, 2005) to improve the accuracy of UCP.

Thirdly, many researchers use UCP in different specific areas. Nageswaran built the map from use cases to test cases by using UCP and estimate the test effort (Nageswaran, 2001).

To extract more concrete and objective references for the analysis and measurement of software effort factors in Use Case, the paper proposes Use Case Specification Estimation model (UCSE). The paper is organized as follows: section 2 discusses related and previous work; section 3 elaborates UCSE and section 4 presents a case study of the model; and section 5 concludes the paper.

2 RELATED AND PREVIOUS WORK

2.1 Use Case Point Model

Firstly, the model calculates weight of actor and weight of use case to get the unadjusted use case points (UUCP); secondly, the model adjusts UUCP using technical factors and environment factors to get UCP; finally, it establishes the relationship between UCP and effort. Figure 1 shows the procedure of UCP method.

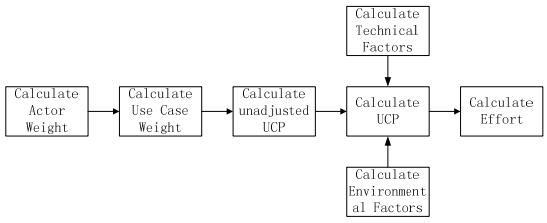

Figure 1: Use case point model.

1）Unadjusted Actor Weight

The model classifies actors into 3 types: Simple, Normal and Complex. We count the number of actors in each type and multiply weighting factor (WF) which the actor of every type corresponds to. The unadjusted actor weight (UAW) is the sum of all the results above. The symbol n is the quantity of use cases.

$$UAW=\sum_{i=1}^{n} Actor_i*WF_i \quad (1)$$

2）Unadjusted Use Case Weight

The model classifies use cases into 3 types according to the scale of use case: Simple, Normal and Complex. We count the number of use cases in each type and multiply weighting factor (WF) which the use case of every type corresponds to. The unadjusted use case weight (UUCW) is the sum of all the results above. The symbol n is the quantity of use cases.

$$UUCW=\sum_{i=1}^{n} UseCase_i*WF_i \quad (2)$$

3）Unadjusted Use Case Points

UAW plus UUCW equals unadjusted use case points (UUCP).

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

4）Other factors

The model uses technical complexity factors (TCF) and environmental complexity factors (ECF) to adjust the UUCP. TCF reflects the internal properties of software, such as security, reusability etc. ECF reflects the external properties of software, such as personnel experience etc. Every factor has a value of 0 to 5. The value reflects the degree of every factor and judged subjectively. The equations of TCF and ECF are shown below:

$$TCF=0.6+0.01*(\sum_{13}^{i=1} T_i*Weight_i) \quad (4)$$

$$ECF=1.4+(-0.03)*(\sum_{8}^{i=1} E_i*Weight_i) \quad (5)$$

We can calculate the UCP and Effort using Equation (6) and (7):

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

$$Effort=UCP*Productivity\ Factor \quad (7)$$

Productivity Factor is a coefficient of software project which is calculated by historical data. Dr. Karner thought that productivity factor equalled to 20 man-hours.

The traditional UCP considered sufficiently information of use case in software development and some impacts, but there are three problems: Firstly, there is no definite reference to classify the actor and use case. It is difficult to use. Secondly, some

information in use case which is also relevant with software effort, such as extended event, is neglected. Thirdly, the value of technical and environmental factors is subjective.

2.2 COCOMO II

COCOMO II was proposed by Barry Boehm in 2000. The model adjusts and updates the cost-driving factors. EM is a multiplier of software effort; SF is an index of scale factor; A and B are the parameters whose value is adjusted by historical data.

$$\text{Effort}=A*(\text{KSLOC})^{E}*\prod_{i=1}^{n}\text{EM}_i \quad (8)$$

$$E=B+0.01*\sum_{j=1}^{5}\text{SF}_j \quad (9)$$

To COCOMO II：A＝2.94, B=0.91

Every cost-driving factor is corresponding to a value of EM. Every value of EM should be put into the function above to calculate software effort, cost and schedule. The key of COCOMO II is the Kilo source line of code (KSLOC). The accuracy of effort depends largely on the accuracy of software scale. Boehm thought the important input in COCOMO II is KSLOC (Boehm, 2000).

3 UCSE MODEL

UCSE consists of several parts. First, based on the framework of the traditional UCP, we analyse the structure of use case specification, calculate the use case weight (UCW) according to the document of use case specification, and establish the relationship between UCW and KSLOC. Subsequently, software effort is calculated by COCOMO II based on the estimated size in KSLOC. Use case specification is used because it has full information to make up the second problem of UCP presented in section 2.1.

The overview of UCSE model is shown in Figure 2 and its main components are introduced in following.

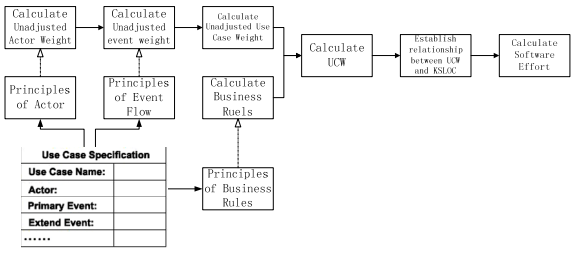

Figure 2: UCSE Model.

3.1 Unadjusted Actor Weight

The type of actors is important. In the use case specification, actors could be easy to classify. According to UCP and use case specification, the actor factor is described in Table 1, which including its classes and corresponding classification rules and weights.

Table 1: Actor Factor.

Actor Class	Rule	Weight
Simple	Only one actor	1
Medium	Two actors	2
Complex	Three or more actors	3

As shown in formula 10, unadjusted actor weight (UAW) equals to the sum of the product of actor class (AC) and class weight (CW), where *n* is the quantity of use cases.

$$\text{UAW} = \sum_{i=1}^{n}\text{AC}_i*\text{CW} \quad (10)$$

3.2 Event Weight

Event flow in use case specification describes the scene of use case and reflects the steps of software execution. Event flow is the main part in use case specification. It has two types: primary event and extended event. Effort is influenced differently by two types. Primary event flow is important to scale of use case, while extended event flow occurs in some abnormal or special occasions. The traditional UCP neglected the extended event. That may produce some errors.

UCSE model provides classification rules of the primary event and each class of primary event has its weight, which is shown in Table 2:

Table 2: Primary Event Factor.

Event Type	Rule	Weight
Simple	The number of primary event flows ≤ 3	10
Medium	The number of primary event flows is between 4 and 7	15
Complex	The number of primary events flows ≥ 7	20

As shown in formula 11, unadjusted primary event weight (UPEW) equals to the sum of the product of primary event class (PEC) and class weight (CW), where *n* is the quantity of use cases.

$$\text{UPEW} = \sum_{i=1}^{n}\text{PEC}_i*\text{CW} \quad (11)$$

UCSE model provides classification rules of the extended event and each class of extended event has its weight, which is shown in Table 3:

Table 3: Extended Event Factor.

Event Type	Rule	Weight
Simple	The number of extended event flows ≤ 5	5
Complex	The number of extended event flows ≥ 5	7

As shown in formula 12, unadjusted extended event weight (UEEW) equals to the sum of the product of extended event class (EEC) and class weight (CW), where *n* is the quantity of use cases.

$$UEEW = \sum_{i=1}^{n} EEC_i * CW \quad (12)$$

As shown in formula 13, unadjusted event weight (UEW) equals to the sum of UPEW and UEEW.

$$UEW=UPEW+UEEW \quad (13)$$

3.3 Unadjusted Use Case Weight

Actor and event are the most important parts in use case specification, so unadjusted use case weight (UUCW) the sum of UAW and UEW.

$$UUCW=UAW+UEW \quad (14)$$

3.4 Business Rules

Traditional UCP has 13 TCFs and 8 ECFs, which reflect two points in the software development: one is restraint of actors; the other is the restraint of operation rules. In the UCP, we identify these values of factors subjectively. Business rules (BR) in the document of use case specification reflect directly the status of persons and technology and is more objective to reflect the real status in software development.

BR has two types: one is the global rules, the other is special rules. Global rules usually are related to every use case. For example, an actor needs to be authorized to execute a use case, so the execution of use case is corresponding to the level of Authority Classes, or operations of users need to be recorded in the system and so on. Special rule only exists in the extended event. It cannot change as external environment changes. For example, an order at least has one type of goods, and the quantities of goods need to be less than 5 and so on. Besides, data verification belongs to special rule. For example, the number of identification card must be 15 or 18 digits, and zip code must be 6 digits and so on. We classify special rules into Standard, Medium and Complex, which is shown in Table 4:

Table 4: Business Rules.

Type		Rule	Weight
Global Rules		global rules exist	1.05
Special Rules	Simple	No special rule	1.0
	Medium	≤ 50 percent of use cases has special rules	1.5
	Complex	≥ 50 percent of use cases has special rules	2

Use case weight (UCW) equals to the product of UUCW and BR.

$$UCW=UUCW*BR \quad (15)$$

3.5 Relationship between UCW and KSLOC

KSLOC measures the scale of software, and UCW measures the scale of use case. The UCSE model establishes the relationship between KSLOC and UCW.

Regression is a common statistical method to determine the quantitative relationship between two or more variables. Regression has two types: one is linear regression, the other is non-linear regression. Exponential function and logarithmic function are common non-linear function. We will determine the function between KSLOC and UCW by using data of 14 projects. The function is shown as:

$$KSLOC=f(UCW) \quad (16)$$

3.6 Effort Function

We input KSLOC into COCOMO II and the function of effort is shown as:

$$Effort=A*(f(UCW))^{E}*\prod_{i=1}^{n} EM_i \quad (17)$$

$$E=B+0.01*\sum_{j=1}^{5} SF_j \quad (18)$$

4 CASE STUDY

We carry out a case study on 14 software projects based on their use case specification. The first 10 projects (EP1 to EP10) are used to make regression

analysis on KSLOC and UCW. The other four (VP1 to VP4) are used to validate the UCSE model.

EP1 is a human resource system; EP2 is a tool which integrating some estimation tools; EP3~EP10 are different versions of a software process management platform. The data of these projects are shown Table 5:

Table 5: Data of Projects.

Project	Number of Use case	UCW	KSLOC
EP1	5	125	6.034
EP2	7	199.5	13.52
EP3	60	3307.5	169.7
EP4	164	4192	250
EP5	186	4110	194
EP6	189	3687.08	191
EP7	195	5471.55	228
EP8	222	6575.1	294.8
EP9	312	13657.8	618
EP10	278	16989	824.8

4.1 Regression Analysis

We use UCW and KSLOC to make regression analysis. The result is shown in Figure 3:

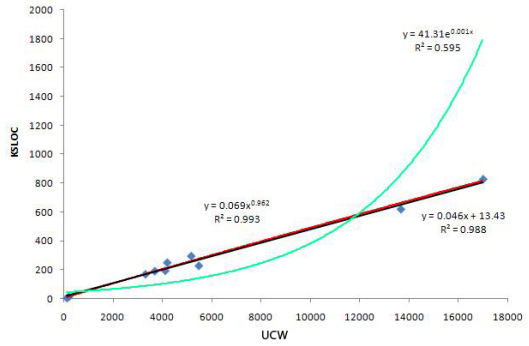

Figure 3: Regression Curve.

The result of the three regression equations is shown in Table 6:

Table 6: Regression Equation.

	Regression Equation	R^2
Linear regression	$y = 0.046x + 13.43$	0.988
Indicial Regression	$y = 41.31e^{0.001x}$	0.595
Power Regression	$y = 0.069x^{0.962}$	0.993

The result shows that the power function regression equation is very simple and ideal. The coefficient is highest (0.993). Therefore, UCSE model uses the power function. The graph of power function regression equation is shown in Figure 4:

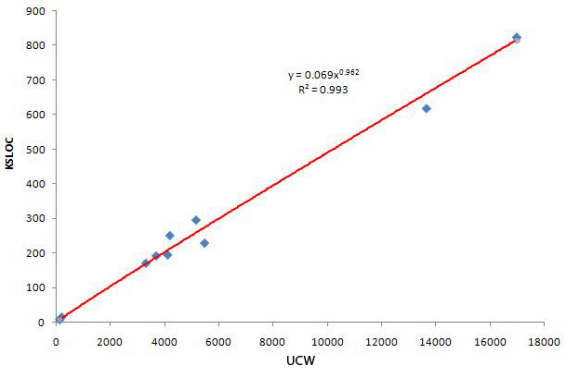

Figure 4: Graph of power regression equation.

So the equation is shown as:

$$KSLOC=0.069*UCW^{0.962} \quad (19)$$

4.2 Result Analysis

VP1-VP4 was developed based on their use case specifications. VP2 is a customized software process management platform. The four projects were estimated by UCP and UCSE model respectively to get estimated effort. We calculate relative error, average relative error and standard deviation to compare two models. In Table 7, EE is the Estimated Effort and RE is the Relative Error. The result is shown in Table 7:

Table 7: Estimation Results of UCSE and UCP.

	Real Effort	UCSE Model		UCP Model	
		EE	RE	EE	EE
VP1	2281.39	2840.24	0.24	3114.40	0.37
VP2	3426.65	3865.64	0.13	3931.20	0.15
VP3	2625.58	2505.10	-0.05	2479.80	-0.06
VP4	3110.75	3697.62	0.19	3587.90	0.15
Average Relative Error		0.1519		0.1803	
Standard Deviation		0.0739		0.1135	

From Table 7, we can see that the estimation results of UCSE model are better than those of UCP model except as for VP4. The average relative error and standard deviation of UCSE model are less than the UCP model. The UCSE model analyses the document of use case specification sufficiently, includes the extended event which was neglected by UCP model, improves the way to identify the weight of actors and event flows, and uses business rules instead of TCFs and ECFs. UCSE model improves

the operability so that the results are more accuracy and steady.

5 CONCLUSIONS

Use case is a common tool and penetrates through the software development. Use case specification is the document of use case and is easy to reflect and describe the software process. The paper proposes UCSE model to improve the accuracy of estimating and be more operable since it provides more concrete and objective references for the analysis and measurement of software effort factors in use case. The result of UCSE model is better than traditional model. However, the usage of the model is related to the form of the use case specification, which lacks uniform standard. So the future work includes how to overcome this limit and enhance the applicability of UCSE model.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China under Grant Nos.60873072， 61073044, and 60903050; the National Science and Technology Major Project; the National Basic Research Program under Grant No.2007CB310802; CAS Innovation Program.

REFERENCES

The Standish Group. (2004). *The 3rd Quarter Research Report.* (n.d.), from http://www.standishgroup.com.

Boehm B. W., Valerdi R. (2005). *Achievements and challenges in software resource estimation.* Retrieved May 13, 2005, from http://sunset.usc.edu/publications/TECHRPTS/2005/usccse2005-513/usccse2005-513.pdf

Boehm. B. W. (1981). *Software Engineering Economics* (1st ed.). NJ: Prentice-Hall

Boehm. B. W., Madachy R. and Steece B. (2000). *Software Cost Estimation with COCOMOII* (1st ed). NJ: Prentice-Hall

Guarv Karner. (1993). Resource Estimation for Objectory Projects. *Objective Systems SF AB (Copyright owned by Rational Software)*

John Smith. (1999). The estimation of effort based on use cases. *Rational Software White Paper*

Neill C. J., Laplante P. A. (2003). Requirements engineering: the state of the practice. IEEE. 20 (11). 40-45, doi:dx.doi.org/10.1109/MS.2003.1241365.

Carroll E. R. (2005). Estimating software based on use case points. *Proceedings of the Companion to the 20th annual ACM SIGPLAN conference.* 257-265. doi:dx.doi.org/10.1145/1094855.1094960.

Nageswaran S. (2001). Test effort estimation using use case points. *Proceedings of the 14th Internet & Software Quality Week.*