Information Systems International Conference (ISICO), 2 – 4 December 2013

A Web-Based Tool Support for Automating Software Effort Estimation

Zhamri Che Ani*, Shuib Basri**
* School of Computing, Universiti Utara Malaysia, Malaysia.
** Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Malaysia.

Keywords:

Software effort estimation
Use case points
Google web toolkit

ABSTRACT

Software effort estimation has become one of the most important concerns of software industries. Developers use effort estimation to ensure the quality of software systems can be delivered within time and budget. Current development tools for effort estimation based on UCP model help with automating the effort required for a software project. However, they offer no opportunity for future extension as a web-based application. Therefore, this paper describes a new web-based tool support for automating software effort estimation. The tool has been developed based on use case points model using rich internet application technology to improve the reusability of software effort estimation applications.

Copyright © 2013 Information Systems International Conference.
All rights reserved.

Corresponding Author:

Zhamri Che Ani,
School of Computing,
Universiti Utara Malaysia,
06010 UUM Sintok, Kedah, Malaysia.
Email: zhamri@uum.edu.my

1. INTRODUCTION

Software effort estimation is a process to gain a general understanding of the effort required to develop a software system or software product. It has been focused by many researchers over the past 40 years [1] and nowadays, it has become one of the most important concerns of software industries [2,3,4]. There are a number of models that have been proposed as basis of estimating effort, schedule and cost of a software project [5,6]. These models, which include the Parametric Review of Information for Costing and Evaluation—Software (PRICE-S), Software Evaluation and Estimation of Resources—Software Estimating Model (SEER-SEM), Putnam Software LIfecycle Management (SLIM), Constructive Cost Model (COCOMO), Use Case Points (UCP), ObjectMetrix, and many more. However, some estimation methods are not designed to work well with object-oriented technology that introduces inheritance and actively encourages reuse strategies.

Reusability is the highest priority that needs to be considered before developing good software application. Without reusability, software applications are very hard to maintain or extent [7,8,9,10]. Existing software effort estimation applications were developed by using various programming languages [11]. However, Microsoft Excel is the most popular tool applied by many software developers to produce accurate effort estimation, particularly by using UCP approach. Therefore, it is impossible to extend the development as a web-based application that can be run in multiple platforms.

In this paper, we present an estimating tool, Ext-UCP, a new web-based tool support for automating software effort estimation. The tool has been developed based on use case points model using rich internet application technology to improve the reusability of software effort estimation applications. The remainder of the paper is structured as follows. Section 2 describes background of use case points model. Section 3 presents the tool support called Ext-UCP, how it was developed, and the functionality in details. Section 4 includes conclusion and suggestion for future work.

2. BACKGROUND OF USE CASE POINTS MODEL

Use Case Points (UCP) is a software sizing and estimation method adopted from the classic Function Point (FP) method in solving the specific needs of object oriented systems based on use cases [12,13]. It was developed by Gustav Karner at Objectory Systems [14]. The accuracy of the UCP estimations was compared to expert estimates by experienced software developers, and the results showed that estimated effort for each project was quite close to the actual estimates [15]. The results indicate that the UCP can be successfully used to estimate the development of software effort. Unlike traditional approaches, UCP provides the ability to estimate at the early stages of a software project [13].

In general, projects with large, complicated use cases take more effort to design and implement than small projects with less complicated use cases. The necessary steps to generate the estimate based on the UCP method are as follows [12,14,16]:

2.1. Determine and compute the Unadjusted Use Case Points (UUCP)

This can be done by calculating Unadjusted Use Case Weight (UUCW) and Unadjusted Actor Weight (UAW). UUCW can be obtained by defining use cases as simple, average or complex, depending on the number of transactions in the use case description, including secondary scenarios. A transaction is an event that occurs between an actor and the target system. Alternatively, the number of transactions can also be done by counting the use case steps. If the use case contains more than seven transactions, it is considered complex. On the other hand, the use case is considered simple if it contains less than four transactions. The UUCW is calculated by counting the number of use cases in each category, multiplying each total by its specified weighting factor, and then adding the products. The example of the UUCW calculation is shown in Table 1.

Table 1. Example of calculating unadjusted use case weight (UUCW)

Use case complexity	Number of transactions	Weight	Number of use cases	Product
Simple	3 or fewer	5	0	0
Average	4 to 7	10	29	290
Complex	More than 7	15	0	0
Total			29	290

UAW can be obtained by classifying the actors as simple, average, or complex. For example, a simple actor represents another system that communicate via a pre-defined API, an average actor can be either human beings or another system interacting though well-defined protocol such as TCP/IP, and a complex actor is a person interacting through GUI or a Web page. The UAW is calculated by counting the number of actors in each category, multiplying each total by its specified weighting factor, and then adding the products. The example of the UAW calculation is shown in Table 2.

Table 2. Example of calculating unadjusted actor weight (UAW)

Actor type	Weight	Number of actors	Product
Simple	1	0	0
Average	2	0	0
Complex	3	5	15
Total		5	15

Then, the UUCP is computed by adding the UUCW and the UAW. For the data used in Tables 1 and 2, the UUCP = 290 + 15 = 305. The UUCP is unadjusted because it does not account for the technical and environmental complexity factors (TCF and ECF**).**

2.2. Determine and compute the Technical Complexity Factors (TCF)

The second step is to determine and compute the Technical Complexity Factors (TCF). For each project, the technical factors are evaluated by the development team and assigned a perceived complexity value between zero and five. The perceived complexity factor is subjectively determined by the development team's perception of the project's complexity. For example, concurrent applications require more skill and time than simple web-based applications. A perceived complexity of zero means the technical factor is irrelevant for this project, three is average, and five is strong influence. It is advisable to use three when in doubt condition. Each factor's weight is multiplied by its perceived complexity factor to produce the calculated factor. The calculated factors are summed to produce the Total Technical Factor. Table 3 shows the example of technical factors calculation.

Table 3. Example of project's technical complexity factors (TCF) calculation

Factor	Description	Weight	Assessment	Impact
T1	Distributed system required	2	3	6
T2	Response time is important	1	3	3
T3	End user-efficiency	1	3	3
T4	Complex internal processing required	2	3	6
T5	Reusable code must be a focus	1	0	0
T6	Easy to install	0.5	3	1.5
T7	Easy to use	0.5	3	1.5
T8	Cross-platform support	2	0	0
T9	Easy to change	1	3	3
T10	Highly concurrent	1	3	3
T11	Custom security	1	3	3
T12	Dependence on third-party code	1	0	0
T13	User training	1	3	3
	Total TFactor			33

Based on Table 3, TCF is calculated by using the following formula:

TCF = 0.6 + (0.01 x TFactor)
= 0.6 + (0.01 x 33)
= 0.6 + 0.33 = 0.93

2.3. Determine and compute the Environmental Complexity Factors (ECF)

The third step is to determine and compute the Environmental Complexity Factors (ECF). Larger values for the environmental factor will have a greater impact on the UCP equation. A value of one means the factor has a strong negative impact for the project; three is average; and five means it has a strong positive impact. A value of zero has no impact on the project's success. For example, team members with little or no motivation for the project will have a strong negative impact (one) on the project's success while team members with strong object-oriented experience will have a strong, positive impact (five) on the project's success. Each factor's weight is multiplied by its perceived impact to produce its calculated factor. The calculated factors are summed to produce the Environmental factor. Table 4 shows the example of environmental factors calculation.

Table 4. Example of project's environmental complexity factors (ECF) calculation

Factor	Description	Weight	Assessment	Impact
E1	Familiarity with the project	1.5	3	4.5
E2	Application experience	0.5	3	1.5
E3	Object-oriented programming experience	1	3	3
E4	Lead analyst capability	0.5	3	1.5
E5	Motivation	1	3	3
E6	Stable requirements	2	3	6
E7	Part-time staff	-1	0	0
E8	Difficult programming language	-1	0	0
	Total EFactor			19.5

Based on Table 4, ECF is calculated by using the following formula:

ECF = 1.4 + (-0.03 x EFactor)
= 1.4 + (-0.03 x 19.5)
= 1.4 + (-0.585) = 0.815

Then, the adjusted UCP is obtained by applying the following formula:

UCP = UUCP x TCF x ECF
= 305 x 0.93 x 0.815
= 231.17475

2.4. Determine the Productivity Factors (PF)

The forth step is to determine the Productivity Factor (PF). Actually there is no consistent project dataset presenting UCP data as the ISBSG (International Software Benchmarking Standards Group) does for some other methods [11]. The reason behind this is due to the fact that use case specification varies for each project. UML does not explain in details about how to structure the use case models or how to document each use case [13]. But based on the previous studies, some suggestions proposed by the experts are 20 hours/UCP [14], an average of 20-28 hours/UCP [17], and range between 15 to 30 hours/UCP [12,13]. However, the chosen value is depending on the development team's overall experience. For instance, if it is a brand-new team, use a value of 20 for the first project [12,17].

2.5. Compute the estimated number of hours

The final step is to compute the estimated number of hours. Below is an example of the calculation based on 28 hours/UCP.

$$\begin{aligned}\text{Man-Hours} &= \text{UCP} \times \text{PF} \\ &= 231.17475 \times 28 \\ &= 6{,}472.893 \\ &= 6{,}473 \text{ hours}\end{aligned}$$

3. TOOL SUPPORT

Based on the UCP method described in section 2, a new tool support was developed to automate software effort estimation. The tool called Ext-UCP, is a web-based application that is aimed at providing automating software effort estimation based on use case points model. It has been built using Google Web Toolkit (GWT), which is among the most popular rich internet application technologies. GWT is a Java-based Ajax application development framework that allows developers to use Java programming language to build and maintain high performance JavaScript front-end applications quickly. Another reason why GWT was chosen because it can support most of browsers and operating systems [18], thus helping enhance the compatibility of web system. Unlike typical web applications development, writing code on the client side and the server side using GWT are much easier.

There are 12 use cases have been determined to explain about the functionality of Ext-UCP. Details of the conceptual functions provided by the tool are illustrated in Figure 1.

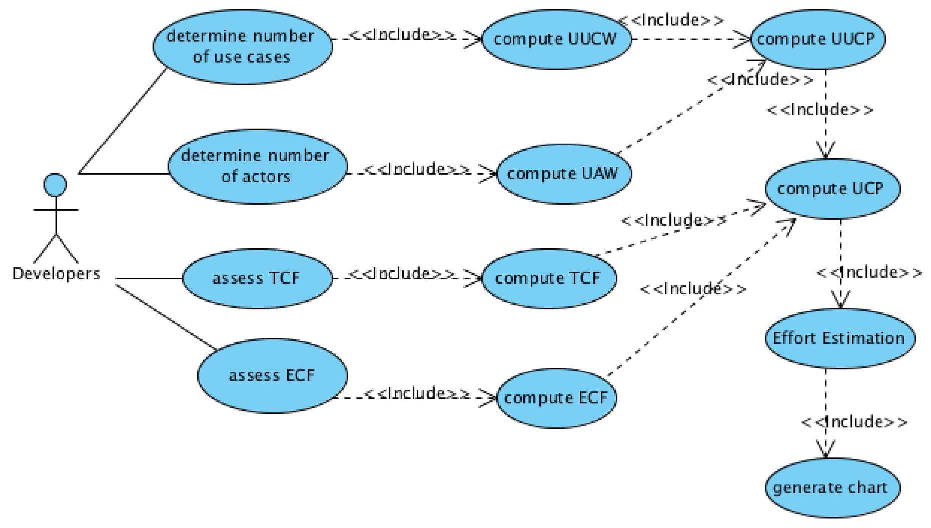
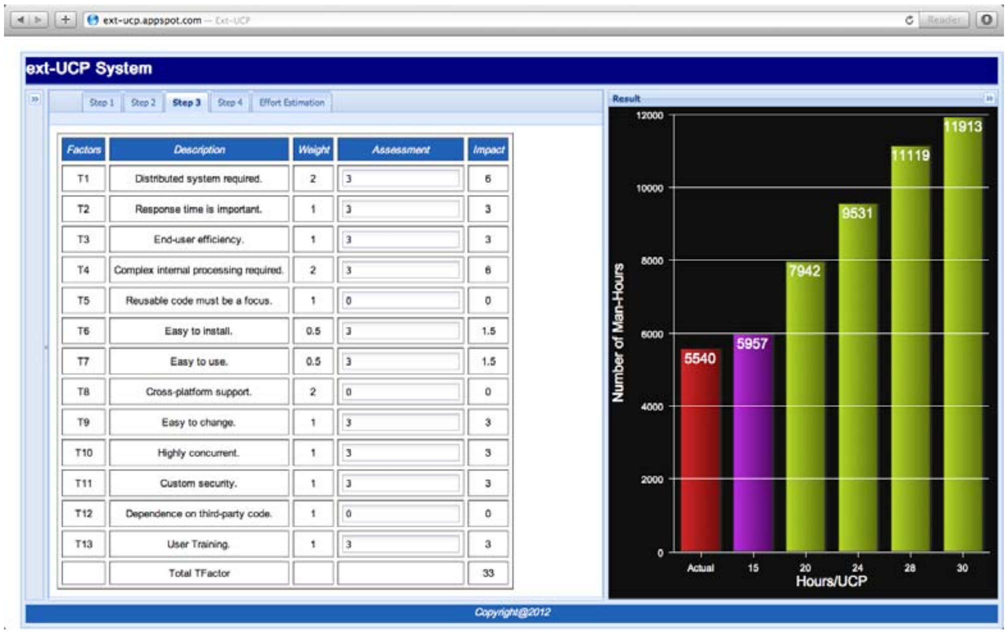


Figure 1. Use Case Diagram of Ext-UCP

Basic functions of Ext-UCP are to manage variables of UUCW, UAW, TCF and ECF. Each variable is defined and computed separately using weighted values, subjective values, and constraining constants. The weighted values and constraining constants were permanently set in the system based on the UCP model. The subjective values are determined by the development team based on their perception of the project's technical complexity and efficiency. Then, using the subjective values, the system will calculate the total values of UUCW, UAW, UUCP, TCF, ECF and UCP automatically. Finally, this tool support will propose the estimated hours needed for developing a software project based on the productivity factors. In our case, the productivity factors are hard coded to ensure the research-based results are well maintained.

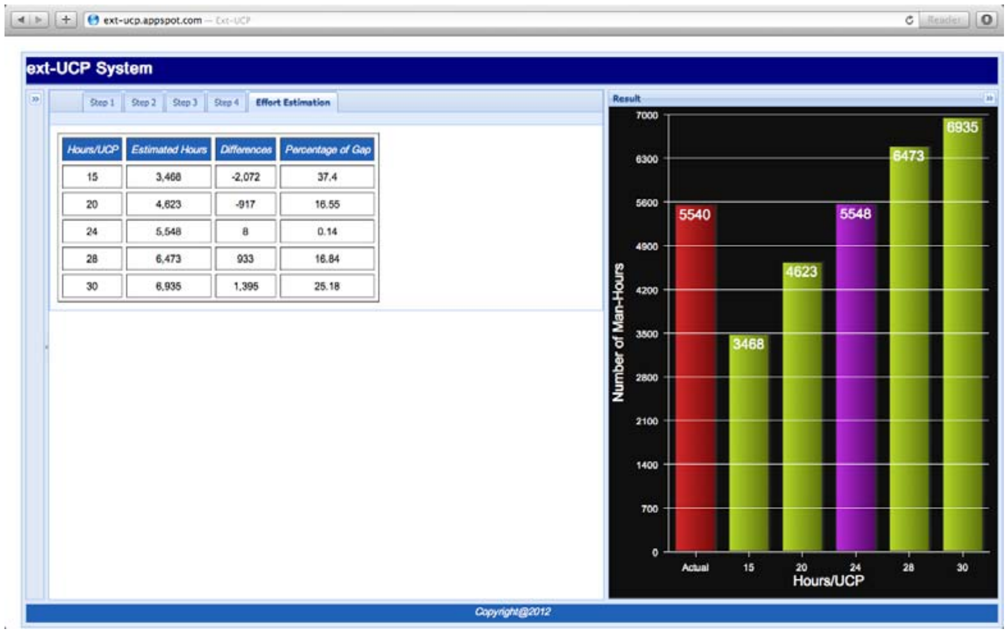
To ensure the accuracy of the system, this tool was developed using Test-Driven Development (TDD) approach. TDD is a style of development where every code of each use case must be tested to prevent errors in effort calculation. Basically, the main interface of the Ext-UCP is divided into four steps and results generation. Prior to computing the effort estimation, developers or development team must key few values of in the identified variables in step 1 until step 4 tabs through interactive user interfaces. Figure 2 shows how TCF is calculated based on the subjective values input by the development team. Details of the TCF calculation can be referred in section 2.2



Factors	Description	Weight	Assessment	Impact
T1	Distributed system required.	2	3	6
T2	Response time is important.	1	3	3
T3	End-user efficiency.	1	3	3
T4	Complex internal processing required.	2	3	6
T5	Reusable code must be a focus.	1	0	0
T6	Easy to install.	0.5	3	1.5
T7	Easy to use.	0.5	3	1.5
T8	Cross-platform support.	2	0	0
T9	Easy to change.	1	3	3
T10	Highly concurrent.	1	3	3
T11	Custom security.	1	3	3
T12	Dependence on third-party code.	1	0	0
T13	User Training.	1	3	3
	Total TFactor			33

Figure 2. Screenshot of the Third Step of Ext-UCP

As mentioned earlier, only UUCW, UAW, TCF and ECF need users interaction to fulfill the required values. The rest of the jobs will be handled by the tool automatically. Instead of having typical functions of effort estimation, the tool also provides colorful bar chart to assist development team for making quick decision. The chart is classified into three main colors. The red bar means the actual estimate made by development team. The actual estimation can be obtained from software development plan (SDP) where project gantt chart is normally used for project planning. The green bar means the estimated values generated by the system. The purple bar means the most accurate values proposed by the tool based on the smallest percentage of gap between the actual and the estimate generation. Referring to the screenshot of Ext-UCP result generation as shown in Figure 3, 24 hours/UCP is the best PF where the estimated value is so close to the actual effort. This is a unique feature of Ext-UCP where the best value of PF can be determined in each company. This value can be used as a basis for future effort estimation.



Hours/UCP	Estimated Hours	Differences	Percentage of Gap
15	3,468	-2,072	37.4
20	4,623	-917	16.55
24	5,548	8	0.14
28	6,473	933	16.84
30	6,935	1,395	25.18

Figure 3. Screenshot of Ext-UCP Result Generation

4. CONCLUSION AND FUTURE WORK

We have presented a web-based tool support for automating software effort estimation based on the use case points model. The tool was developed using GWT, a Java-based Ajax application development framework to ensure the reusability of the tool can be maintained easily. Unlike many other commercial estimation tools, the concepts and methods behind our tool are openly described and available for further investigation. Based on our experiments, the result of UCP calculation using this tool was absolutely accurate as discussed in section 2. Furthermore, the best value of productivity factor for each company can be determined as a basis for future effort estimation. We are hoping that this tool will give some benefits to the developers in improving the accuracy of software effort estimation.

We plan to further validate the tool by deploying it so that developers can immediately give their feedbacks. For information, this tool has been tested using Internet Explorer, Safari, Firefox and Google Chrome. But the best view is using Google Chrome. We also plan to include this tool with other estimation models to provide more options in estimating software project.

REFERENCES

[1] C. Jones, “Software cost estimation in 2002,” *The Journal of Defense Software Engineering,* vol. 15, no. 6, pp. 4–8, 2002.

[2] A. Trendowicz, J. Münch, and R. Jeffery, “State of the practice in software effort estimation: a survey and literature review,” *Software Engineering Techniques,* vol. 4980, pp. 232–245, 2011.

[3] F. Brooks Jr, “Three great challenges for half-century-old computer science,” *Journal of the ACM,* vol. 50, no. 1, pp. 25–26, 2003.

[4] M. Jørgensen and D. Sjøberg, “The impact of customer expectation on software development effort estimates,” *International Journal of Project Management,* vol. 22, no. 4, pp. 317–325, 2004.

[5] B. Boehm, C. Abts, and S. Chulani, “Software development cost estimation approaches: A survey,” *Annals of Software Engineering,* vol. 10, pp. 177–205, 2000.

[6] B. Boehm, “Software engineering economics,” *Software Engineering, IEEE Transactions,* vol. 10, pp. 4–21, 1984.

[7] M. Grand, *Java enterprise design patterns.* Wiley, 2002.

[8] C. Alexander, S. Ishikawa, and M. Silverstein, *A pattern language: towns, buildings, construction.* Oxford University Press, USA, 1977, vol. 2.

[9] P. Kuchana, *Software architecture design patterns in Java.* CRC Press, 2004.

[10] C. Lasater, *Design patterns.* Jones & Bartlett Learning, 2006.

[11] Ç. Gencel, L. Buglione, O. Demirors, and P. Efe, “A case study on the evaluation of cosmic-ffp and use case points,” in *3rd Software Measurement European Forum,* Rome, Italy, 2006.

[12] R. Clemmons, “Project estimation with use case points,” *The Journal of Defense Software Engineering*, pp. 18–22, 2006.

[13] G. Banerjee, “Use case estimation framework,” in *Annual IPML Conference.* Citeseer, 2004, pp. 1–12.

[14] G. Karner, “Resource estimation for objectory projects,” *Objective Systems SF AB*, vol. 17, 1993.

[15] B. Anda, H. Dreiem, D. Sjøberg, and M. Jørgensen, “Estimating software development effort based on use cases - experiences from industry,” *The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, vol. 2185, pp. 487–502, 2001.

[16] M. Damodaran and A. Washington, “Estimation using use case points,” *Computer Science Program. Texas–Victoria: University of Houston. Sd*, 2002.

[17] G. Schneider and J. Winters, *Applying Use Cases - A Practical Guide*, 2nd ed. Boston: Addison-Wesley, 2001.

[18] E. Burnette, “Google web toolkit–taking the pain out of ajax,” *USA: The Pragmatic Programmers LLC*, 2006.

BIBLIOGRAPHY OF AUTHORS

	Zhamri Che Ani received Master of Software Engineering from Universiti Teknologi Malaysia and Bachelor of Information Technology from Universiti Utara Malaysia. Currently, he is a PhD student at Universiti Teknologi Petronas, Malaysia. His research interests lie in software engineering particularly focuses on software effort estimation, software development, software design and design patterns. He previously worked for Malaysia Airlines System and successfully certified by Oracle Corporation.
	Shuib Basri is a senior lecturer at Universiti Teknologi Petronas, Malaysia and currently as deputy head of Computer and Information Sciences department. He is a member of software engineering research group and his research interests include software effort estimation, software quality and software process improvement. He has a PhD in Software Engineering from Dublin City University, Republic of Ireland, Master in Software System Engineering, University of Melbourne and Bachelor of Information Technology, Universiti Utara Malaysia.