A Reusability Assessment of UCP-Based Effort Estimation Framework using Object-Oriented Approach

Zhamri Che Ani[1], Shuib Basri[2] and Aliza Sarlan[2]
[1]*School of Computing, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia.*
[2]*Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, 31750 Tronoh, Perak, Malaysia.*
zhamri@uum.edu.my

***Abstract*—Software effort estimation has become one of the most important concerns of software industries and Use Case Points (UCP) is seen as one of the most popular estimation models for object-oriented software development. Since year 2005, more than 10 UCP-based effort estimation techniques have been proposed either to give more options or to enhance the capability of UCP. However, there is no guidance for software practitioners to develop a quality UCP-based effort estimation applications. Therefore, we have proposed a new design framework for UCP-based technique to promote reusability in developing software applications. This paper will experiment and provide evidence showing that the framework achieved a good quality design using Quality Model for Object-oriented Design (QMOOD). The results showed that the framework has met five quality attributes and good to be reused as a guideline at the early stages of software development.**

***Index Terms*—Estimation; Reusability; Software Effort Use Case Points; QMOOD.**

I. INTRODUCTION

Software effort estimation is a process to gain a general understanding of the effort required to develop a software system or software product. There are many models that have been proposed as basis of estimating effort, schedule and cost of a software project [1, 2]. 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.

UCP has gained popularity among researchers and software practitioners because of the simplicity of use in estimating software effort. However, due to the evolution of object-oriented paradigm and the rapid changes in software technology, effort estimation is seen to be more flexible to adapt new environments. Since then, many techniques have been proposed to increase the capability of the basic UCP. Based on previous studies, there are 14 UCP-based estimation techniques have been proposed and most probably there are many more techniques to come in future. The techniques are Use Case Points (UCP) [3], Adapted Use Case Points (AUCP) [4], Industrial use of Use Case Points (IUCP) [5], UCPm [6], Use Case Size Points (USP) [7], Fuzzy Use Case Size Points (FUCP) [7], Transactions [8], Paths [8], Extended Use Case Points (EUCP) [9], Extended Use Case Points (e-UCP) [10], Simplified Use Case Points (SUCP) [11], Interactive Use Case Points (iUCP) [12], Revised Use Case Point (Re-UCP) [13] and Advancement of UCP (AUCP) [14].

Based on the evolution of UCP as mentioned above, it shows that UCP-based techniques are still relevant in today's software effort estimation and there is a need of systematic tool supports to ensure the credibility of the models in producing accurate results. Currently, most of the techniques were tested using MS Excel. The main problem of using MS Excel is it does not support the reusability of the object model. Therefore, it is impossible to extend the development as a proper application in line with the growth of those new techniques.

So far, there is no guidance for software practitioners to develop a quality UCP-based effort estimation applications. Therefore, we have proposed a new UCP-based framework to promote reusability in developing UCP-based software applications. Reusability needs to be considered before developing good software application because a reusable software is more stable in terms of reduced change density when compared to a non-reusable application [15]. Without reusability, software applications are very hard to maintain or extent [16, 17, 18, 19].

To ensure the proposed framework is good enough in terms of reusability, this paper will experiment and provide evidence showing that the framework has achieved a good quality design using Quality Model for Object-oriented Design (QMOOD). QMOOD was selected because it is the most complete, comprehensive, and supported suite, and has been validated against numerous real-world projects [20].

The remainder of the paper is structured as follows. Section II provides some basic concept of UCP-based framework. Section III describes the research methodology. Result and discussion are discussed in Section IV. Section V includes conclusion and suggestion for future work.

II. THE UCP-BASED FRAMEWORK

A new UCP-based framework for designing software effort estimation has been developed using UML notation [21]. The framework captures the important aspects of the UCP-based techniques to visualize and experiment the possible designs later. Overall, there are 19 classes and 12 of them are the key classes. In order to easily maintain the framework, generally, the framework is divided into three main areas: project size,

project complexity and risk factors. Project size consists of six classes. Five of the classes are compulsory while another one is the extendible class. Project complexity includes four compulsory classes as well as four extendible classes. Meanwhile, risk factor only has one compulsory and two extendible classes. The details of the proposed framework are illustrated in Figure 1.

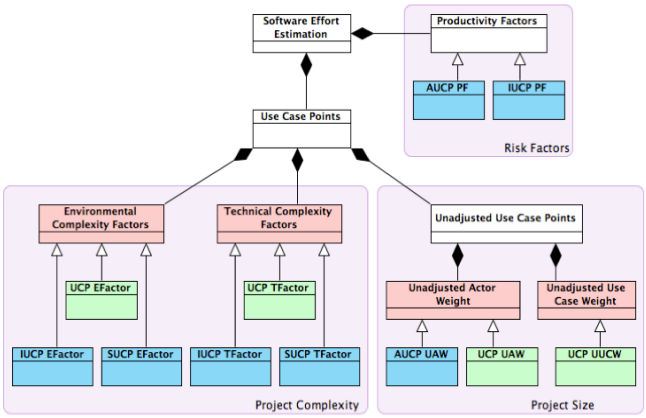

Figure 1: A framework for designing UCP-based effort estimation

As we can see in Figure 1, the main components of the framework are classes and the relationships such as association and generalization. The classes describe the concept from the domain knowledge where all software engineers may understand and agree on them. Classes can be described at various levels. In the early stages of design, the framework captures more logical aspects of the problem. In the later stages, the framework also captures design decisions and implementation details. In this study, classes are drawn as rectangles.

Relationships among classes are drawn as paths connecting class rectangles. Generalization shows the relationship between a more general description and a more specific variety of the general thing which is used for inheritance. In this case, eleven classes are inherited from their parent classes. This means that this framework can be reused by new UCP-based techniques of software effort estimation. For instance, AUCP UAW class is extended from Unadjusted Actor Weight class. Associations carry information about relationship among objects in a domain knowledge. All main classes which are captured from nine steps of UCP are associated with association relationship. This means that without these classes the effort estimation cannot be done.

III. Research Methodology

A number of metrics tools exist such as C and C++ Code Counter (CCCC), Chidamber & Kemerer Java Metrics (CKJM), Dependency Finder, Sonar, SourceMonitor, JHawk, IBM Rational Logiscope and McCabe QA [22]. However, none of the tools capable of analyzing the quality of the Java source codes based on QMOOD. Therefore, in order to ensure the proposed UCP-based framework has a good quality and can be easily reused at the early stages of product development, we have developed a new supporting tool to analyze the product metrics. The tool was developed using Java programming language based on the theoretical formula of Quality Model for Object-oriented Design (QMOOD) [23]. Figure 2 illustrates the process of analyzing the quality metrics.

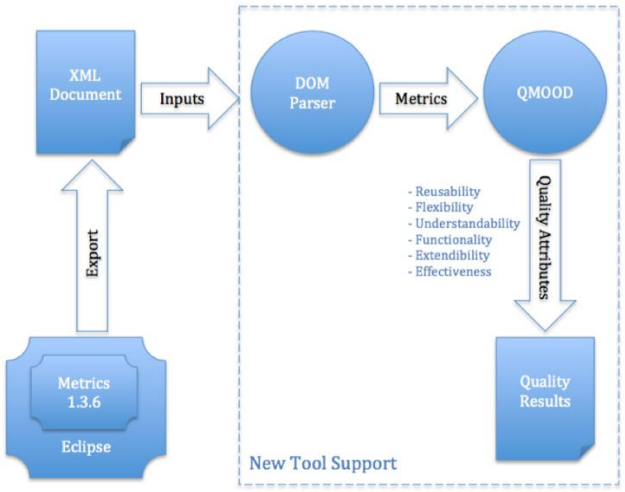

Figure 2: Process for assessing quality of UCP-based framework

In this study, basically we have divided the process into two phases. Phase 1 is about how to obtain the metrics whereas phase 2 is more on how to analyze the metrics and produce the quality results.

In phase 1, Metrics-1.3.6 [24], an Eclipse IDE's plug-in was used to measure the framework. Since it is an eclipse plug-in, the UCP-based framework which has 1030 line of source codes with 5 packages and 23 classes, was required to be imported into eclipse IDE. In our case, eclipse IDE version 4.2 (Juno) was used to compile the source codes and the codes must be compiled successfully before can be analyzed by Metrics-1.3.6. Then the obtained metrics was exported to XML file. XML file was used as a medium because it is independence and easier to be transferred into other platforms.

In phase 2, Java programming language was used to develop an automation tool support for analyzing the obtained metrics and producing the quality results. The tool will extract metrics from XML file using DOM parser and then analyze using QMOOD computation formula. QMOOD is a comprehensive quality model that establishes a clearly defined and empirically validated model to assess six quality attributes namely reusability, flexibility, understandability, functionality, extendibility and effectiveness, based on the framework of quality models defined in [25, 26]. It identifies 11 design properties for the object-oriented paradigm. The detail definition of design properties is described in Table 1.

QMOOD consists of six equations that establish relationship between six quality attributes and 11 design properties. The mathematical formulas are explained in Table 2.

QMOOD has been referred by many researchers [27, 28, 29, 30, 31]. However, due to some limitations of Metrics-1.3.6, the technique of metrics generation was adopted from Chawla and Chhabra [32]. The customization can be made to meet particular quality requirements [33]. The step of replacing metrics is described in Table 3.

Table 1
Design Properties Definitions [23]

Design Property	Definition
Design Size	A measure of the number of classes used in the design.

Design Property	Definition
Hierarchies	Hierarchies are used to present different generalization-specialization concepts in a design. It is a count of the number of non-inherited classes that have children in a design.
Abstraction	A measure of the generalization-specialization aspect of the design. Classes in a design which have one or more descendants exhibits this property of abstraction.
Encapsulation	Defined as the enclosing of data and behavior within a single construct. In object-oriented designs, the property specifically refers to designing classes that prevent access to attribute declarations by designing them to be private, thus protecting the internal representation of the objects.
Coupling	Defines the interdependency of an object on other objects in a design. It is a measure of the number of other objects that would have to be accessed by an object for that object to function correctly.
Cohesion	Assesses the relatedness of methods and attributes in a class. Strong overlap in the method parameters and attribute types is an indication of strong cohesion.
Composition	Measures the 'part-of', 'has', 'consists-of' or 'part-whole' relationships, which are aggregation relationships in an object-oriented design.
Inheritance	A measure of the 'is-a' relationship between classes. This relationship is related to the level of nesting of classes in an inheritance hierarchy.
Polymorphism	The ability to substitute objects whose interfaces match for one another at run-time. It is a measure of services that are dynamically determined at run-time in an object.
Messaging	A count of the number of public methods that are available as services to other classes. This is a measure of the services that a class provides.
Complexity	A measure of the degree of difficulty in understanding and comprehending the internal and external structure of classes and their relationships.

Table 2
Computation Formulas for Quality Attributes [23]

Quality Attributes	Index Computation Equation
Reusability	-0.25 * Coupling + 0.25 * Cohesion + 0.5 * Messaging + 0.5 * Design Size
Flexibility	0.25 * Encapsulation - 0.25 * Coupling + 0.5 * Composition + 0.5 * Polymorphism
Understandability	-0.33 * Abstraction + 0.33 * Encapsulation - 0.33 * Coupling + 0.33 * Cohesion - 0.33 * Polymorphism - 0.33 * Complexity - 0.33 * Design Size
Functionality	0.12 * Cohesion + 0.22 * Polymorphism + 0.22 * Messaging + 0.22 * Design Size + 0.22 * Hierarchies
Extendibility	0.5 * Abstraction - 0.5 * Coupling + 0.5 * Inheritance + 0.5 * Polymorphism
Effectiveness	0.2 * Abstraction + 0.2 * Encapsulation + 0.2 * Composition + 0.2 * Inheritance + 0.2 * Polymorphism

IV. Result and Discussion

After successfully compiling the source codes, the new supporting tool as mentioned before was used to analyze the statistics. Table 4 presents the descriptive statistics of metrics computed by the tool. In this study, the Z-value was calculated by applying the following formula: Z-value = (Max-Avg)/(Std-Dev).

Based on the statistics obtained in Table 4 and after applying the formula mentioned in Table 2, Figure 3 shows the quality attributes of UCP-based framework.

Table 3
QMOOD Design Metrics & Substitute Metrics [32]

Design Property (QMOOD)	Design Metrics (QMOOD)	Equivalent Metric Computed by Metrics-1.3.6	Construct
Coupling	Direct Class Coupling (DCC)	Efferent Coupling (EC)	Package
Cohesion	Cohesion Among Methods of Classes (CAM)	* Lack of Cohesion of Methods (LCOM)	Class
Messaging	Class Interface Size (CIS)	Number of Methods (NOM)	Class
Design Size	Design Size in Classes (DSC)	Number of Classes (NOC)	Package
Encapsulation	Data Access Metric (DAM)	1	
Composition	Measure of Aggregation (MOA)	Number of Attributes (NOF)	Class
Polymorphism	Number of Polymorphic Methods (NOP)	Number of Overridden Methods (NORM)	Class
Abstraction	Average Number of Ancestors (ANA)	Abstractness (RMA)	Package
Complexity	Number of Methods (NOM)	Weighted Methods per Class (WMC)	Class
Hierarchies	Number of Hierarchies (NOH)	Depth of Inheritance Tree (DIT)	Class
Inheritance	Measure of Functional Abstraction (MFA)	* [∑ NORM / NOM * 100]	Class

Table 4
Descriptive Statistics of UCP-Based Framework

Metrics	Max	Avg	Std-Dev	Sum	Z-value (normalized)
CE	4	0.8	1.6	-	2.0
*Cohesion (1/LCOM)	0.667	0.029	0.136	-	4.69
NOM	18	4.826	4.575	111	2.88
NOC	8	4.6	2.47	23	1.41
*Encapsulation =1	-	-	-	-	#
NOF	13	2.522	3.999	58	2.62
NORM	1	0.087	0.282	2	3.24
RMA	0.333	0.117	0.145	-	1.49
WMC	18	5.31	4.6	118	2.8
DIT	2	1.478	0.5	-	1.04
*Inheritance =6.61	-	-	-	-	#

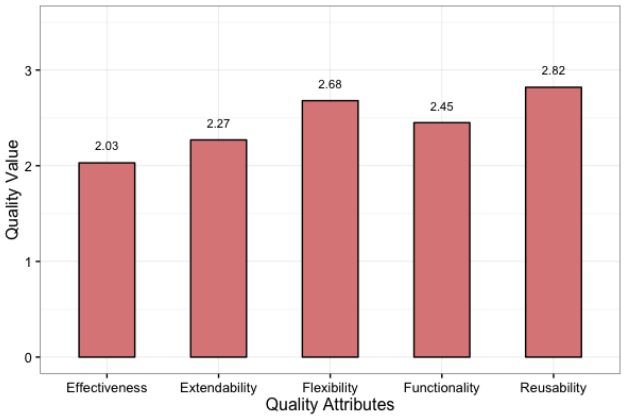

Figure 3: Quality attributes of UCP-based framework

The result shows that the UCP-based framework has achieved a good quality design. Out of six quality attributes

defined in Table 2, only one attribute has negative value. Understandability which is not plotted in the chart, is the only quality attribute that has a negative value, -1.73. However, due to the small negative value which is less than -2, this means less efforts are needed to maintain the UCP-based software projects [23]. Meaning that the framework is not too hard to learn and understand.

On the positive side, the most interesting finding is that the new UCP-based framework is easily to be reused. This finding gives a major impact to the overall quality of the software products. Overall, the quality value of UCP-based framework is 10.52. Since this is the first attempt to assess the framework, the achieved values still need to be verified with several replication studies in order to set the high-quality standard of reusability for this framework.

V. Conclusion and Future Work

This study presents an evaluation of UCP-based framework to assess the reusability of the design. The framework was analyzed using QMOOD to find out what are the levels of reusability quality attribute to be achieved. The results show that the framework has met five quality attributes and reusability is the highest. This means that the framework is good enough to be reused as a guideline especially at the early stages of the UCP-based software development. In other words, those who are planning to develop new UCP-based software products, these results can be used as a benchmark to improve the quality of their designs. However, we believe that several replication studies need to be done to verify these results in order to set the high-quality standard of reusability for this framework. For future research, we also plan to further investigate on this topic by using other object-oriented design metrics and make comprehensive comparison against these results.

References

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

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

[3] G. Karner, "Resource estimation for objectory projects," *Objective Systems SF AB*, vol. 17, 1993.

[4] P.Mohagheghi, B.Anda, and R.Conradi, "Effort estimation of use cases for incremental large-scale software development," in *Proceedings of the 27th* IEEE *International Conference on Software Engineering*, 2005, pp. 303–311.

[5] E. Carroll, "Estimating software based on use case points," in *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, San Diego, CA, USA, 2005, pp. 257–265.

[6] S.Diev, "Use cases modeling and software estimation: Applying use case points," *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 6, pp. 1–4, 2006.

[7] M. R. Braz, and S. R. Vergilio, "Software effort estimation based on use cases," in *Proceedings of the 30th IEEE Annual International Computer Software and Applications Conference (COMPSAC'06)*, vol. 1. 2006, pp. 221–228.

[8] G. Robiolo, C. Badano, and R. Orosco, "Transactions and paths: two use case based metrics which improve the early effort estimation," in *Proceedings of the 3rd IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2009, pp. 422–425.

[9] W. Fan, Y. Xiaohu, Z. Xiaochun, and C. Lu, "Extended use case points method for software cost estimation," in *International Proceedings of the Conference on Computational Intelligence and Software Engineering*. IEEE, 2009, pp. 1–5.

[10] K. Periyasamy, and A. Ghode, "Cost estimation using extended use case point (e-ucp) model," in *Proceedings of the International Conference on Computational Intelligence and Software Engineering*. IEEE, 2009, pp. 1–5.

[11] M. Ochodek, J. Nawrocki, and K. Kwarciak, "Simplifying effort estimation based on use case points," *Information and Software Technology*, vol. 53, no. 3, pp. 200–213, 2011.

[12] N. Nunes, L. Constantine, and R. Kazman, "iucp: Estimating interactive-software project size with enhanced use-case points," *IEEE Software*, vol. 28, no. 4, pp. 64–73, 2011.

[13] M. M. Kirmani, and A. Wahid, "Revised use case point (re-ucp) model for software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 3, pp. 65–71, 2015.

[14] A. Srivastava, S. Singh, and S. Q. Abbas, "Advancement of ucp with end user development factor: Aucp," *International Journal of Software Engineering & Applications (IJSEA)*, vol. 6, no. 2, pp. 1–10, 2015.

[15] A. Gupta, J. Li, R. Conradi, H. Rønneberg, and E. Landre, "Change profiles of a reused class framework vs. two of its applications," *Information and Software Technology*, vol. 52, no. 1, pp. 110–125, 2010.

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

[17] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, USA, 1977, vol. 2.

[18] P. Kuchana, *Software Architecture Design Patterns in Java*. CRC Press, 2004.

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

[20] M. El-Wakil, A. El-Bastawisi, M. Boshra, and A. Fahmy, "Object-oriented design quality models a survey and comparison," in *Proceedings of the 2nd International Conference on Informatics and Systems (INFOS 2004)*, 2004.

[21] H. Gomaa, *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. Cambridge University Press, 2011.

[22] M. K. Chawla and I. Chhabra, "A quantitative framework for integrated software quality measurement in multi-versions systems," in *Proceedings of the International Conference on Internet of Things and Applications (IOTA)*. IEEE, 2016, pp. 310–315.

[23] J. Bansiya and C. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 4–17, 2002.

[24] "Metrics 1.3.6," January 2016. Available at http://metrics.sourceforge.net/

[25] R. G. Dromey, "A model for software product quality," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 146–162, 1995.

[26] R. Dromey, "Cornering the chimera," *IEEE Software*, vol. 13, no. 1, pp. 33–43, 1996.

[27] D. Arora, P. Khanna, A. Tripathi, S. Sharma, and S. Shukla, "Software quality estimation through object oriented design metrics," *International Journal Computer Science and Network Security*, vol. 11, no. 4, pp. 100–104, 2011.

[28] V. Yadav, and R. Singh, "Predicting design quality of object-oriented software using uml diagrams," in *Proceedings of the 3rd IEEE International Advance Computing Conference (IACC)*, 2013, pp. 1462–1467.

[29] G. Samarthyam, G. Suryanarayana, T. Sharma, and S. Gupta, "Midas: a design quality assessment method for industrial software," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 911–920.

[30] M. K. Chawla, and I. Chhabra, "Implementation of an object-oriented model to analyze relative progression of source code versions with respect to software quality," *International Journal of Computer Applications*, vol. 107, no. 10, 2014.

[31] P. K. Goyal, and G. Joshi, "Qmood metric sets to assess quality of Java program," in *Proceedings of the IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, 2014, pp. 520–533.

[32] M. K. Chawla, and I. Chhabra, "Capturing Object-oriented software metrics to attain quality attributes – A case study," *International Journal of Scientific and Engineering Research*, vol. 4, pp. 359–363, 2013.

[33] S. Barney, K. Petersen, and M. Svahnberg, A. Aurum and H. Barney, "Software quality trade-offs: A systematic map," *Information and Software Technology*, vol. 54, no. 7, pp. 651–662, 2012.