On a Use Case Points Measurement Tool for Effective Project Management

Shinji Kusumoto*, Michio Tsuda*+, Katsuro Inoue*
*Graduate School of Information Science and Technology, Osaka University
+Hitachi Systems & Services, Ltd.
{kusumoto, m-tsuda, inoue}@ist.osaka-u.ac.jp

Abstract

Use case point (UCP) method has been proposed to estimate software development effort in early phase of software project and used in a lot of software organizations. This paper briefly describes an automatic use case measurement tool, called U-EST.

1. Introduction

One of the important topics of software development is estimation. To attain comprehensive estimation for both software companies and customers is essential for success of the project. So far, several effort models have been proposed and most of them include software "size" as an important parameter.

To estimate the effort in earlier phase, use case point method has been proposed[2]. Use case point (UCP) is measured from a use case model that defines the functional scope of the software system to be developed. Since it is measured in the earlier phase, it is necessary to evaluate the correctness and appropriateness after the software is delivered. Thus, UCP of the released software seems to be good information for software companies and customers. However, it is time-consuming and difficult to measure UCP from the source code. So, developing a support tool to measure UCP is worthwhile.

This paper briefly describes the UCP measurement tool and shows the case study of applying it to several use case models.

2. Use Case Point Method

Use case point (UCP) is calculated mainly from system level use case diagram and flow of events in use case model. System-level use case diagram includes one or more use case diagrams showing all the use cases and actors in the system. Flow of events includes a section for the basic path and each alternative path in each use case. Intuitively, UCP is measured by counting the number of actors and transactions included in the flow of events with some weight. A transaction is an event that occurs between an actor and the target system, the event being performed entirely or not at all.

The main activities of counting UCP include the following two steps:

Step1 (Counting actors weight): The actors in the use case model are categorized as simple, average or complex as shown in Table 1. Then, the number of each actor type is calculated and then each number is multiplied by a weighting factor. Finally, actors weight is calculated by adding these values together.

Step2 (Counting use cases weight): Each use case is categorized as simple, average or complex as shown in Table2. The basis of this decision is the number of transaction in a use case, including alternative paths. Then, the number of use case type that the target software includes calculated and then each number is multiplied by a weighting factor shown in Table 2. Finally, use case weight is calculated by adding these values together.

Table 1. Actor Weighting Factors

Type	Description	Factor
Simple	Program interface	1
Average	Interactive or protocol-driven interf	2
Complex	GUI	3

Table2: Use-case Weighting Factors

Type	Description	Factor
Simple	3 or fewer transactions	5
Average	4 to 7 transactions	10
Complex	More than 7 transactions	15

3. Use Case Point Measurement Tool

3.1 Overview

In order to develop an automatic use case point measurement tool, it is necessary to develop a way of determining the weight for each actor and use case. To attain it, on the assumptions that the method is applicable in a specific software organization, we propose several rules to classify the weight for actor and use case.

3.2 Rules for weighting actors

In order to judge the type of actors, we used the following three steps: (1) We judge whether the actor is a person or an external system based on the name of it using the list of keywords which can be included in the name of software system. (2) Based on the keyword included in the flow of events related to the actor, we determine the type of the actor.

3.3 Rules for weighting use cases

The type of use case is determined by the number of transaction. So, we focus on the flow of events in

the use case model. Intuitively speaking, the simplest way to count the transaction is to count the number of event. But, since there are no standard to write the flow of events, the developer can write the description freely using natural language. It is quite possible that several transactions are described in one event. On the other hand, several guidelines to write events in use case model have been proposed [1]. There are ten guidelines to write a successful scenario (flow of events). Among them, we focus on the following two guidelines. (G1) Use simple grammar: The sentence structure should be absurdly simple. That is, it is easily understand what is the subject, verb, direct object and prepositional phrase. (G2) Include a reasonable set of actions: Jacobson has described a step in a use case as representing a transaction. He suggests the following four pieces of a compound interactions should be described. (1)The primary actor sends request and data to the system, (2)The system validates the request and the data, (3)The system alters its internal state and (4) The system responds to the actor with the result.

So, based on the above guidelines, we propose the way to analyze the events using the morphological analysis and syntactic analysis. Through these analyses, we can get the information of morpheme from the statement and dependency relation between words in the statement. We conduct the morphological analysis for all events (statements) and get the information of the subject word and predicate word from each event (statement). Then, we regard each set of the subject and predicate word as a candidate of a transaction. Then, among the candidates, we identify the one that related to actor's operation and system response as a transaction. For each use case, we conduct the above processing and then get the number of transactions. Based on the number of transaction, we judge the type of each use case.

3.4 Implementation

Based on the proposed method, we have implemented a prototype tool called U-EST(Use case based Estimation Supporting Tool). The input is a XMI file. The U-EST is implemented in Java and Xerces2 Java Parser is used to analyze the model le. Since the U-EST is mainly used in Japanese engineers, it has to deal with the Japanese description. In order to conduct morphological analysis and syntactic analysis for event written in Japanese in the use case, we adopt a tool called CaboCha[3]. CaboCha is the most famous and precise syntactic analyzer for Japanese.

4. Case study

In order to evaluate the usefulness of the U-EST, we applied it to actual use case models developed in a software company. We collected use case models from five software projects where middle-size Web application programs were developed. As they are for Japanese use, the name of actors, use case and the descriptions of flow of events are written in Japanese. All use case models were developed on a UML-design tool called Describe. In the evaluation, we focused on the results of the automatic type classification of actors and use cases. So, we compared the measurement results calculated by our tool and ones calculated by a specialist of use case point counting.

With respect to the results of actor classification, the values measured by the tool are similar to the ones by the specialist. However, typed for actors that are external systems are quite different. On the other hand, the classification results of use case are shown in Table 3. The values measured by the tool are similar to the ones by the specialist.

Table3. Classification of use cases

Project	Specialist			U-EST		
	Simple	Average	Complex	Simple	Average	Complex
A	13	2	0	13	2	0
B	10	4	0	10	4	0
C	14	6	0	12	8	0
D	27	1	0	27	1	0
E	2	8	3	2	8	3

5. Conclusions

This paper proposed an automatic use case point tool, the U-EST. The U-EST calculates use case point from use case models written in XMI files. We have also applied the U-EST to five use case models developed in the actual software projects. As the results, the UCP calculated by the U-EST are considerably adequate.

Acknowledgment

This work is being conducted as a part of Stage Project, the Development of Next Generation IT Infrastructure, supported by Ministry of Education, Culture, Sports, Science and Technology and Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (C) (17500022).

References

[1] A. Cockburn: Writing Effective Use Cases, Addison-Wesley (2000).

[2] G. Schneider and J. P. Winters: “Applying Use Cases, Second Edition", Addison Wesley (2001).

[3] CaboCha : Yet Another Japanese Dependency Structure Analyzer, http://cl.aistnara.ac.jp/ taku-ku/software/cabocha/