Information and Software Technology 53 (2011) 200–213

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Simplifying effort estimation based on Use Case Points ☆

M. Ochodek *, J. Nawrocki, K. Kwarciak

Poznan University of Technology, Institute of Computing Science, ul. Piotrowo 2, 60-965 Poznań, Poland

ARTICLE INFO

Article history:
Received 7 April 2010
Received in revised form 18 October 2010
Accepted 20 October 2010
Available online 26 October 2010

Keywords:
Use Case Points
Software cost estimation
Use cases
Use-case transactions
TTPoints

ABSTRACT

Context: The Use Case Points (UCP) method can be used to estimate software development effort based on a use-case model and two sets of adjustment factors relating to the environmental and technical complexity of a project. The question arises whether all of these components are important from the effort estimation point of view.
Objective: This paper investigates the construction of UCP in order to find possible ways of simplifying it.
Method: The cross-validation procedure was used to compare the accuracy of the different variants of UCP (with and without the investigated simplifications). The analysis was based on data derived from a set of 14 projects for which effort ranged from 277 to 3593 man-hours. In addition, the factor analysis was performed to investigate the possibility of reducing the number of adjustment factors.
Results: The two variants of UCP – with and without unadjusted actor weights (UAW) provided similar prediction accuracy. In addition, a minor influence of the adjustment factors on the accuracy of UCP was observed. The results of the factor analysis indicated that the number of adjustment factors could be reduced from 21 to 6 (2 environmental factors and 4 technical complexity factors). Another observation was made that the variants of UCP calculated based on steps were slightly more accurate than the variants calculated based on transactions. Finally, a recently proposed use-case-based size metric TTPoints provided better accuracy than any of the investigated variants of UCP.
Conclusion: The observation in this study was that the UCP method could be simplified by rejecting UAW; calculating UCP based on steps instead of transactions; or just counting the total number of steps in use cases. Moreover, two recently proposed use-case-based size metrics Transactions and TTPoints could be used as an alternative to UCP to estimate effort at the early stages of software development.
© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Software effort estimation is one of the key aspects of successful project management. If an unrealistic assumption about the development cost is made, the project is in danger. Both underestimated and overestimated effort is harmful. Underestimation leads to a situation where a project's commitments cannot be fulfilled because of a shortage of time and/or funds. Overestimation can result in the rejection of a project proposal, which otherwise would be accepted and would create new opportunities for the organization.

Unfortunately, effort estimation at the early stages of software development is a challenge. Firstly, very little is known about the project. Secondly, there is a threat that the project will not be accepted for further development, so limited resources can be spent on effort estimation. Thus, there is a trade-off between the level of estimation error and the resources assigned to the estimation activities (typically, the smaller the estimation error the bigger the estimation cost associated with acquiring knowledge about the project at hand).

In this context two kinds of research could be useful:

- simplifying effort estimation methods without compromising their accuracy;
- making effort estimation more accurate without increasing the time and money spent on effort estimation.

Typical inputs available at early stages of software development are functional requirements, which describe what a system is expected to do. These kinds of requirements can be used to measure the size of a system, and estimate its development effort.

The idea of functional size measurement (FSM) was introduced by Allan Albrecht [1], who proposed a method called Function Point Analysis (FPA). Since the introduction of the method, its construction has been broadly discussed and frequently questioned (see, e.g., [2–6]). Nevertheless, it still remains one of the most popular FSM methods, and since 1986, it has been

☆ This research project operated within the Foundation for Polish Science Ventures Programme co-financed by the EU European Regional Development Fund.
* Corresponding author.
E-mail addresses: mochodek@cs.put.poznan.pl (M. Ochodek), jnawrocki@cs.put.poznan.pl (J. Nawrocki), kkwarciak@cs.put.poznan.pl (K. Kwarciak).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.10.005

maintained by a non-profit organization called the International Function Point User Group (IFPUG).

Albrecht's FPA method stimulated evolvement of other FSM methods, e.g., Mark II Function Points proposed by Symons [7], COSMIC [8], or Use Case Points[1] (UCP) introduced by Karner [9].

The latter method is especially valuable in the context of early size measurement and effort estimation, because it employs *use cases* as an input. Use cases, proposed by Jacobson [10,11], are a popular form of representing functional requirements (according to the survey conducted by Neill and Laplante in 2003 [12], 50% of projects have their functional requirements presented as scenarios or use cases). They are also available in the early stages of software development.

The mechanism of the Use Case Points method was inspired by both Albrecht's FPA [1] and MK II Function Points [7], and since its introduction many variants of the method have been proposed, e.g., Use Case Size Points (USP) and Fuzzy Use Case Size Points (FUSP) [13]; UCPm [14]; and the adapted version for incremental large-scale projects [15].

As use cases are gaining popularity also the UCP method (and its derivatives) are getting more popular. However, some people pointed out problems concerning the construction of the method (differences in use case models [16,17], assessment of the use-case-model complexity [13,17], assessment of adjustment factors [14,17,18], and involvement of calculations that are based on algebraically inadmissible scale-type transformations [18,19]). Therefore, the question arises whether the method is well designed. Maybe it could be simplified without loosing much of its accuracy.

This question is even more important in the context of recently proposed use-case-based size metrics, i.e., Transactions [20], and TTPoints [21]. These metrics seem simpler than UCP.

Therefore, the goal of this study is to analyze the construction of the UCP method, investigate the influence of its components on the accuracy of the method, and propose possible simplifications.

The paper is organized as follows. The next section provides a brief introduction to use cases and the UCP method. Section 3 presents a set of projects used in this study as a historical database. Section 4 describes the research method that was used to evaluate the estimation accuracy of the different variants of UCP and other use-case-based size metrics. In the following sections components of UCP are analyzed: actors complexity – in Section 5; adjustment factors – in Section 6; use-case complexity – in Sections 7 and 8. The role of transactions in use-case-based effort estimation is investigated in Section 9. The threats to validity of this study are discussed in Section 10. The summary and the list of the most important findings can be found in Section 11.

2. Use cases and the Use Case Points method

2.1. Use cases

The main aim of use cases is to present interaction between end-user (called actor) and the described system in terms of user-valued transactions – using natural language. Such use cases are called system-level use cases. (There are also business-level use cases: they describe interaction between people who cooperate to obtain a business goal.)

According to the guidelines for writing use cases [22,23] the most important parts of a use case are as follows: *name*/*title* which describes the goal, *actors* participating in the use case (people or cooperating systems), *main scenario* which is the most common sequence of steps leading to the goal, and *extensions* to the main scenario describing alternative steps associated with the occurrence of some events. An example of a use case is presented in Fig. 1.

UC1: Submit a paper
Level: **User**
Main actor: **Author**

Main Scenario:
1. **Author** chooses the option to submit a *paper*.
2. **System** presents the submission form.
3. **Author** provides necessary information about the *paper*.
4. **System** informs Author that the *paper* was submitted.

Alternatives, Extensions, Exceptions:
3.A. Not all required data was provided.
  3.A.1. **System** displays error message.
  3.A.2. Go to step 2.

Fig. 1. An example of a use case presented as a structured text.

2.2. The Use Case Points method

In order to obtain UCP for the system one has to start with the assessment of the complexity of actors and use cases; and then adjust it with two kinds of factors characterizing the development environment and the technical complexity of the system under development.

2.2.1. Actors complexity

The first step of the UCP method is to assign each actor to one of three complexity classes:

- *simple*: an actor representing a system which communicates with other actors using API;
- *average*: a system actor which communicates through a protocol (e.g. HTTP, FTP), or a person who interacts with a system through a terminal console;
- *complex*: a person who uses graphical user interface (GUI) in order to communicate with a system.

Each actor-complexity class, c, is characterized by two numbers:

- $aWeight(c)$ = 1 for *simple*, 2 for *average*, and 3 for *complex*;
- $aCardinality(c)$ is the number of actors assigned to class c (depends on a described system).

For a given system, the *unadjusted actor weights (UAW)* are computed as a sum of products – the weight of complexity class and the number of actors assigned to that class, see the following equation:

$$UAW = \sum_{c \in C} aWeight(c) \times aCardinality(c), \quad C = \{simple, average, complex\} \tag{1}$$

2.2.2. Use-cases complexity

The second step of the UCP method is the assessment of use-case complexity. This complexity depends on the number of transactions identified in each use case. (Transaction is a set of activities

[1] It is not clear whether UCP is a size measure or a software estimation method. Some sub-components of UCP (presented in Section 2.2) such as UUCW, UAW, and UUCP could be clearly treated as functional size measures. However, when UUCP is multiplied by TCF and EF, it is no longer clear whether it represents size of the system or its predicted development effort. (The environmental factors represent commonly used cost drivers.)

Download English Version:

https://daneshyari.com/en/article/551832

Download Persian Version:

https://daneshyari.com/article/551832

Daneshyari.com