Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 1524–1531

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Proposal of an Evaluation Method of Individual Contributions using the Function Point in the Implementation Phase in Project-Based Learning of Software Development

Koyo Tanabata[a], Atsuo Hazeyama[a]*, Yuki Yamada[b], Kiichi Furukawa[b]

[a]*Tokyo Gakugei University, 4-1-1 Nukuikita-machi, Koganei-shi, 184-8501, Japan*
[b]*Graduate School of Education, Tokyo Gakugei University, 4-1-1 Nukuikita-machi, Koganei-shi, 184-8501, Japan*

Abstract

In recent years, software development is generally done in the form of a team project. Correspondingly, educational institutions have adopted project-based software development exercises (called project-based learning (PBL) of software development in this paper). The PBL of software development can perform various processes in software development, which will be a great experience for students to play an active role in their profession in the future. However, an individual evaluation of students in such a team project is difficult and involves several problems. Therefore, to evaluate students fairly, this research proposes an evaluation method to measure the contributions of individuals in the implementation phase in the PBL of software development. Particularly, it is an evaluation method that applies the function point method, which is known as the size estimation method.

Keywords: Project-based software engineering education; measurement of contributions; the function point method.

* Corresponding author. Tel.:+81-42-329-7465; fax: +81-42-329-7465.
E-mail address: hazeyama@u-gakugei.ac.jp

10.1016/j.procs.2021.08.156

1. Introduction

A team project is widely recognized to play a major role in an undergraduate or graduate software engineering course. Accordingly, project-based learning (PBL) is adopted in recent years. However, it is difficult to evaluate each student in a team project, and there exist several issues in the evaluation [1]. Particularly, it is difficult to represent the quantity of work done by students, to evaluate them in a fair manner. Generally, certain phases such as the requirements analysis, design, implementation, and testing are conducted in software development. Implementation is performed by referring to requirements specification and/or design documents. Contributions by students within a team differ significantly during the implementation phase compared to other phases. In the requirement analysis and certain design activities such as the data design, activities as a team are primary. In these phases, it is difficult to quantitatively distinguish the contributions of each member based on their outputs. However, tasks are assigned to each member during the detailed design (for example, the creation of sequence diagrams), implementation, and unit testing phases. Therefore, it is rather easier to assign contributions to them. We believe that a quantitative representation of the contributions allows the individual evaluation of students in the implementation phase. Although various studies have been conducted for quantifying the contributions of individuals in a software project, few studies have been conducted for quantifying the contributions of individuals based on the functions they have implemented.

This study focuses on the implementation of functions by each student during the PBL of software development. We propose a method that quantifies the implementation of functions by each student using the function point (FP) method [2]. The FP method was originally developed for estimating the size of software by quantifying functions that compose a software system. The original FP method is applied in the upstream phase for estimating the size of software. However, we apply the FP method to quantify the contributions of each student after completing the implementation phase because the implementation may vary from the requirements specification, or the intended functions may not be implemented owing to various reasons, such as a course schedule or assignment change. The FP method can consider the complexity of functions; therefore, we believe that it enables to quantify the contributions of individual students who implement those functions.

The structure of this paper is as follows: section 2 describes related work. Section 3 presents trial usage of the FP method to our PBL. Section 4 proposes an evaluation method of individual contributions in the implementation phase using the FP method. Section 5 gives a preliminary evaluation of our proposal. Section 6 summarizes this paper.

2. Related work

This section describes related work regarding evaluation of individuals in a team project and application of the FP method. Then we show our position in this field.

2.1. Evaluation of individuals in a team project

Parizi et al. measured the contributions of individuals by weighing the data obtained from GitHub [3]. Their study used the number of commits, merges, files and lines of code per month, and work hours per day.

Gamble and Hale described that it is necessary to evaluate the performance of individuals to identify how and how much the individuals participate in projects [4]. Although their study needed measures for differentiating the evaluation of individuals and teams, they described that the contributions of individuals tend to depend on self-assessment.

Miyashita et al. proposed a method to evaluate contributions by learners in the PBL of software development based on the group engagement method (GEM) [5]. They associated each assessment item in GEM with the data obtained from GitHub. They focused on all the phases in the PBL of software development.

Buffardi described that if the instructor assigned the same grade to all the students without conducting performance evaluation, students would devote less efforts to the team goal [6].

2.2. Usage of the FP method

Huang and Zheng applied the FP method to earned value analysis (EVA) [7]. Except for [7], we do not find studies in which the FP is applied to other purposes except for size estimation.

2.3. Position of our study

We focus on the implementation phase of software development and aim to objectively measure the contributions of individuals to the implementation phase.

Lines of code (LoC) is a method to quantify the implementation of functions. The LoC has an advantage in that it is easy to measure. However, we observed that the LoC significantly differed in the way of implementation for the same requirement specification. Therefore, the LoC does not correlate with the quantity of functions. We propose to apply the FP method, not the LoC, to measure the contributions of individuals.

3. A trial of quantification of contributions using the FP method

This study proposes a method to quantify the contributions of individuals in the implementation phase of the PBL of software development based on the FP method. We discuss the applicability of the FP method to quantify the contributions of individuals using the actual data of our past conducted PBL.

3.1. The FP method

We briefly explain the FP method.

(1) The IFPUG method

The FP method is originally developed for size estimation of software [2]. It focuses on functions that are visible to users such as data input or output and quantifies size of software by the number of functions and their weight. This study uses the IFPUG method that is a major FP method [8]. The function in the FP method stands for Internal Logic File (ILF) and External Interface File (EIF) in the data functions (DF), and External Input (EI), External Output (EO) and External Query (EQ) in the transaction functions (TF). We explain them in more detailed.

• DF

– Internal Logic File (ILF): a group of logical related data or control information, identifiable by the user and maintained within the application boundary

– External Interface File (EIF): a group of logical related data or control information, identifiable by the user and referenced by the application, but maintained within other application boundary

• TF

– External Input (EI): the process of receiving data or control information beyond the boundary of the system

– External Output (EO): the process of sending data or control information out of the system for presenting to the user logically manipulated data or control information other than the extracted data.

– External Query (EQ): data or control message sent out of the system boundary to show the user a set of extracted data or control information

(2) The FP calculation method

Here describes the FP calculation method. Eq. (1) is a formula for calculating the FP. As the adjustment value is optional, we use the Unadjusted Function Points (Unadjusted FP) only.

The coefficient in Table 1 is used for calculating the Unadjusted FP. The complexity level of Low, Average and High in the table is determined by the criterion in Table 2.

The FTR and DET in Table 2 and RET are determined as follows [9]:

FTR: File Type Reference, a data function referred to or read by a transaction function

DET: Data Element Type, non-repetitive or non-recursive unique field read from file

RET: Records Element Type

For example, when we calculate the FP of EI, assuming that FTR is two and DET is ten, the complexity level is “Average” from Table 2. From Table 1, the FP is obtained as 4. Thus we calculate the FP for each function.

$$\text{FP} = \text{Unadjusted FP} \times (0.65 + \text{the adjustment value} * 0.01) \qquad \text{Equation (1)}$$

Table 1. Coefficient for function type

	Low	Average	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Table 2. Complexity level of EI

	1-4 DET	5-15 DET	16 DET+
1 FTR	Low	Low	Average
2 FTR	Low	Average	High
3 FTR+	Average	High	High

3.2. Trial

This study proposes to use the IFPUG method to quantify functions that each student implemented at the time of completing implementation.

In this subsection, we give an overview of our PBL and then describe the trial of applying the FP method in the calculation of contributions.

(1) Overview of our PBL

The development process of our PBL was based on the waterfall model. The following types of artifacts were created: requirements specification, user interface design document, class diagram, database design document, sequence diagram, source code, unit/system testing report, development plan, group progress report (every week), and project completion report. The detailed design (we assumed the artifact to be a sequence diagram) and source code were created, and unit testing was conducted for each function and by the assigned student. We assumed that the assignment of tasks in the detailed design, implementation, and unit testing was determined at the detailed design phase. Changes in the assignment in the implementation phase were allowed for certain reasons (for example, the level of difficulty and delay of schedule). A team comprised three to five students.

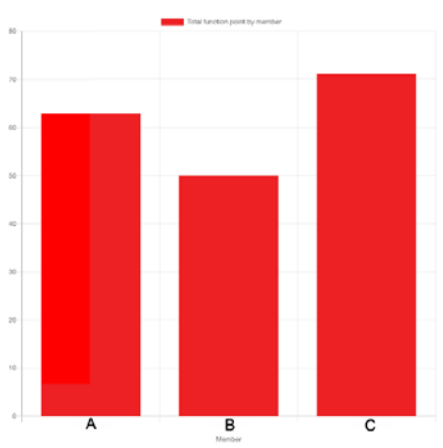
(2) Trial calculation of the FP

We calculated the FP value for a team in the 2019 academic year. We obtained the necessary data from the GitHub repository [10]. Fig. 1 depicts the result. We decided who implemented which functions based on the milestone function of GitHub. The milestone function enabled to register tasks (and subtasks) and the students who were responsible for them.

When each task was completed, the status of completion, as shown in Fig. 2, was updated. Thus, we ascertained whether a function was completed and by whom. The bar graph in Fig. 1 shows the total number of calculated FP for each student, and the table in green presents the total number of FP and number of implemented functions by members in a team. A–C is a pseudonym of students.

The difference in the number of functions implemented by B and C is two. However, the difference in the number of FP implemented by B and C is more than two. What happened? Fig. 3 shows that the function that C was responsible for had more ILF functions than others. C was responsible for more complicated functions. By introducing the FP method, the contribution evaluation method has the possibility to consider the complexity of functions.

We noted that the milestone in Fig. 2 indicated the student who modified files. When certain team members modified a file, they might be responsible for the function, or a member might support the function. In addition, in this study, we observed that although certain files existed, the function that is consisted of the files was not embedded in the system. That is, the function was unfinished.



Name	Total No. FP	No. implemented functions
A	63	6
B	50	5
C	71	7

Fig. 1 A graph of calculated FP

Fig. 2 Example milestones

Name	Function name	TF/DF	Type of function		FTR	DF DET	RET	FP
A	Register a new member	TF	EI	9	1	0	0	3
A		DF	ILF	0	0	9	1	10
A	Change a password	TF	EI	5	1	0	0	3
A		DF	ILF	0	0	5	1	10
A	Buy a ticket	TF	EO	8	1	0	0	4
A		DF	ILF	0	0	1	1	10
A	Display a ticket	TF	EO	6	3	0	0	5
A	Display a flight history	TF	EO	4	3	0	0	4
A	Reserve a seat	TF	EO	1	2	0	0	4
A		DF	ILF	0	0	1	2	10
B	Search	TF	EO	10	3	0	0	5
B	Reserve a flight	TF	EO	9	3	0	0	5
B		DF	ILF	0	0	2	1	10
B	Cancel a flight	TF	EI	5	2	0	0	3
B		DF	ILF	0	0	2	1	10
B	Cancel a flight automatically	TF	EI	1	1	0	0	3
B		DF	ILF	0	0	1	1	10
B	Display a list of flights	TF	EO	5	1	0	0	4
C	Login	TF	EI	2	1	0	0	3
C	Logout	TF	EQ	0	0	0	0	3
C	Register an airport	TF	EI	2	1	0	0	3
C		DF	ILF	0	0	2	1	10
C	Register a line	TF	EI	6	1	0	0	3
C		DF	ILF	0	0	6	1	10
C	Register a plane type	TF	EI	2	1	0	0	3
C		DF	ILF	0	0	2	1	10
C	Register a flight	TF	EI	8	1	0	0	3
C		DF	ILF	0	0	8	1	10
C	Delete a flight	TF	EI	1	2	0	0	3
C		DF	ILF	0	0	1	2	10

Fig. 3 FP per function (excerpt)

3.3. Issues found from the result of trial using the FP method

We identified two issues from the aforementioned trial: invisible mutual aid in a team does not lead to a proper evaluation of contributions, and unfinished functions that try to finish cannot be evaluated.

Regarding mutual aid: Technical ability differs among team members in software development. This leads to mutual aids. Learning from one another and interdependence in a team project will lead to better learning [1].

Regarding unfinished function: Since PBL is an educational endeavor, we think that the process should be evaluated even if the implementation has not finished. However, it is difficult to identify the quantity that have finished around unfinished functions.

4. Proposal of an evaluation method of individual contributions in the implementation phase

This section proposes an evaluation method of individual contributions in the implementation phase that solves the issues we raised in the previous section.

4. 1 Evaluation of contributions on mutual aid

As we described in Section 3.3, mutual aid within a team is a good opportunity for learning. Therefore, we positively evaluate those who have conducted mutual aid for team members. The evidence is collected from GitHub. We request the students to record mutual aid in the milestone of GitHub. The members who have performed mutual aid are appended 30% of the FP of the corresponding functions to their FP. Further, we do not decrement the FP from the students who received mutual aid.

4. 2 Evaluation of contributions on unfinished implementation

To calculate the contributions for an unfinished function, we use the milestone. We request the students to create subtasks for unfinished functions. For example, when a system is developed using the Java Server pages (JSP)/Servlet technology, we can create subtasks: jsp, servlet, control, and data access object (dao). Assuming that the FP of a function is 12 and only the dao file is finished, the contribution of the function is three (= 12/4).

5. Preliminary evaluation

We evaluated the effectiveness of our proposed method. We set the following questions to be clarified:

Q1: Are the contributions calculated using our proposed method valid?

Q2: Is our concept, which considers mutual aid and unfinished implementation, valid?

We calculated the contributions of all members who took the course in the 2020 academic year (nine students). We presented the data to the students and requested them to respond to the aforementioned two questions on the Likert rank scale (strongly agree, slightly agree, slightly disagree, and strongly disagree) and provide free comments.

We obtained the responses from four students. Table 3 presents the results.

Table 3. The result of the questionnaire

Item of the questionnaire	Strongly agree	Slightly agree	Slightly disagree	Strongly disagree
Q1: Is the contribution calculated by this method valid?	25%	50%	25%	0%
Q2: Is introducing mutual aid and unfinished implementation to evaluation of contributions valid?	25%	75%	0%	0%

The results for Q2 in Table 3 indicated positive responses for considering mutual aid and unfinished functions in the evaluation of contributions. However, the results for Q1 indicated slightly negative responses. The comments described that the mutual aid had to determine the contributions by considering LoC and interview with members, and not by assigning fixed 30% of the FP, and that the contributions of unfinished implementation had

to be evaluated by creating another measure of the degree of completion, and not by using the number of files finished.

6. Summary

This study has proposed a method to quantify the contributions of individuals in the implementation phase of the PBL of software development using the FP method. The characteristics of the method is to consider mutual aid in learning and unfinished functions. We applied the proposed method to the actual PBL for a preliminary evaluation. The results of the questionnaire indicated that the students demonstrated positive responses for the concept of considering mutual aid and unfinished functions in the evaluation of contributions. However, certain negative comments for the quantity of the assignment were provided.

As future work, we will improve the assignment based on the comments provided for mutual aid and unfinished implementation. In addition, we would like to develop a tool to semi-automatically calculate the FP from the data stored in GitHub.

Acknowledgment

This study is partially supported by the Grant-in Aid for No. (C) 20K12089 from the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] Jane H. Hayes, Timothy C. Lethbridge, and Daniel Port. (2003) “Evaluating individual contribution toward group software engineering projects.” Proceedings of the 25th International Conference on Software Engineering: 622–627.

[2] Allan J. Albrecht and John E. Gaffney. (1983) “Software function, source lines of code, and development effort prediction: a software science validation.” IEEE Transactions on Software Engineering, (6): 639-648.

[3] Reze M. Parizi, Paola Spoletini, and Amritraj Singh. (2018) “Measuring team members’ contributions in software engineering projects using git-driven technology.” Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE): 1–5.

[4] Rose F. Gamble and Matthew L. Hale. (2013) “Assessing individual performance in agile undergraduate software engineering teams.” Proceedings of the 2013 IEEE Frontiers in Education Conference (FIE): 1678–1684.

[5] Yutsuki Miyashita, Yuki Yamada, Hiroaki Hashiura, and Atsuo Hazeyama. (2020) “Design of the inspection process using the GitHub Flow in project based learning for software engineering and its practice.” arXiv preprint arXiv:2002.02056.

[6] Kevin Buffardi. (2020) “Assessing individual contributions to software engineering projects with git logs and user stories.” Proceedings of the 51st ACM Technical Symposium on Computer Science Education: 650–656.

[7] Huijun Huang and Jiguang Zheng. (2018) “Quality earned value analysis based on IFPUG method in software project.” Proceedings of 2018 International Conference on Big Data Technologies: 101-108.

[8] Ramon Asensio Monge, Francisc Sanchis Marco, Fernando Torre Cervigon, Victor Garcia Garcia, and Gustavo Uria Paino. (2004) “A preliminary study for the development of an early method for the measurement in Function Points of a software product.” arXiv e-prints, cs-0402015.

[9] Irawati, Anie Rose, and Khabib Mustofa. (2012) “Measuring software functionality using function point method based on design documentation.” International Journal of Computer Science Issues 9.3/1: 124-130.

[10] GitHub. https://github.com/ (accessed 10 Nov. 2020).