International Journal of Advanced Smart Convergence Vol.12 No.2 187-192 (2023)
http://dx.doi.org/10.7236/IJASC.2023.12.2.187

IJASC 23-2-23

A Method for Inferring Development Progress in a Waterfall Model-based Software Development Environment

Seong-Hoon Lee*, Dong-Woo Lee**

*Professor, Division of Computer Engineering, Baekseok University, Korea
**Professor, Department of Computer Information, Woosong University, Korea

shlee@bu.ac.kr, dwlee@wsu.ac.kr

Abstract

Currently, our society is showing many changes due to the influence of information and communication technology (ICT). At the center of these information and communication technologies are software, intelligence, and sensing technologies. The software-related industry is steadily developing due to various software development policies implemented by the government and related organizations. Software development is desirable, but on the other hand, some negative aspects are also appearing. In this study, we proposed an objective way to infer the progress of software development for reasonable resolution of cases when a dispute related to the progress of development occurred during the software development process. The proposed solution was based on the waterfall model. The outputs generated in each process of the waterfall model are contents excluded from subjectivity. Therefore, it can be used as an objective method for calculating software development progress.

Keywords: *Software Development lifecycle, Software Engineering, Development Progress, ICT, IOT.*

1. Introduction

The core content in the process of industrial structure change that is currently appearing around the world can be called the 4th Industrial Revolution. Through the 4th industrial revolution, the current social and industrial structure is changing from a machine-oriented structure to a sensing and intelligent industrial structure. Therefore, the main core technologies of the 4th industrial revolution era include sensing-based Internet of Things (IoT) technology, intelligent technology, and 5th generation communication. All these technical aspects basically require software as well as electronic components.

In Korea, ICT-based industries, which are based on software, can be seen as leading domestic demand and exports while playing a role in driving the entire industry. The government is also supporting the software industry through various system improvements and software development programs for the development of

Manuscript Received: May. 25, 2023 / Revised: May. 28, 2023 / Accepted: June. 3, 2023
Corresponding Author: dwlee@wsu.ac.kr
Tel: +82-42-630-9713
**Professor, Department of Computer Information, Woosong University, Korea

the software industry. One example is the 'software-centered university' support project, which has been supported to nurture software talents at universities. Support for these various universities and industries is desirable. It is because the software industry can be one of the alternatives that can solve the food for the future society from our point of view, which absolutely lacks various resources. It seems that there will be positive results for software industry support as a whole, but on the other hand, negative aspects may emerge.

Computer software or software is part of a computer system. The software industry that develops such software must invest a long time and effort in terms of creating something out of nothing. Therefore, there are cases in which some companies voluntarily or intentionally try to achieve results without investing a long time and effort. Situations that disprove this are emerging as software protection. Most sincere companies are developing the software industry by investing a lot of time, money and effort. On the other hand, some companies copy or steal the software of existing companies without investing time and money, hindering the sound development of the software industry.

In this study, in order to support the sound development of the software industry, the progress of software development in the waterfall model-based software development environment can be inferred. In Chapter 2, we looked at the waterfall model of software engineering, which is the subject of this study, and the contents related to the software development progress problem. Chapter 3 describes how to infer objective development progress when a dispute related to software development progress occurs during the software development process. Lastly, the conclusion is described in Section 4.

2. Waterfall Model based Software Development

2.1 Waterfall Model Life Cycle

For software development, software engineering that deals with the entire software development life cycle is used as a basis. Software engineering is a discipline that systematically deals with the entire life cycle of software development, operation, and maintenance. There are various meaning interpretations related to this [1-3]. The life cycle of software development in software engineering is a description of the entire process performed to develop software [4-6]. Through this entire process, it can be said to be a tool to understand and share the contents and processes of software development. Among these models, the most common model is the waterfall model.

The waterfall model is a model in which the development process is performed sequentially, and is referred to as the sequential software development life cycle [1-3]. It is a structure in which the development cycle process starts from the requirements analysis stage and ends with software design, software implementation, software testing (test), software integration stage, and software maintenance stage. The basis of this waterfall model is that the processes performed in the entire software cycle are 100% sequential. Therefore, when one step is completed, the next step process is performed. Figure 1 below shows a typical waterfall model.

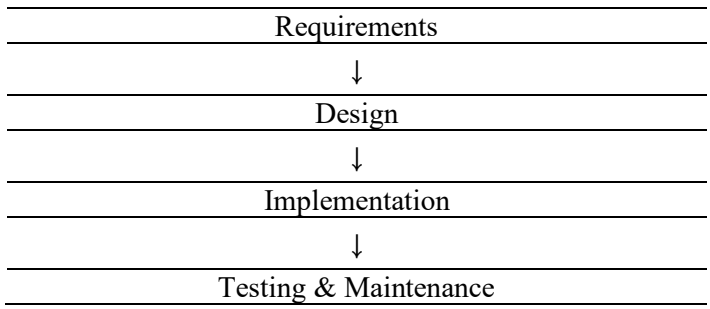

Figure 1. Waterfall model

Software development companies document their software development cycles for their development programs. Of course, there may be slight variations for each company, but it is common that it consists of contents that include the entire process of software development.

In this study, we tried to suggest an objective solution to resolve the dispute when a dispute occurred between the developer side (supplier) and the operator side (consumer) in relation to software development. For an objective solution, a method for estimating objective software development progress based on the documented contents by development life cycle was presented in Chapter 3.

2.2 Problems in The Software Development Process

Various problems have arisen in relation to software development. This includes issues related to whether software development results are copied or not, issues related to development progress between parties that occur in the software development process, and issues regarding the adequacy of software development costs [7-11]. Among these problems, this study dealt with software engineering-based solutions that can improve objectivity when disputes related to development progress between a developing company (supplier) and a development requesting company (consumer) that may occur during the software development process occur.

There are various problems raised regarding the determination of the progress or completeness of the development process. Problems and considerations that may be caused in performing judgment on issues raised related to the software development cycle are as follows. First, the diversity of evaluation item settings can be cited. Since the contents of the entire software development life cycle are different for each company, evaluation items are variously displayed, making it difficult to carry out standardization. There should also be a criterion that can be used to calculate weights for item elements. An example is the period required for each stage in the development cycle. However, if the required period for each stage in the development cycle is not described, other elements are required. Therefore, standards that can be used to calculate objective weights for evaluation factors must be prepared or secured.

3. Software Development Progress Inference Method

In relation to the entire life cycle of software development, when a dispute arises between a software supplier and a consumer about the progress of software development, the most important thing in dispute resolution is that an objective data-based method should be presented. Only then will both suppliers and consumers accept the solution. In this study, an objective data-based solution was presented. The process to determine the software development progress problem related to the software development cycle goes through four steps. The process of inferring the progress of software development is to set the evaluation items first. Next, a weight value is assigned to each evaluation item. Next, the development progress for each evaluation item is inferred. Finally, the development progress for the entire software is calculated. The entire process is shown in Figure 2 below.

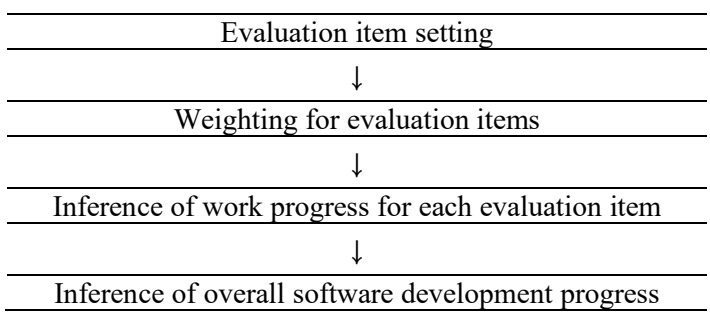

Figure 2. Software development progress inference process

The first step is to derive evaluation items. The software development cycle currently being dealt with basically means a series of tasks that are promoted step by step in the entire process required to develop software. The step-by-step process and the results generated in this process are shown in Table 1 below. Also, in Table 1, the main items included in the results of each stage that can be used to infer the progress of software development. Evaluation items for judging problems related to the development cycle can be used as objective evaluation items because the outputs of each stage described in the table below are clear and objective.

Table 1. Key data at each stage to infer software development progress

Step name		Output	Key check point
Step 1	Basic design	Basic design report	• Main features (contents)of system • Network diagram, database
Step 2	Detailed design	Detailed design report	• Number of DB tables in the DB design document • Number of programs
Step 3	Program design	Program design report	• Number of programs written in the programming language
Step 4	Implementation	Source program	• Number of developed source programs
Step 5	Testing	Test result report	• Whether to run the test job
Step 6	Maintenance	Operation plan report	• System operation manual

Next, in the second step, weights are assigned to the evaluation items determined in the previous step. In this part, it is based on the content of documents such as 'software development plan' or 'development plan specification' attached to the contract, which is the content agreed between the parties. These documents usually describe information on the development period and input manpower for each stage of the software development cycle. Therefore, by using this information as basic data to determine the weight for each item, it is possible to secure more objectivity by preventing subjective matters that may be involved in judging problems.

First, the method based on the required period (input time) for weight calculation is as follows. If only the program development period is specified in the contract, which is a document agreed between the parties, weights are set based on the development period required for each stage of program development. Therefore, it means the ratio of the time required to develop the function corresponding to each item to the total time required for program development.

Next, it is a case of applying a weight calculation method based on input manpower information. Although the program development period is not specified in the contract, which is a document agreed between the parties, if there is input manpower information for each function, the weight calculation that reflects this can be an objective content excluding subjectivity. This means the ratio of the manpower invested in developing the function corresponding to each item to the total manpower input according to program development.

In order to calculate the progress of software development by considering the importance of each evaluation item, it is important to assign the importance based on objective data excluding subjectivity when assigning the importance to each item. To this end, the importance can be assigned using general characteristics that appear during software development. As a general characteristic, the required period and input manpower are relatively small for simple tasks during software development, and most of them are performed by less than intermediate level personnel. On the other hand, for important or complex functions, it is common for the required time and the input of high-quality manpower to increase relatively.

In the process of developing software and programs, the input manpower decides by considering the

characteristics of the work that appears at each stage. The expertise of these input manpower is usually divided into beginner, intermediate, and advanced levels. Therefore, it is necessary to reflect the importance of each item in consideration of the expertise of these personnel. For example, the importance of entry-level personnel is 1, intermediate personnel is 2, and advanced personnel is 3, reflecting the importance differently depending on the expertise of the personnel. However, the appropriateness of the gap in importance according to the expertise of human resources seems to require more discussion.

As the third step, it is the process of judging the progress of the work for each evaluation item. Since subjectivity may be involved in this process, it is important to make efforts to calculate the progress based on objective data as much as possible. In this regard, a few suggestions are described below.

Evaluation of the progress of the basic design items is based on the data of the requirements analysis document or basic design document agreed between the parties for the functions to be developed in each system. Therefore, the progress is calculated by checking whether the contents that the development system should include, the network configuration diagram, and the required database are included. Therefore, the main contents that should be included for calculating the progress are shown in the table below.

Table 2. Major checklist when evaluating the progress of basic design

Checklist	• Contents and network configuration diagram that should be included in the development system • Check if the required database etc. are included

The evaluation of the progress of the detailed design stage is based on the contents shown in the detailed design document. Therefore, the main contents that should be included to calculate the progress are shown in the table below.

Table 3. Major checklist when evaluating the progress of detailed design

Checklist	• Compare the number of databases described in the detailed design with the number of DB tables in the actual DB design • Calculated by referring to the number of programs to be included in the system in the plan and the number of actually implemented program designs and source codes

The progress calculation for the implementation stage is the content of determining whether the main functions indicated in the basic design are actually implemented. Therefore, the main contents that should be included for calculating the progress are shown in the table below.

Table 4. Major checklist when evaluating the progress of implementation stage

Checklist	• Calculated by comparing the number of programs that must be included in the system with the number of source codes developed in actual programming languages

Finally, the progress calculation related to testing and operation can be seen as a reflection of the results of whether the functions implemented as programs can be utilized. Therefore, the main contents that should be included for calculating the progress are shown in the table below.

Table 5. Major checklist when evaluating the progress of the test and operation stage

Checklist	• Whether or not there is a test work on whether the developed programs can be used from the user side • Whether training is required for system operation according to the plan

4. Conclusion

The government and related organizations are continuously making many efforts to foster the software industry. As a result, domestic related industries are growing with more specialization, and new companies are being created. On the other hand, there is also an undesirable problem of doing business by easily duplicating or stealing software without spending a lot of time and effort for software development.

In this study, we studied the solutions to possible disputes in the software development process. In particular, we proposed a solution to disputes that may arise between the development company and the client company over the software development progress problem in the entire software development cycle process. For an objective solution, the results from the entire software development cycle were used. Therefore, it will be possible to increase the dispute resolution rate by excluding subjective content and maintaining objectivity in inferring the progress of software development.

References

[1] R. S. Pressman, Software Engineering – A Practitioner's Approach. 3rd edn. McGraw-Hill, Inc: New York, 1992.
[2] Sommerville, Software Engineering, Addison-Wesley Publishing Co, 2001.
[3] J. Whitten and L. Bentley, System Analysis & Design methods, IRWIN, 1994.
[4] T. Gildersleeve, Successful Data Processing Systems Analysis, 2nd edn, Englewood Cliffs, New York, Prentice Hall, 1985.
[5] E. Sebastian, The Economic Properties of Software, Jena Economic Research papers, 2008.
[6] C. Satish C and M. Anand, "Software Documentation Management Issues and practices: A Survey. Indian Journal of Science and Technology," 9 (20), pp. 1-7, 2016.
[7] F. Culwin, A. MacLeod and T. Lancaster, "Source Code Pragiarism in UK HE computing school, issues, attitudes and tools," Joint Information Systems Committee, Technical report, 2001.
[8] F. Culwin, "Pragmatic Anti-pragiarism in Action," Proceeding of 3rd al Ireland conference on the teaching of computing, Dublin, 1995.
[9] P. Vamplew and J. Dermoudy, "An Anti-Plagiarism Editor for Software Development Courses," Seventh Australasian Computing Education Conference (ACE 2005), 2005.
[10] A. Hellas, J. Leinonen and P. Ihantola, "Plagiarism in Take-home Exams: Help-seeking, Collaboration, and Systematic Cheating," the 2017 ACM Conference, 2017.
[11] A. Jadalla and A. Elnagar, "PDE4Java: Plagiarism Detection Engine for Java source code: a clustering approach," International Journal of Business Intelligence and Data Mining 3(2):121-135, 2008.