International Journal of Multidisciplinary Research and Literature
IJOMRAL
Vol. 1, No. 1, January 2022 hal. 1-120
Journal Page is available at http://ijomral.esc-id.org/index.php/home

IMPLEMENTATION OF METRICS ON SOFTWARE DEVELOPMENT IN THE PROVINCIAL ADMINISTRATION SYSTEMS

Adnan Abdullah[1]
[1]School of Electronic Engineering, Tianjin University of Technology and Education, China
Email: adnanabdullahkhan@hotmail.com

ABSTRACT

Metrics in software development projects related to the effort and thought required to complete the project and the resources used. The methodology applied, for example, the time needed to complete, experts required, costs incurred, and methods used in software development.

This research is an observational study to implement the theory of software metrics in a software development project. The expected result is a guide used to calculate the number of human resources needed in software development, which is calculated in units of person-hours.

According to the analysis above, software estimation is used to forecast the resources and costs necessary to complete a software development project the integrity of the new software estimation results. To obtain the most precise estimate, to study and calibrate the estimation formulation constant is necessary to light the current software development project's conditions.

Keywords: Metrics, Software

INTRODUCTION

Measurement is one of the foundations of all engineering disciplines. Software engineering in IEEE Standard 610.12 is defined as follows: " *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software*. " (Pressman, R. S, 2000). Like the engineering profession, the software engineering process also has metrics.

Metrics are used by the software industry to measure the process of creating, operating, and maintaining software. Through metrics, valuable information and parameters can be obtained as objective evaluation material regarding the attributes and status of software development. Implementing software metrics in a software development process and a software product involves complex stages that require continuous learning, which can provide knowledge about the status of a software development process and or a software product.

By evaluating the attributes that exist in the software, one can obtain the status of the software. From this, existing situations can be identified and classified, which can be used to assist in finding new opportunities that can use for software development and improvement. This kind of evaluation can help ultimately make plans for changes that may need to be implemented in the future. These identified attributes can also use as a reference and consideration for other software development processes.

Software metrics have broad limitations. Software metrics depend on the software attributes you want to assess for quantity and quality. In general, software metrics are divided into two classes: those used in software development projects and those used in software products.

Metrics in software development projects related to the effort and thought required to complete the project and the resources used. The methodology applied, for example, the time needed to complete, experts needed, costs incurred, and methods used in software development.

Many types of software attributes can measure. The metrics applied to depend on the nature of the software product being created. For example, you want to know how many requirements a project has, the software specifications (to minimize ambiguity), and the completeness used to fulfill all the required functions. In the product of an application, you may want to know the number of lines of code, complexity, functionality fulfilled, the number of errors that may occur, and the number of tests performed to ensure that all requirements are implemented. On the other hand, software reliability can be measured when the product has been distributed to consumers. Experts in software engineering have not yet succeeded in deciding on a precise metric that is universally acceptable. In addition, each person usually uses various specific methods to measure the different attributes of the software they make.

The problem in this research is the implementation of metric software theory in a real software development project. We are implementing a software metrics-based approach in a software development effort. The expected result is a guide used to calculate the number of human resources needed in software development, which is calculated in units of person-hours.

Metrics in Process and Projects

Measurement is one of the things that is part of the engineering world, with some of them being: power measurement, physical dimension measurement, weight measurement, frequency measurement, and so on. However, size is rarely done in the world of software engineering. The problem that is often encountered is the difficulty in determining the value of the object being measured and the difficulty in determining the parameters that can count. One way that can use is to collect metrics in a software development process or project to obtain a process indicator and project indicator.

Process indicators are used by organizations or companies engaged in software engineering to obtain data to increase efficiency in the software development process.

1. Process Metrics and Software Process Improvement

   According to Paulish et al. (2009), the process of software development is described in Figure 1 as follows:

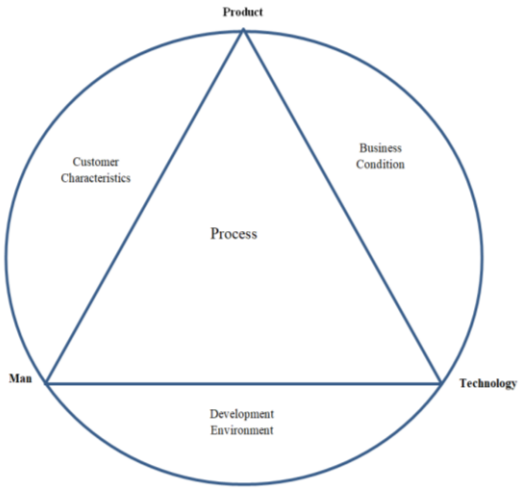

Figure 1 Software Development Environment

The figure shows that the process is in the middle of a triangle connected to three main factors: human resources, product complexity, and technological developments. The most

influential quality and process performance are human resources of these three factors. The process triangle is contained in a circle of environmental conditions of the running process, which consists of the system development environment, business conditions (e.g., *deadlines*, business rules), and customer or user characteristics (e.g., ease of communication).

It is possible to measure a process's efficiency indirectly by concluding it. Errors that did not fix before the software was released and users' reported errors could all be used to make decisions. Results can also be derived from more specific productivity measures, labor used, time spent, and other aspects of time management.

Grady (1992) argues that private and public uses are used to obtain the data. Some process metrics can be personal to the team in the project, but they can share with all team members. A technique is known as *Statistical Software Process Improvement* (SSPI) was developed to analyze an organization's processes. SSPI uses error analysis contained in the software to collect information about all errors and flaws that can occur in an application or system.

2. Project Metrics

Project managers and development teams use project metrics to adapt to technical workflows and activities. This project metric was first used during project planning. The metrics collected from past projects are used as a reference for estimating current projects. Project managers use these data to monitor and control projects.

Project metrics have two kinds of goals. First, the metric is used to optimize the schedule to shorten the schedule. Second, metrics are used to assess and improve product quality. If quality increases, errors can reduce, then the number of revisions that need to be made is also reduced to minimize production costs. According to the model suggested by Hetzel [1993], the things that should measure in a project are:

1. Input, measuring the resources (labor, environment) required in the project
2. Output, measuring the results obtained from a project during the software engineering process
3. The results indicate the product's effectiveness. The model above can be applied to both processes and projects.

Software Measurement

Measurement can separate into two categories, namely direct measurement and indirect measurement. Direct measurements in the software engineering process relate to costs and required resources, such as the number of lines of code, execution speed, memory size, and errors encountered over time. Indirect measurement of a product is related to functionality, quality, complexity, efficiency, reliability, etc. Direct measurements are easier to do because the results can obtain directly. In contrast, indirect measures are more difficult because they have to undergo a more complex process.

Metrics are separated into process, project, and product metrics. Product metrics are private to individuals and are often combined to create public project metrics for the development team. The project metrics are then consolidated to develop general process metrics for the entire organization or company. The difficulty is usually faced when combining the measured metrics because there are often differences between individuals. This problem is usually overcome when normalization is carried out in the measurement process. With normalization, comparisons of metrics can be carried out on a wider scope.

1. Size Oriented Metrics

Size-oriented metrics are obtained by normalizing quality and productivity measures by calculating the size of the software created. The size usually used as a reference for normalization is LOC (*lines of code*). From measuring the number of in a software, can obtain it:

a. Errors per KLOC (thousands of LOC)

b. Deficiencies or defects in specifications per KLOC
c. Price per LOC
d. Number of documentation pages per LOC

Additionally, some of the metrics that can be calculated are:

a. Errors per person-month
b. LOC per person-month
c. Price per documentation page

According to Jones (2007), measure-oriented metrics cannot be universally accepted as the best way to measure software engineering processes. The reason is that sometimes program functionality can achieve with fewer program lines. In addition, to estimate the LOC must be used a high-level design analysis.

2. Function Oriented Metrics

Function-oriented metrics use the functionality measure generated by the application as a normalized value. Functionality cannot measure directly, so to obtain it, direct measurement is used first, then the results of the direct measurement are used as input. Function-oriented metrics were first proposed by Albrect (1979), who suggested a measure called *function point* (FP). FP is obtained using empirical relationships based on direct measurements and estimates of software complexity.

Software Project Estimation

Estimation of costs and labor is difficult to be an exact science. Many factors, such as human, technical, environmental, or political, can affect the number of fees and labor required to develop software (Murti, 2016). However, software project estimates can transform from something abstract into a sequence of systematic steps that produce estimates with an acceptable level of risk.

Several ways can use to perform project estimates:

a. Estimated delay in a project can be completed
b. Calculations based on similar completed projects
c. Using simple decomposition techniques to create a range of costs and labor
d. Using one or more empirical models for cost and labor estimates

The technique suggested by experts in the software field is to use decomposition techniques and empirical models. The decomposition technique uses an approach by dividing a project into several major software engineering-related functions. Empirical estimation models can be used as a complement to the decomposition technique.

Empirical Estimation Model

The software estimation model uses an empirically derived formula to estimate power as a function of LOC or FP. The most supportive empirical data in the estimation models are obtained from a limited number of project samples. For this reason, there is no one-size-fits-all estimation model for software development environments. Therefore, the results obtained from the provided models must be used wisely according to each developer's environmental conditions.

Labor Time Conversion

The conversion of labor time in this final project obtained from the comparison figures used in the ConvertAll software, with the relationship between person-months (OB), person-hours (OJ), person-weeks (OM), and person-years (OT) are as follows:

$$OM=40OJ$$
$$OT=12OB$$
$$OT=52OM$$

METHOD / SYSTEM DESIGN

Software Requirements Specification

SIMANCA is used as a tool to automate data processing in planning activities in local governments.

1. Existing System

   The system used as a reference for SIMANCA development is a manual planning system. After a Development Planning Deliberation is held in an area, proposals for activities are produced from the Regency. These proposals are combined with proposals from provincial agencies as input for the actions of the Provincial Development Planning Deliberation. These proposals are then selected by conducting an assessment using a weighted method, which is then issued as a national planning document.

   In general, the flow of the planning system is described in the following figure:

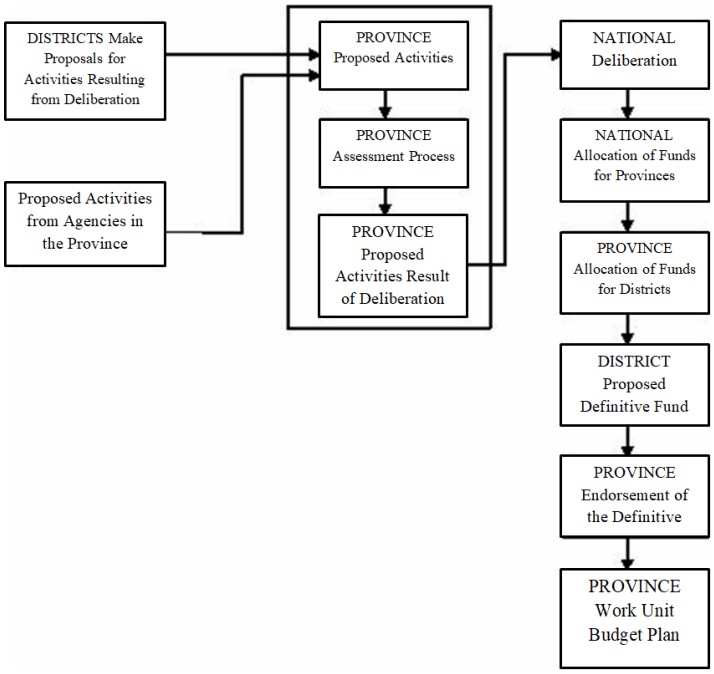

Figure 2 Flow of Regional Planning Process

The use of SIMANCA is expected to facilitate the relevant agencies in the data input process and the assessment process. The automated system is expected to make the system usable in real-time so that reports can obtain immediately after the data entry process is completed.

2. Target User

   The target users of SIMANCA are the Regional Government Planning Board, the Activity Proposal Assessment Team, and agencies in the Regional Government, which use the SIMANCA interface to enter activity proposals and activity assessments. Characteristics of users are employees of work units in local governments who master computers to access the web.

3. Proposed System

   The proposed system is an information system with facilities for entering activity proposals, activity assessments, and printing the results of activity assessments. The permissions granted are administrator, operator, and rater. The system administrator has the facility to enter additional data needed for submitting activity proposals and activity assessments, which include: fiscal year data, field and work unit data, user data, appraiser data, and assessment parameters.

   The system is designed using a web-based interface, with access to the database in real-time, so that the latest data conditions can be obtained directly when the user accesses the system.

4. Software Environment

   SIMANCA is run on a *server* environment with a Linux operating system and a web-based interface, which can access via a computer network. The web server used is Apache PHP, and the database used is Firebird.

5. Computer Network and Hardware Environment

   The hardware required as a SIMANCA data storage center is a computer capable of functioning as a web server and Firebird database server, which users can access via intranet or internet computer networks. The proposed network scheme is as follows:

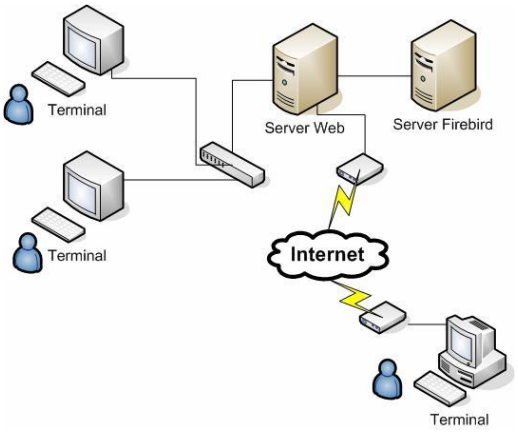

Figure 3 Proposed Computer Network

   The device used to access SIMANCA is a computer or equipment that can connect to the SIMANCA server via an intranet or internet, which has a web browser facility capable of displaying HTML and executing javascript.

6. System Documentation

   The documentation planned to be made instructions for use, intended for SIMANCA users. The documentation includes brief information on regional planning, installation instructions, and SIMANCA operating instructions.

Software Design Specification

The design specifications contained in SIMANCA are used as a reference for project managers and development teams to be applied to the SIMANCA development process.

1. Data Design

   SIMANCA data is stored in a Firebird database. The database creates to meet the data administration needs described in the software requirements specification. The database design is made by collecting the entities related to SIMANCA and connecting them regarding the relationships between these entities. The entities in question are: year, field, work unit, program, proposed activity, criteria, sub-criteria, appraiser, assessment, and users.

2. Interface Design

   The interface is designed to have 4 (four) parts, namely: *header*, menu, body, and *footer*. *The header* contains the title and SIMANCA user information. The menu is inside the *header*, and it is used as a link to access the modules in SIMANCA. The content section is used to place modules containing the desired functions in SIMANCA. The standard of interface design is divided into 3 (three) parts: displaying data, adding data, and changing data. The menu for deleting and adding data is placed as a link on the data display page.

3. Menu Design

The menu is displayed when the user successfully logs in to the system after correctly entering the user and password. The displayed menu is different for each user level. Users are given a menu according to the access rights they have.

General Menu

What is meant by a general menu is a menu that links to modules that all levels of users can access. All users can access the modules, including the change password module and report printing module.

1. Administrator menu. Administrators can access the modules: setting budget year, setting field, setting work unit, setting program, setting activity proposal, setting criteria, setting sub-criteria, and setting assessor.
2. Operator menu, the modules that the operator can access include program settings and activity settings.
3. Appraiser Menu. The module that administrators can access is the activity proposal evaluation module.

RESULT AND DISCUSSION

The Observations were made through the *SIMANCA prototype* development process. Statements are made by calculating the number of days and hours required based on each team member's previously prepared presence tables.

The human resources involved in developing the *SIMANCA prototype* are 4 (four) people with a composition of 1 (one) system or database analyst and 3 (three) programmers. The results of accumulated attendance are shown in table 1

Table 1 Results of Accumulated Presence of the Development Team

Person	Number of days	Number of Hours
A	18	79
B	18	81
C	16	76
D	17	83
Total people/hour	**-**	**319**

Comparison of Estimated Results

From the calculations carried out, the estimated value of E_{OJ} obtained from several models is shown in the following table:

Table 2 Comparison of estimation results between models

Model	Estimated E_{OJ}
LOC-Based Estimation	
General LOC analysis	630,92
Waltson-Felix models	1891,32
Bailey-Basili model	1278,79
Boehm's Simple Model	1304,47
COCOMO models	1294,69
FP-based estimation	
Model Albrect and Gaffney	2358,99
Kemerer Model	98730,44

Matson, Barnett, and Mellichamp . models	1399864,94
Process-based estimation	
Process-based estimation	649,99

The calculation shows that each model has different calculation results for the same LOC and FP. Further developments are obtained because each model uses various indicators when making estimates. In addition, each estimation result is obtained based on the experience of each creator of the estimation model. Suppose the estimated data is compared with the results of observations on the project. In that case, a difference is obtained, where the results of the estimated power are greater than the reality that occurs in the field. One factor that can be accepted is that the team recruited to carry out SIMANCA development already has experience working on similar projects. The total system time can be smaller than estimated.

The smallest power estimation results are obtained in the general LOC estimation model. From the observations on the estimation of the SIMANCA development project, the results of the LOC analysis implemented in several models have a smaller mean E_{OJ} than in a similar study using FP. The Kemerer and Matson, Barnett, and Mellichamp models provided the most accurate estimates. In the Kemerer and the Matson, Barnett, and Mellichamp models, the estimation results show some E_{OJ}, which is much larger than the average estimate using other models, as shown by the calculations.

In process-based estimation, independent analysis is used by dividing the process into smaller activities. Counting E_{OJ} performed directly on each exercise, then summed, so the total E_{OJ} is now knowable. The SIMANCA development project estimate shows that the number of E_{OJ} on the process-based approach estimates the number of E_{OJ} on LOC-based estimation.

From the results of the implementation of the SIMANCA project, it is found that the analysis model closest to the developments in the field is a LOC-based analysis that uses historical data indications. The analysis results that are farthest from the results obtained from the area are FP-based estimates using the Matson, Barnett, and Mellichamp models. However, suppose the average of each estimation model is taken. In that case, the process-based estimation model has an average closest to the results in the field, followed by the LOC-based average estimation. The FP-based estimation average is the one that is farthest from the observations in the SIMANCA project implementation.

CONCLUSION

Based on the implementation and discussion contained in this final project, it can be concluded that:

1. Software estimation is used to estimate the resources and costs required to complete a software development project.
2. The truth of the new software estimation results can be known when a software development project has been completed.
3. The formulation of the estimation constant must be studied and calibrated according to the conditions that apply to the current software development project to obtain an accurate estimate.

ACKNOWLEDGMENTS

The author would like to thank all people and society for their support and encouragement throughout the process until the completion of this research. It is hoped that it can use as reading material to add insight and views of readers and a source of reference and

information for the same research size and a comparative study in the context of studying science.

REFERENCES

Albrecht, A. J., (1979) *Measuring Application Development Productivity*, IBM Applications Development Symposium, Monterey, CA

B. Hetzel, (1993) *Making Software Measurement Work: Building an Effective Measurement Program*, QED Technical Publishing Group, Boston, Massachusetts

Berenbach, Brian, Paulish, Daniel J, Kazmeier, Juergen, Rudorfer, Arnold, (2009) "Software & Systems Requirements Engineering: In Practice," McGraw-Hill, United States

Charles P. Jones. (2007). *Estimating Software Costs*, McGraw-Hill

Grady, Robert B., (1992) *Practical Software Metrics for Project Management and Process Improvement*, Prentice-Hall, Inc.

Murti, Sambodo Vishnu. (2016). "Comparison Analysis of Weight Value Changing in Function Point Analysis Between Fuzzy Inference System Mamdani and Tsukamoto for Software Size Estimation." 5(2): 104–10

Pressman, R. S. (2000) *Software Engineering A Practitioner's Approach*, 5th Edition, New York, McGraw Hill,