THE MEASUREMENT IN SOFTWARE ENGINEERING

Oya Kalipsiz

Department of Computer Science, Yildiz Technical University, Istanbul, 80750 Turkey
Phone (90) 212 2597070 Fax (90) 212 2274470 e-mail: kalipsiz@yildiz.edu.tr

Abstract - *Measurement can be used throughout a software project to assist in estimation, quality control and project control. Software measurement is concerned with capturing information about attributes of entities. Cost estimation models are developed using data from completed software projects and accuracy tests are usually done using data sets with known project size. Software metrics provide a quantitative way to assess the quality of internal attributes. This paper will focus on ways and methods to measure the software system.*

Keywords - Measurement of electrical and non-electrical quantities - Software Project Management - Software Metrics

1. INTRODUCTION

Software engineering (*SE*) is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software. It is the establishment and use of engineering principles in order to obtain economically software [1, 2].

Software production suffers from excessive costs, low productivity and continued poor quality. It is often out of control because we do not measure. Project planning is the main problem for the software managers [3]. Successful project planning relies on a good estimate of the effort required to complete a project.

Individuals involved in software development will have specific objectives for measurement [4,5].

Software measurement activity is to identify the entities and attributes of interest, which we wish to measure. The entities of interest in software can be classified as processes, products or resources.

Cost estimation for software has special difficulties because, unlike conventional manufacturing industries, software production usually involves producing new products using new methods and tools. Good software cost estimation depends on the availability of accurate records of past projects and using a repeatable, defined development process.

This paper describes objectives of measurement, software quality, classifications of software measures; summaries software cost estimation techniques and discusses a framework for software measurement process.

2. FUNDAMENTALS OF MEASUREMENT

Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world. An entity may be an object or an event. The attribute is the feature or property of entity. Measurement assigns numbers or symbols to these attributes of entities in order to describe them. Software metrics is a quantitative measure of degree to which a system, component or processes possesses a given attribute [6].

Direct measurement of an attribute is measurement, which does not depend on other attribute. Some attributes can be measured in terms of one other attribute. Direct measures of the include cost and effort applied Direct measures of the software engineering process include cost and effort applied. Direct measures of the product include lines of code (LOC) produced, execution speed, memory size and defects.

Indirect measures of the product include functionality, complexity and reliability [7].

2.1 Objectives of Measurement

Measurement has become a natural part of many *SE* activities. Individuals involved in software development must have specific objectives for measurement [4]. Customers look to measurement to help determine the quality and functionality of products. Maintainers use measurement to inform their decisions about reusability, reengineering.

Software developers need to measure:

- Product and process attributes for purpose of certification
- Attributes of existing products and current processes to make predictions about future ones.

Managers need to measure [4,5]:

- The cost of various processes within software production
- The productivity of staff
- The quality of software products

The productivity of individual engineers working in an organization is affected by the following factors [1]:

- Application domain experience
- The development process quality
- The size of project
- Technology (tool) support
- Working environment

ISBN 972-98115-4-7

2.2. Software Quality

There are a number of quality factors of software products, such as reliability, usability. These factors are determined by lower level criteria such as modularity, accessibility. Actual measures (metrics) are proposed for the criteria. Software quality factors focus on three aspects of software product [2]:

- Its operational characteristic (operation)
- Its ability to undergo change (revision)
- Its adaptability to new environments (transition)

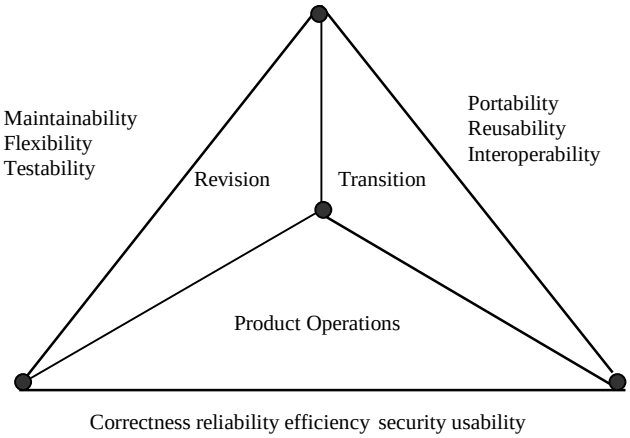

Fig. 1 - Factors in Software Quality [2].

It is difficult to develop direct measures of the software quality factors. A set of metrics are used to develop expressions for each of the factors according to the following relationship:

$$F_q = c_1 * m_1 + c_2 * m_2 + ...+ c_n * m_n \quad (1)$$

Where F_q is software quality factor, c_n are regression coefficients, and m_n are the metrics that affect the quality factor.

3. CLASSIFICATIONS OF SOFTWARE MEASURES

In any software measurement activity, the first obligation is to identify the entities and attributes. Entities can be classified as products, process and resources [4, 7].

Products are deliverables such as requirement specifications, design documents and code. Products are more concrete than process and resources and are easier to measure.

Processes are software related activities such as development and maintenance. They have a time factor. Examples of Process include reasonably well-defined and coherent activities like that of developing a software system from requirements [1]. Process measures include large-grain quantifications and small grain evaluations of particular process activities such as Capability Maturity Model (CMM) and test effectiveness [4, 7].

Resources are inputs for software production such as people, equipment and tools. Some entities, for example a CASE tool that are considered to be resources for some processes are clearly products of other processes.

Measurement is obtained by measuring the attributes of entities. Attributes are either internal or external (table I). Internal attributes can be measured purely in terms of product. External attributes can be measured with respect to how the entity relates to its environment.

Software managers and users most like to measure and predict the external attributes. They would like to know the cost-effectiveness. External attributes cannot be measured directly. There is a relationship (1) between internal and external attributes [8]

Table 1: Components of Software Measurement [4].

ENTITIES	ATTRIBUTES	
Products	**Internal**	**External**
Specifications	Size, functionality redundancy, correctness	Maintainability, comprehensibility
Designs	Modularity, coupling reuse,	Quality, complexity
Code	Algorithmic complexity, size, reuse, modularity, control flow structures	Reliability, maintainability, usability
Test data	Size, coverage level	Quality
Processes		
Constructing specifications	Time, effort	Cost, stability
Detailed designs	Number of fault	Cost-effectiveness
Testing	Number of bugs found, size, effort	Stability, cost
Resources		
Personnel	Age, price	Experience, intelligence
Teams	Size, communications level	Productivity, quality
Software	Price, size	Usability, reliability
Hardware	Speed, memory size	Reliability

4. SOFTWARE COST ESTIMATION TECHNIQUES

Organizations need to make software effort and cost estimates. There is no simple way to make an accurate estimate of the effort required to develop a software system [9, 10].

The major techniques relevant to software cost estimation may be summarized as:

- Expert judgment: Several experts on the proposed software development techniques and application domain are consulted. They each estimate project cost. These estimates are compared and discussed.
- Estimation by analogy: This technique is applicable when other projects in the same application domain have been completed. The cost of a new project is estimated by analogy with these completed projects [11].
- Statistical cost modeling: A statistical cost model can be built by analyzing the costs and attributes of completed projects

4.1 Statistical cost modeling

A model is developed using historical cost information that relates some software metric to project cost. A mathematical formula is used to predict costs based estimates of project size, effort. Most statistical cost estimation models have an exponential component [1, 4, 12]. This reflects the fact that costs do not increase linearly with project size. As the size increases, extra costs incurred. Cost models can be expressed as (2):

$$\text{Effort} = C*PM^{s}*M \qquad (2)$$

In (2), C is a complexity factor, PM is a software metric, s reflects the increasing effort required for large projects, and M is multiplier. Product metric may either be a size metric (Lines of Code, LOC) or a functionality metric (Function Point, FP).

To use any estimation model effectively, it must be tuned to reflect local circumstances. The calibration process must be based on a database of schedule and effort measurements made for completed projects [13].

5. A FRAMEWORK FOR SOFTWARE MEASUREMENT PROCESS

A measurement process consists of several measurement activities and steps to perform estimation efficiently in software projects. A framework is used for designing and maintaining a measurement process [14, 15]. A framework for designing a successful software measurement process involves three main phases: planning, implementing and improving (fig. 2).

5.1 Planning

The measurement process should be planned where to start, how to start, and what is the suitable schedule.

The measurement process originates with a need for measurement. Needs have to prioritized and must not be confused with idealistic wishes. Scope identification is the first activity in planning and includes identification of objectives and measures. Then, operational definitions and procedures of measurement process should be constructed and documented

Every measurement programme must have very clearly stated objectives and goals. Objectives may be specific to individual projects or they may have a company-wide angle. Wherever possible objectives should be expressed in terms of specific products, processes, and resources, together with attributes of these Measures that support decision making with respect to goals and objectives are identified which will lead to measures being identified [4].

The measurement process should be planned where to start, how to start, and what is the suitable schedule.

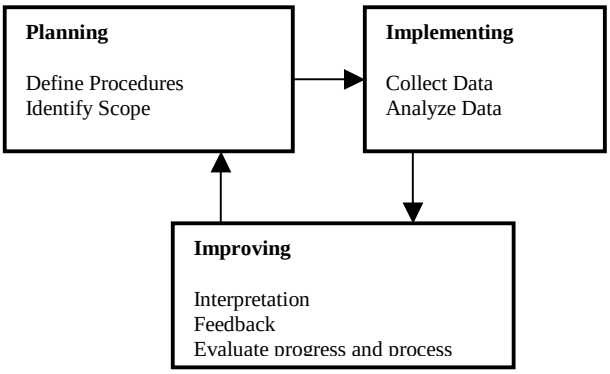

Fig. 2 - Software Measurement Process

5.2 Implementing

The implementing of measurement process includes the following sequence of tasks [4]:

- Collecting and storing of software measurement data
- Analyzing and reporting data

Data collection provides the direct measurements on which all else depends. Each item of data must contribute to a direct measure of some attribute of the process, products, or resources with we are concerned.

In setting up the data collection activity, we must address the following tasks [1, 16]:

- Decide which products to measure
- Ensure that the product is under configuration control
- Decide which attributes to measure
- Devise coding schemes
- Choose suitable scales
- Design forms to suit the project

Individual organizations must build and maintain a database of quantitative software information. The database of measurements must be reasonable large. Once a sufficiently large database has been established, comparisons across projects may be made. To get the best return on investment, a metrics program must ensure that is collecting useful and accurate data [15].

The statistical techniques can be used to describe the distribution of attribute values, and the relationship between two attributes. Datasets of software attribute values must be analyzed with care because software measures are not usually normally distributed. We must be very cautious about the use of techniques, which assume an underlying normal distribution. There are a number of approaches to dealing with non-normality:

- We can use robust statistics and non-parametric methods
- We can attempt to transform our basic measurements, into a scale in which the measurements conform more closely to the normal distribution

5.3 Improving

The recommended cost estimation processes are based on the concept of continuous feedback [17]. Local models may be refined each time a new project is completed. Also, Feedback should include an assessment of the quality of the overall cost estimation process [14, 7].

The measurement process improvement includes two aspects. The first one is the evaluation of progress towards goals and measurement needs. The second one is improvement towards better process efficiency, reliability and utilization. The improvement mechanism should investigate whether these defined ones are also the right ones and the most efficient ones.

The assessment mechanism evaluates the whole estimation process and estimates. The assessment model depicted in Fig. 3 suggests that the software practices are to be evaluated based on qualitative and quantitative data [18].

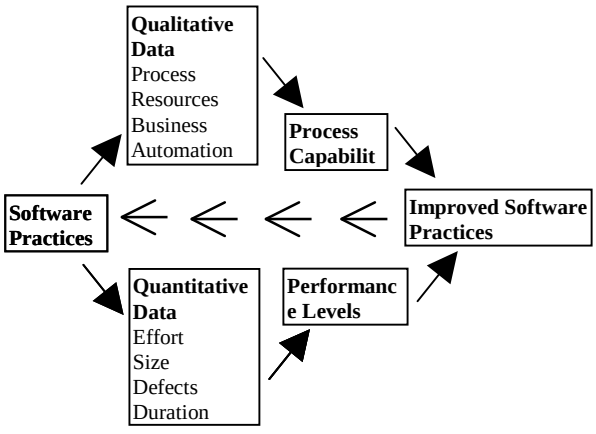

Fig. 3 - A Complete Process Assessment Model [17].

6. CONCLUSION

Successful project planning relies on a good estimate of the effort required to complete a project. Measuring is the pre-condition for improving estimating accuracy.

In any software measurement activity, the first obligation is to identify the entities and attributes. Entities can be classified as products, process and resources

Much software metrics work has lacked the rigor associated with the measurement in other engineering disciplines. The development of software tools for measures is still in its infancy.

An effort estimation process consists of activities that are needed to make or formulate, store, report and analyze estimates and to use them in project tracking and management activities.

An effective software metric must be simple and computable, persuasive, consistent and objective. It should be programming language-independent and provide effective feedback to the software engineer.

The most important objectives and challenges for software measurement in the future are as follows:

- Quality control and assurance (certification of systems)
- Evaluating methods
- Design methodologies and software measurements

REFERENCES

[1] I. Sommerville, *Software Engineering*, Addison Wesley, 1995.

[2] R. Pressman, *Software Engineering*, Mc. Graw Hill, 1997.

[3] J. Hallows, *Information Systems Project Management*, AMACOM, American Management Association, 1998.

[4] N. Fenton, *Software Metrics; A Rigorous Approach*, Chapman & Hall, 1993

[5] L. Pfleeger, R. Jeffery, B. Curtis and B. Kitchenham, " Status Report on Software Measurement", *IEEE Software* March/ April 1997 pp: 33– 43.

[6] IEEE, Software Engineering Standards Collection, Los Alamitos, CA, IEEE Press, 1994.

[7] J. Verner and G. Tate, "A Software Size Model", *IEEE Transactions on Software Engineering*, vol.18, No.4 April, 1992, pp.265-278.

[8] T. Hastings, "A Vector Based Approach to Software Size Measurement and Effort Estimation", *IEEE Transactions on Software Engineering*, vol. 27, No.4 April 2001, pp.337-349.

[9] Kitchenham B.; "Software Development Cost Models", *Software Reliability Handbook*, Elsevier, 1990.

[10] G. Cantoni and P. Donzelli, "Developing and Maintaining Software Measurement Models", *10th European Software Cost Modeling (ESCOM) Conference*, 1999, pp.225-233..

[11] M. Shepperd, C. Schofield and B. Kitchenham, "Effort Estimating Using Analogy", *Proceedings of ICSE-18*, 1996, pp. 170-178.

[12] R. Hugo, "Software Size: The Past and the Future", *10th European Software Cost Modeling (ESCOM) Conference*, 1999, pp.200-209.

[13] N. Ebrahimi, "How to Improve the Calibration of Cost Models", *IEEE Transactions on Software Engineering*, Vol. 25, No. 1, January 1999, pp.136-140.

[14] L. Putnam and W. Myers, " How Solved is the Cost Estimation Problem*?" IEEE Software*, November/ December 1997 pp: 105 – 107.

[15] P. Vesterinen, " A Framework and Process for Effort Estimation ", *10th European Software Cost Modeling (ESCOM) Conference*, 1999, pp. 89-95.

[16] J. Jacquet and A. Abran, "From Software Metrics to Software Measurement Methods: A Process Model", *The Third International Symposium and Forum on Software Engineering Standards, ISESS '97*, Walnut Creek (CA), 1997.

[17] B. Boehm, "Safe and Simple Software Cost Analysis", *IEEE Software* September/October 2000, pp.14-17.

[18] D. Garmus and D. Herron, *Measuring The Software Process: A Practical Guide To Functional Measurements*, Yourdon Press, 1996