A Lifecycle Model Which Incorporates Software Metrics

By Wei Li and Sallie M. Henry

TR 90-51

A Lifecycle Model Which Incorporates Software Metrics

Wei Li
and
Sallie Henry

Computer Science Department
Virginia Tech
Blacksburg, VA 24061
(703) 231-6931
FAX: (703) 231-6075
email: henry@vtodie.cs.vt.edu

Abstract

The traditional waterfall life cycle model of software development provides a systematic method to separate the development process into different stages with explicit communication boundaries between each subsequent stage. But the waterfall model does not provide quantitative measurements for the products of each phase in the software life cycle. The model provides a base to develop methodologies which emphasize on the completeness of the documents, the use of the certain disciplines, and the consistency among documents. On the other hand, it is very hard to use the model to develop methodologies which could provide 1) the quantitative evaluation of the quality of the documents (products) from each phase, 2) feedback information to help manager make management decision, 3) criteria for redesign or recode a system. To ensure the quality of software products, a common basis for more meaningful evaluation leading to better understanding of software quality must be provided.

Software metrics attempt to uncover difficult or complex components of a software system. The hypotheses are 1) that complex components are more difficult to understand, and therefore they are hard to maintain and more prone to error; 2) these errors are more costly to fix due to the ripple effect associated with a high complexity measure. The likelihood of a design error increases as the complexity of the program increases. But the quality indication by metrics may come too late to correct the structural deficiencies that have been completely implemented, possibly at great cost. Therefore, discovery of these complex components at design time can aid the software developer 1) to select which components to redesign, 2) to direct the testing effort, and 3) to give an indication of the maintenance effort required.

The previous studies have shown the encouraging results in 1) measuring the design [12,13,14], 2) predicting code quality from the design measurement [13], 3) predicting the maintainability from the code measurements [15].

Therefore a life cycle model which incorporates software quality metrics is being proposed. The model presented in this paper is measurement driven because the transition criteria of progressing from one phase to another involves software metric measurements. The model provides a possibility of quantitative evaluation of the phase products. It also provides a base for the development of methodologies aimed at improving some software quality factors.

I. Introduction

In order to better control software development, software life cycle process models have been developed. The currently existing models are as follows: the code-and-fix model, the waterfall model (the stage-wise model), the evolutionary development model, the transform model, and the spiral model [1]. However, none of these models provide any mechanism to assess the quality of the phase products. The proper inclusion of measurement phases in software life cycle makes it possible to assess the phase products quantitatively in order to better control the quality of software products.

Software maintenance has been identified as one of the factors which contributes the most to the high cost of a software system throughout its life cycle [2, 3, 4]. Software maintainability is identified as one of the software quality factors [5, 6]. It has been noted that the poor maintainability of software systems is partly because maintenance is not considered at early stages of software life cycle, such as design phase [3, 4]. The proper inclusion of measurement phases early in the software life cycle provides a possibility of assessing maintainability as well as other software quality factors at early stages of software development. This early assessment of maintainability can help produce more maintainable software products.

Software metrics have been developed to measure different aspects of software [7, 8, 9, 10, 11, 12]. Some of them have a high correlation with the maintainability of the software systems [13, 14], while some other metric(s) may give indication of software reliability [12].

Some desired software quality factors include maintainability, reliability, testability, interoperability, portability, usability, reusability, efficiency, integrity, survivability, verificability, expendability, flexibility, and correctness [5, 6]. This paper proposes a software life cycle model which incorporates software metrics. The inclusion of software metrics is intended to provide a base for the development of methodologies which may improve software quality factors [6]. The order of the stages as well as the transition criteria are given. The model is based on the waterfall software life cycle model. However, the concept of measurement phase is not limited to the waterfall model.

Section II briefly introduces some software metrics and related research results. Section III proposes a software life cycle model which incorporates software metrics. Section IV proposes the future research. Section V summarizes the new software life cycle model.

II Software Metrics

Software metrics can be defined as the measurement of some aspects of software. Software metrics can be divided into two categories : process metrics and product metrics. Software process metrics measures the software development process, while software product metrics measures software products such as design documents or source code. The discussion of software metrics in this paper is limited to software product metrics.

Software metrics can be either quantitative, such as the number of lines of source code, or qualitative, such as the functionality of a procedure. Quantitative metrics are more important than qualitative ones in industrial applications of software metrics.

Several quantitative software product metrics have been developed [7, 8, 9, 10, 11, 12]. Some of them are based on the counts of certain tokens of a software system. The tokens could be operator/operand [7], control flows [9], or the bits needed to represent a program [10]. These metrics are called *code metrics* because they are more meaningful when collected from source code. Some other metrics measure the overall hierarchy of a system or the

interconnectivity among the system components [8, 11, 12]. These metrics are called *design metrics* because they can be collected from the system design documents and should remain unchanged or have very little changes throughout the design and the implementation phases [14].

Previous studies have shown that the prediction of maintainability of a software system is possible using software metrics [14, 15]. It has also been shown that the design could be measured [12, 13, 14]. These results make it possible and meaningful of the inclusion of measurement phases after the design and the coding phases. The next section discusses the order of the stages in the life cycle model which incorporates software metrics, as well as the transition criteria of progressing from one phase to another.

III. The Software Life Cycle Model Which Incorporates Software Metrics

Based on the waterfall software development process model, two measurement phases are included as shown in Figure. 1

The Design Measurement phase is to assure the assessment of certain software quality factors such as the maintainability before any effort is put into the implementation of the system. While the System Measurement phase is to assure the assessment of the implementation of the system design. It can be used to check the consistency between the design document and the implementation. It can also be served to provide a guidance for testing and maintenance.

The transition criteria from the Design phase to the Design Measurement phase is the same as the ones of progressing from the Design phase to the Coding phase in the waterfall model. The activities involved in the Design Measurement phase includes :

1. The metric collection for each system components and for the overall system.

2. The generation of the Design Metric Assessment Document for the design.

The phase product is the Design Metric Assessment Document. The transition criteria of progressing from the Design Measurement to the Coding phase are the completion of the metric collection procedure and the existence of the Design Metric Assessment Document for the design. When the industrial standard of software metrics for software quality factors are established, the meeting of the standard should be included as part of the transition criteria.

The transition criteria of progressing from the Coding phase to the System Measurement phase is the same as the ones of progressing from the Coding phase to the Testing phase in the waterfall model. The activities involved in the System Measurement phase includes:

1. The metric collection for each system component and the overall system.

2. The generation of the System Metric Assessment Document.

The phase product is the System Metric Assessment Document. The transition criteria of progressing from the System Measurement phase to the Testing phase are the completion of the metric collection procedure and the existence of the System Metric Assessment Document. The System Metric Assessment Document should be compared with the Design Metric Assessment Document to check any inconsistency between the two. Therefore, the existence of the consistency between the two documents should also be considered as part of the transition criteria. However, how to check for the consistency between the two documents is dependent on what metrics are collected in both documents and what methodologies is used in the development of the software system.

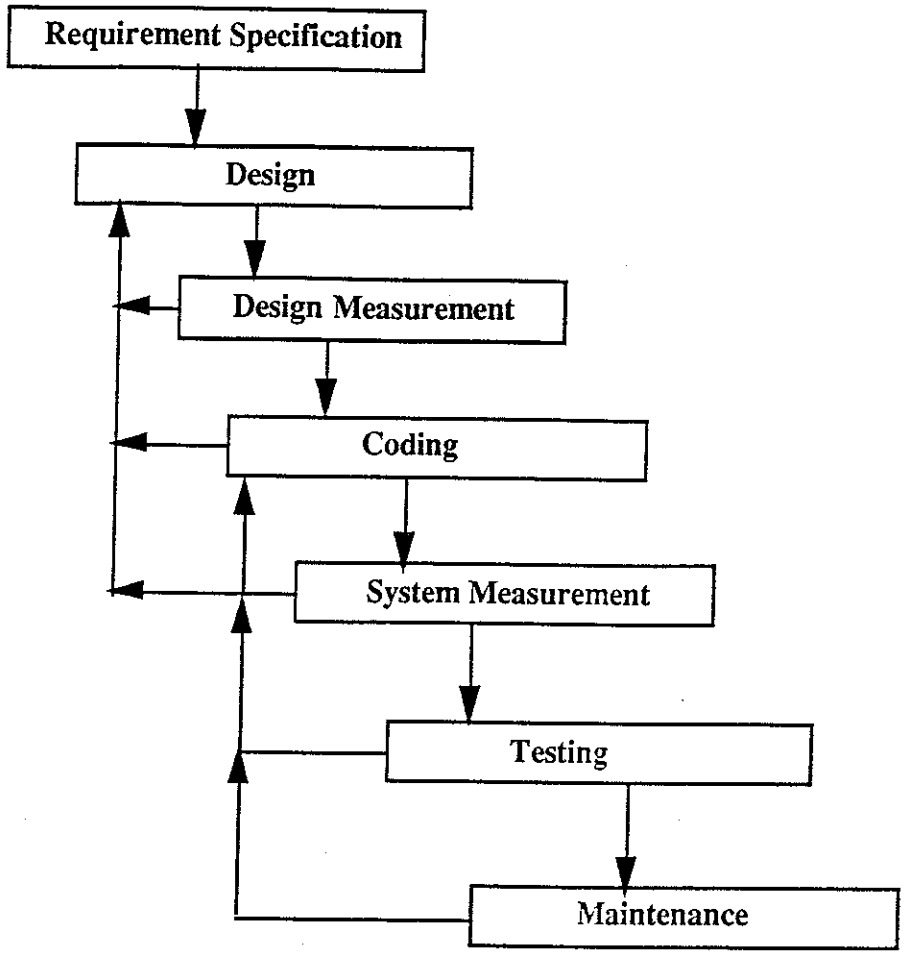

Figure 1. The Measurement-Driven Software Life Cycle Model

If the industrial criteria could be set up for the software quality factors., then the Design Metric Assessment Document can be used to control the quality of the software at early stages in the software life cycle. The System Metric Assessment Document could be used to guide the testing effort and the maintenance.based on the industrial criteria.

The model also leaves the content and the format of the two documents open to methodologies. A methodology could be developed based on the model presented in a particular software development environment to improve any of the software quality factors interested. The next section discusses the future research work which may help the development of methodologies based on the model.

IV. Future Research

As mentioned before, the model presented in section III is based on but not limited to the waterfall life cycle model. Although the model proposes the Design Metric Assessment Document and System Assessment Document as the phase products. What metrics should be used and how they should be organized are left to the methodologies which have different goals to improve the software quality factors.

How the software quality factors [5, 6] could be quantified and what kind of relationship they have with software metrics are still the research topics. The development of new metrics to capture the characteristics of the software quality factors can certainly help the development of methodologies based on the model The further research to reveal the relationship between software quality metrics and the software quality factors are also important and necessary to develop methodologies aimed at improving software quality.

An ongoing research project in Virginia Tech is designed to investigate the relationship of software metrics and the maintainability. It will also look into the means of the development of methodologies to improve the maintainability of the software based on the model presented in section III.

V. Summary

In this paper, a software life cycle model which incorporates software metrics is presented. The model is based on the waterfall software development life cycle model. However the measurement concept is not limited to the waterfall model. The two phases which are included are Design Measurement phase and the System measurement phase. The transition criteria of progressing from one phase to another is given. The activities involved in the two phases are discussed. The phase documents are proposed. Also discussed are some future researches which could improve the model and help the development of methodologies based on the model.

Bibliography

[1] Boehm, B. W., "A Spiral Model of Software Development and Enhancement," ACM Software Engineering Notes, Vol.11, No.4, pp.14-24, reprinted in Computer Vol.21, No.5, 1988, pp.61-72.

[2] Swanson, E. B., "The Dimensions of Maintenance," Proceedings of 2nd International Conference on Software Engineering, October, 1975, pp.125-133.

[3] Munson, John B., "Software Maintainability: A Practical Concern for Life-Cycle Costs," IEEE Computer, November 1981, pp. 103-109.

[4] Schneidewind, Norman F., "The State of Software Maintenance," IEEE Transactions on Software Engineering, Vol. SE-13, No.3, March 1987.

[5] McCALL, James, "The Utility of Software Quality Metrics in Large-Scale Software System Developments," Proceeding of 2nd Software Life Cycle Management Workshop, August, 1978, pp.191-194.

[6] Software Productivity Solutions, Inc, "Software Quality Framework," Technical Report (interim) Volume IV.

[7] Halstead, Maurice H., "Elements of Software Science," New York: Elsevier North-Holland, Inc., 1977.

[8] McClure, Carma L., "A Model for Program Complexity Analysis," Proceeding of 3rd International Conference on Software Engineering, May 1978, pp.149-157.

[9] McCabe, Thomas J., "A Complexity Measure", IEEE Transactions on Software Engineering, Vol. 2, No. 4, December 1976, pp. 308-320.

[10] Bail, William G. and Marvin V. Zelkowitz, "Program Complexity Using Hierarchical Abstract Computers," Computer Language, Vol. 13, No. 3/4, 1988, pp.109-123.

[11] Henry, Sallie, Dennis Kafura, "Software Structure Metrics Based on Information Flow," IEEE Transactions on Software Engineering, Vol. 7, No. 5, September 1981, pp. 510-518.

[12] McCabe, T., Butler, C., "Design Complexity Measurement and Testing," Communication of the ACM, December, 1989, pp. 1415-1424.

[13] Henry, Sallie, Calvin Selig, "Design Metrics Which Predict Source Code Quality," IEEE Software, March 1990.

[14] Rombach, Dieter, "Design Measurement: Some Lessons Learned," IEEE Software, March 1990, pp.17-25.

[15] Wake, Steve and Sallie Henry, "A Model Based on Software Quality Factors Which Predicts Maintainability," Proceedings: Conference on Software Maintenance-1988, pp. 382-387.