Article can be accessed online at http://www.publishingindia.com

Software Safety Metrics for Safety-Critical Computer Systems

Jayasri Kotti*, Seetha Ramaiah P**

Abstract

Software safety is a composite of many factors. The safety aspects of computer-based systems as increasingly important as the use of software escalate because of its convenience and flexibility. There are many tools and metrics for software safety. Many safety metrics exists to help designers to identify potential safety problems. The area of software metrics play an important role in improving the quality of the software process. It is very important to select and use appropriate software safety metrics in software engineering. This paper reviews the method for selecting appropriate software safety metrics based on AHP. AHP theory is a classical multi attribute decision-making method and is more flexible, accurate, and has better sensitivity and consistency. This paper describes a set of software safety metrics for each phase of software life cycle. The following software safety metrics such as "Fault Density", "Test coverage", "Defect density", "Cyclomatic Complexity", etc. are identified to evaluate the safer software development methodology. These metrics are used to evaluate the software safety for many applications.

Keywords: Software Safety, Software Safety Metrics, Safety Critical Systems

1. Introduction

Development of large complex systems in aerospace, defense and travel industries requires constant attention to safety risks. A safety risk is a risk whose cost is injury or the loss of life directly or through a chain of events. The analysis and mitigation of such safety risks significantly increases development time and cost. The task of controlling safety risk is further complicated when developing complex systems due to the amount of planning, assessment and communication required to manage multiple projects. The issue of controlling software safety risk has become a leading area of concern in systems development as many traditionally hardware-centric systems become more heavily reliant on software.

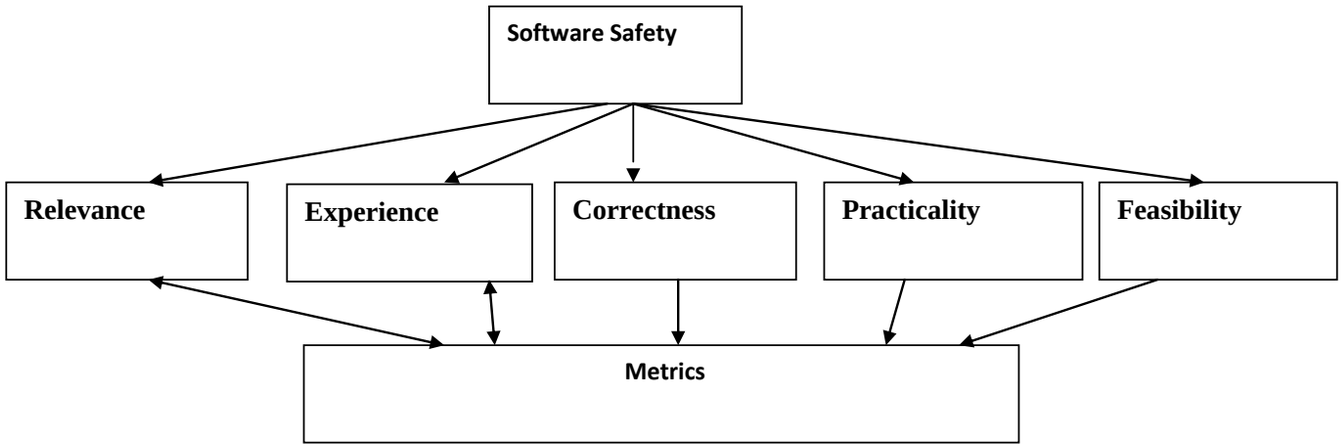
Safety is an emergent system property, and one component cannot make a system safe. Computers and software add an unpredictable element to the system, but there are a number of ways to deal with safety issues. First, it is important to consider safety from the very beginning of system design and a safety team, responsible for system safety issues, should be created. Second, extensive safety analysis should be done to try and come up with as many safety issues as possible. This analysis should better prepare the system designers to deal with safety issues, and it might make it easier to figure out which system functions is safety critical. More effort can be spend on making those identified functions safe and correct. Finally, it is important to use diverse safety mechanisms to increase the chances that one will catch the safety hazard. Although these are not fool proof methods, they are a good starting point for safe system design.

Software safety metrics are used for software safety evaluation and assurance, and they are very important for software safety because they can be used to [6]: 1)

* Assistant Professor, Department of Information Technology, GMRIT-Rajam, Srikakulam District, Andhra Pradesh, India. E-mail: jayasrikotti@gmail.com

** Professor, Dept of CS & SE, Andhra University, Visakhapatnam, Andhra Pradesh, India. E-mail: psrama@gmail.com

Figure 1: Software Safety Metrics According to Selecting Criteria



provide quantitative indicator for safety management 2) be a trade-off among cost schedule, and safety 3) monitor testing process and forecast testing schedule 4) evaluate and validate safety 5) evaluate software engineering technology from the viewpoint of safety 6) interpret safety behavior (recur to software safety models). However, how to select appropriate metrics according to the character of different software and development phase is still a troublesome problem in engineering. Selecting too many metrics randomly will cost more resource and workloads. Therefore, the method used for selecting metrics scientifically is very important for software safety testing and evaluation.

The application of the disciplines of system safety engineering techniques throughout the software life cycle to ensure that the software takes positive measures to enhance system safety and that errors that could reduce system safety have been eliminated or controlled to an acceptable level of risk.

1. Terminology: In this paper, we define the terms safe, safety and Software metric according to definitions found in the literature. Safe is having acceptable risk of the occurrence of a hazard [8].Safety is the freedom from those conditions that can cause death, Injury, occupational illness, or damage to or loss of equipment or property. Software metric is a measurement derived from a software product, process, or resource. Its purpose is to provide a quantitative assessment of the extent to which the product, process, or resource possesses certain attributes.
2. Software safety metrics method based on AHP: The analytic hierarchy process (AHP) is a structured technique for organizing and analyzing complex decisions. The AHP is a decision support tool which can be used to solve complex decision problems. It uses a multi-level hierarchical structure of objectives, criteria, sub-criteria, and alternatives. This paper aims to solve the problem that how to select software safety metrics for each phase of software development life cycle. To achieve this objective, we proposed various software safety metrics based on AHP theory we have also identified selection criteria for safety. The top metrics are strengthening the safety and feasibility of the system. This paper can be used to select the appropriate metrics in each phase of life cycle.

3. Software Safety Metrics for Selection

In this paper we identified the important safety metrics of the different phases of Software life-cycle like requirements analysis, design, implementation and testing.

3.1. Selecting Criteria

Selecting criteria are several attributes by means of which the metrics can be compared. We adopt the following five criteria according to [6] and [9]:

a. **Relevance (to reliability):** This criterion reflects the relationship between metrics and software reliability.

b. **Experience:** This criterion reflects the degree to which this metric has been used and recognized.

c. **Correctness:** This criterion includes 1) Objectivity. The input and results of this metric can't be easily influenced; 2) Justness. The metric is not partial to any specific result; 3) Precision.

d. **Practicality:** This metric should be concerned and required in development.

e. **Feasibility:** This criterion means that 1) the formula of this metric should be understood easily, and supported by tools; 2) data collection should be easily;3) the results of this metric can be evaluated and confirmed conveniently.

4. The Selection Results

Refer to [4][6][9], the top metrics are identified according to every selecting criterion in each phase based on the method.

4.1. Software Safety Metrics in the Requirements Phase

Software Safety Requirements Analysis (SSRA) shall be performed and documented. The system-level PHA and the system conceptual design shall be used as input to the SSRA. The three top metrics in the requirement phase[6] are "Fault Density", "Test Coverage", and "Error Indices". The three metrics are most appropriate in the requirements phase. Errors in the requirement may cause fatal failure and the formula of "Fault density" is easy and convenient. Therefore "Fault Density" has significant advantages on the criteria "Relevance", "Feasibility", and "Practicality". "Test Coverage" here is "Requirement Coverage" which is important to the requirement quality. Moreover this metric has been used frequently, which is only a little lower on "Relevance" than "Fault Density".

Fault Density: Fault (or) Defect Density is the number of confirmed defects detected in software/component during a defined period of development/operation divided by the size of the software/component. By using this metric we can compare the relative number of defects in various software components so that high-risk components can be identified and resources focused towards them.

Fault density = Number of defects / Total Size of the system

Test coverage: Test coverage metric is expressed in terms of a ratio of the metric items executed or evaluated at least once to the total number of metric items. This is usually expressed as a percentage.

Test Coverage = items executed at least once / total number of items

4.2. Software Safety Metrics in the Design Phase

The two top metrics in the design phase [6] are "Defect Density", "Cyclo-matic Complexity". "Defect Density" is often used in software engineering. Cyclomatic complexity is a software metric . It is used to indicate the complexity of a program. It directly measures the number of linearly independent paths through a program's source code.

The measurement of cyclomatic complexity by McCabe [7] was designed to indicate a program's testability and understandability (maintainability). It is the classical graph theory cyclomatic number, indicating the number of regions in a graph. As applied to software, it is the number of linearly independent paths that comprise the program. The general formula to compute cyclomatic complexity is:

$M = V(G) = e - n + 2p$

Where V(G) = Cyclomatic number of G

e = Number of edges

n = Number of nodes

p = Number of unconnected parts of the graph.

4.3. Software Safety Metrics in the Implementation Phase

The four top metrics in the implementation phase [6] are "Defect Density", "Cyclomatic Complexity", "Fault Density" and "Test Coverage". The top-three metrics are the same with the design phase, thus we discuss the fourth top metric. "Test Coverage" here is "Code Test Coverage".

White-box testing often needs many kinds of code coverage information, such as path coverage, statement coverage, etc, and moreover, a lot of auto testing tools can be used to get the code coverage easily. “Test Coverage” is very practical and feasible in the implementation phase.

4.4. Software Safety Metrics in the Testing Phase

The four top metrics in the testing phase are “Failure rate”, “Defect density”, “MTTF” and “Test Coverage”. Firstly, we notice “Failure Rate” and “MTTF”. The two metrics are often used in the reliability testing of safety critical software as the very important quantitative reliability indicator. Therefore they are very good on criteria “Relevance”, “Experience”, “Feasibility” and “Practicality”.

5. Conclusion

This paper aims to solve the problem that how to choose software safety metrics for the software safety engineering in each development phase, which provides safety. To achieve this objective, Selecting criteria for safety metrics were explained in Section 3.1 and the proposed safety metrics are mentioned in section 4.In every phase of the life cycle, the top metrics are strengthening the safety and feasibility. This paper can be used to describe the various safety metrics in each phase correctly, stably and systemically.

References

1. Li, H., Lu, M. & Li, Q. (2006). Software Reliability Metrics Selecting Method Based on Analytic Hierarchy Process. Quality Software, 2006. QSIC 2006, Sixth International Conference, Beijing.
2. ISO/IEC 9126-1: Information Technology-Software Quality Characteristics and Metrics-Part I: Quality Model. Retrieved from http://en.wikipedia.org/wiki/ISO/IEC_9126, (accessed on August 28, 2013)
3. IEEE Std. 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software. (accessed on August 28, 2013)
4. IEEE std. 982.2-1988, IEEE Guided for the Use of IEEE standard Dictionary of Measures to produce Reliable Software. (accessed on August 28, 2013)
5. IEEE100, The Authoritative Dictionary of IEEE Standard Terms, IEEE Press, 2000. (accessed on August 28, 2013)
6. Cai, K. Y. (1995). *Elements of Software Reliability Engineering*. Beijing: Tsinghua University Press.
7. McCabe, T. J. (1976). A Complexity Measure. IEEE Transactions onSoftware Engineering.
8. Seetharamaiah, P. & Swarup, M. B. (2008). Towards a Methodology for Building Safe Software Based Systems. Proceedings of the CONQUEST 2008, 11th International Conference on Quality Engineering in Software Technology, Potsdam.
9. Research on Military Software Reliability Metrics. (2004). National Defense Science Report.