Software & Security Assurance  
Metrics and methods for software certification

Fariborz Entezami  
MSc in Networking and Information Security  
WMN Research Group, Faculty of Computing, Information & Mathematics  
Kingston University, London, England  
K0945507@kingston.ac.uk

Abstract  
Our life has been affected by information technology. Software is used in most of our life even in some critical parts such as air traffic control or power plane to control very sensitive functions. Any misfunctionality or bugs in unreliable software could hide and it may cause problem and make cost even takes lives. Connectivity and complexity are common in software production today and these factors make assuring and certifying software big challenge that needs more money and time. In this paper, security and software certification and assurance have been considered and metrics which are used in this subject will cover and some techniques that reduce the effort of software evaluation will consider more.

Keywords: Software Certification, Reliability, Software, Security, Software metrics

Introduction  
Software plays a major role in today’s human life at work or even at home. Users need the software which are reliable and accurate. However the hidden software’s complexity made it capable to have undetected errors and bugs. And it is hard to evaluate quality of software regarding reliability and accurately. Certify software means that the software fit for users’ purpose. Testing software exhaustively is not possible and there are some examples which errors in software cost money and human lives. For example, a wrong conversion of a 64 bit value made a $370 million cost in the Ariane 5 launcher. On that event the launcher veered off its flight path, broke up and exploded. So even when best practice has been applied, software still have bugs. Also there are many technique are to find errors and bugs but they need time and money. (1)  
Software certification is important in all level of software from drivers and compiler up to applications and Enterprise Resource Planning (ERP) software. For example a compiler should translate the source code into the machine language as it is instructed, but we all know that some bugs in some compilers made a correct program into an incorrect executable software. (2)  
Software certification shows the safety and reliability of software systems in a way which a third party as an independent authority could check the software with minimal trust in tools and techniques which are used in certification process. Independent assessor could assess demonstrated properties of software by the information that is provided by explicit software certificates. These certificates based on existing software assurance, validation and verification techniques. (3)

Another important matter in software certification is security. As complexity and connectivity are increasing day by day in software systems, products and services needs more reliability and trust in software systems. In a systematic approach to measure security in systems needs obtain evidence of security level performance in system services and products. Making decision on concerning security based on sufficient evidence which are credible and reliable is more easy than other cases. (4)  
In the following sections, evaluation assurance levels and definition on security certification will consider more and then software metrics and security metrics are studied. In the final parts some technique which are used to decrease efforts that is needs to assure software will be demonstrate.

Evaluation Assurance Level (EAL)  
Evaluation Assurance Level (EAL) is an IT industry level which is introduced in 1999 as an international Standard. EAL is a numerical grade between 1 to 7. Increasing assurance levels means more assurance requirements is needed to meet achieved common criteria certification. Higher levels of EAL show higher confidence in security and reliability implementation. Although EAL does not measure the reliability or security of the system itself, it shows at what level the system was tested. Higher assurance level need more document, Money and time.  
Assurance levels  
EAL 1: Functionality Tested  
EAL 2: Structurally Tested  
EAL 3: Methodically Tested and Checked  
EAL 4: Methodically Designed, Tested and Reviewed
Software Certification

Software certification usually contains executable software which run by computer and a formal proof which check by machine who certify the software is free of bugs. This proof is with respect of dependability. It can be very hard to achieve because running software is depend on operating system, compiler, hardware, user and physical environment. Consider all these parameter and then certify the software is a real challenge. Lack of metrics in software dependability makes process of certifying software hard. Dependability also involves other system metrics such as reliability, safety, availability and security. Measuring most of these metrics is very hard and need more research. Software also depends on other underlying system software such as OS kernel, Device Driver, hypervisor, garbage collector and compiler. In the following section, software metrics relating to reliability and security are studied and some techniques about software assurance are considered. (5)

Software Metrics

Software metrics could be treated as the part of software development or software assurance. Metrics could be measure by developers or assessors to measure that the software meets the purpose aim or not. The quality of software could be measure by software metrics. As much as the metrics have been classified, measure accurately and validate correctly could be directed to show software quality precisely. (6)

A. Definitions

Regarding to the first part of standard ISO/IEC 9126-1, there are six factors to present software quality models. The factors are Functionality, reliability, usability, efficiency, maintainability and portability. In this standard, each characteristic has some sub-characteristics which define more parameters. Each sub-characteristic is divided into attributes which can be verified or measured in software assurance. (7-10)

Functionality assurance is a form of continues testing to be sure that system working properly. This characteristic has Suitability, Accuracy, Interoperability, Security and functionality compliance as its sub-characteristics. (8)

Reliability is a main key in software certificate and software assurance. In IEEE 610.12-1990, reliability defines as “The ability of a system or component to perform its required functions under stated conditions for a specified period of time.” And to define Software Reliability Management by IEEE, It describes in IEEE 982.1-1998 as “The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources, schedule and performance.” By this definition the software reliability is effected by three activities. Error prevention, Fault detection and removal and measure the software’s metric to maximize reliability are three activities which affect reliability. This characteristic has Maturity, Fault Tolerance, Recoverability and Reliability compliance as its sub-characteristics. Reliability affects into three different criterias. Accuracy, consistency and completeness are three areas which reliability affects them. (8-9)

Usability is one of the parameters in software certification. In International Organization for Standardization (ISO) defines usability as “The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.” This parameter could measure in product operation and are involved into two criterias, communicativeness and accessibility are two areas which usability could be involved. This characteristic has Understandability, Learnability, Operability, Attractiveness and usability compliance as its sub-characteristics. (7-8)

Efficiency is a parameter in software quality models. In general efficiency is a ratio between output of a system and input of that system. It shows how many units of resources are used to run a specific task. This parameter is involved device efficiency and accessibility areas. The factors usability, reliability and efficiency are used in product operation. This characteristic has Time Behavior, Resource Utilization and Efficiency Compliance as its sub-characteristic. (7-8)

In product revision four different metrics should consider, reusability, maintainability, portability and testability. Reusability is a term which use when a part of source code is able to use as a function in other part of software. Reusability is involved accuracy, structuredness, conciseness, device independence, legability, self- descriptiveness and traceability. (7)

Maintainability in software engineering is a characteristic which shows level of modification of a software product after solve all reported faults, to improve performance or other attributes. This characteristic has Analyzability, Changeability, Stability, Testability and maintainability Compliance as its sub-characteristics. (8-10)

Portability is one of the concepts in high level programming. In the software code programming, Portability is a feature that some parts of the code reuse in a new environment instead of creating the new code. This characteristic has Adaptability, Installability, Co-Existence, Replaceability and Portability Compliance is its sub-characteristics. (8-10)

B. Reliability as a Quality Attribute

There are many different models and techniques to assure software quality, but most of them based on
reliability and it is a main key in software certification. As we saw in previous section, reliability is one of the six characteristics of software quality which has been defined in ISO/IEC 9126-1 standard. In IEEE standard 982.2-1998 it is defined “A software reliability management program requires the establishment of a balanced set of user quality objectives, and identification of intermediate quality objectives that will assist in achieving the user quality objectives.” As it is clear that reliability will cause quality, it can be concluded that for high quality in software, we need increase reliability.\(^{(9, 11)}\)

Developing software with high reliability depends on quality attributes at each level of building the software by emphasising on error prevention. The application of quality attributes should be measure on each life cycle phase especially in the early period of software life cycle. Metrics are used on each phase to measure the quality attributes. Reliability should be a main focus on each development phase which is measure by metrics and can evaluate reliability status on each phase. IEEE Standard 982.2-1988 indicate the relationship of reliability in different life cycle phases in a diagram that it is shown in Fig. 2.\(^{(9, 12)}\)

**Fig. 2. Relationship of reliability in different software life cycle**\(^{(12)}\)

Reliability is focusing more on error prevention and for this reason, the quality attributes should identify and measure at different life cycle phases and it should focus on requirement, implementation and test phases.\(^{(9)}\)

- **Reliability Software metrics**

  Software Assurance Technology Centre (SATC) is a department in National Aeronautics and Space Administration (NASA) who used software metrics to help developers to improve reliability by identifying the areas of software which can potentially cause error. SATC also examines the test plan for stratifying all requirements have been covered without excessive and expensive testing. In the following paragraphs the metrics used by SATC will be discovered more for identifying errors before software is released. Error prevention technique and software metrics which can be impact reliability will be addressed in three life cycles: requirements, coding and testing.\(^{(9)}\)

  - **Reliability metrics in requirements life cycle**

    In this stage, it is critical that all requirements be written carefully with all detail and avoids any misunderstanding between client and developer. The requirements should be structured, completed and easy to understand and to apply. Three different primary formats have been provided for requirement specification structure by IEEE, Department of Defense (DOD) and NASA. Using such as these formats make developer to be ensured that critical information is not omitted. Requirements’ specification should not contain phrases which may have negative impact on the design or any cause to architecture such as ‘to be deleted’ (TBD) or ‘to be added’ (TBA). It is important in this stage, the requirement has been written in a common language (English) for both client and developer and avoiding of any ambiguous term in requirements or any terms which could constructed as an optional requirement. Ambiguous terms could be any contains which give developer to decide whether or not to implement that requirement.\(^{(9)}\)

    The SATC has developed a tool to prepare requirements documents. Automated Requirements Measurement (ARM) was developed to scan requirements specification. This software searches the whole documents to find specific words and phrases which indicated by SATC’s studies to be an indicator of the document’s quality as a specification of requirements. Seven measures were developed by ARM as:

    - **Line of Text** – it shows the size and it is the physical lines of text.
    - **Imperatives** – words or phrases which commands to do something and is used as a base requirements count.
    - **Continuances** – phases that follow an imperative and define the specification of requirements at a lower level and is used for supplemental requirement count.
    - **Directives** – references noticed to figures, tables or notes.
    - **Weak Phrase** – Phrases which contain ambiguity and cause uncertainty and make an opportunity of multiple interpretation.
    - **Incomplete** – Clauses contain TBD (To Be Determined) or TBS (To Be Supplied) phrases.
    - **Options** – statements that seem to give the developer not a certain specification and it can be ambiguous.\(^{(9)}\)

    It is clear that this tool does not attempt to assess the requirements specification which is true or not. It assesses the vocabulary which is used in statement but it also capable to evaluate the structure of the
requirement documents. ARM also evaluates the structure of the document. Evaluation of structure is done by distinguishing the number of requirements at each level and it uses hierarchical numbering structure. Lack of structure affects reliability by increasing the difficulty in changing the code or makes improper design.\(^{(13)}\)

- **Reliability metrics in Design and Code**

  Although developer’s software has tools to design program and code, these tools do not provide an automated evaluation or quality metrics as output. The SATC analyses the structure and architecture to identify any possible error prone modules. This analysis is based on complexity, size and modularity. It is generally obvious that more complex code makes understanding more difficult and it increase probability of finding more error prone modules. Then complexity is a direct impact to maintainability and in the end into software quality.\(^{(9)}\)

  There are many different ways to measure complexity. SATC use logical (Cyclomatic) complexity which is calculated as the number of linearly independent test paths. Size is one of the traditional and most common metrics of software quality. Size of modules could show one of the software metrics. Size can be measure by total of line code which comment and blank lines have been deducted. The executable commands are defined by language dependent delimiter. SATC has found relation between reliability and the combination of size and complexity. As the software with high complexity and low size shows the reliability is low because the code should be very complex and modifying should be difficult and also maintainability should be low.\(^{(9)}\)

  The above metrics are also work in object oriented code. In addition, some metrics have been defined for object oriented code quality analysis by SATC. Weighted Methods per Class (WMC), Response for a Class (RFC), Coupling Between Objects (CBO), Depth in Tree (DIT) and Number of Children (NOC) are the metrics for object oriented quality analysis. SATC has developed a guideline to describe how these metrics have been calculated. For example WMC shows how much time and effort is required to develop and maintain the class. The lower the WMC shows the less testing and maintaining is required and reverse. The complexity should not exceed 5, therefore, WMC should be between 100 and 200.\(^{(9)}\)

- **Reliability metrics in Testing**

  To consider metrics in testing phase, two approaches should be considered to evaluate reliability. The first approach is the consideration of test plan based on functionality specified in requirements phases. The second approach that is close to reliability is the evaluation of the number of errors in the code and rate of finding or fixing them. The SATC has developed a model to simulate the finding errors mechanism and shows how to predict remaining errors in project and when they will be revealed.\(^{(9)}\)

  In this stage, the software should be tested to show it does such as all specific functionality which are defined in requirements phases. The test plan is designed to contain some test cases. Each test case is based on each statement which is defined in requirements phases. Some times each statement should test more than one time depends on different system status in varying scenarios or in different ways. As always time and budget are in other side of balance, designing test plan based on multiplex statements is a challenge to meet mention resources. SATC developed a model which implements a nonlinear approximation based on an error trending perspective to predicate errors found to date. The other technique is using probability to determine error date curve.\(^{(9)}\)

C. **Security metrics**

The security metrics term refer to standard term when the subject is about security level, security performance, security indicators or security strength. In practice, the terms security strength, security indicators or security measurement are often in place of security metrics. It should be notice that measurement results refers to a specific data on a specific time, while metric shows data driven from measured data to make easy taking a decision.\(^{(4)}\)

- **Security Objectives**

  The most common security objectives are Confidentiality, Integrity and Availability (CIA). And some times it referred to CIA model. Confidentiality refers to the situation which objectives or information are accessible by authorized body/bodies only. Integrity shows the accuracy and completely of information through the system process. Availability is an objective which refers to a situation that information and associated assets are available for authorized party/patties when they require them. Although CIA model is a useful guideline for security objectives, authentication and non-repudiation are two another factors which should be considered in security objectives. Authentication is a mechanism to identify the user and managing to control right and access based on authorization policy. Non-repudiation mechanism is a process that prevent users to deny what they did before regarding the authorization policy.\(^{(4,14)}\)

- **Security Metrics Objective Segments (SMOS) model**

  Yee believes that with composing multi-dimensional security metrics, it is possible with the new multifaceted metrics, we emphasize into different relevant metrics objectives. They are called metric objective segments. Measuring the security level and/or performance of the System Under Investigation (SUI) is very challenging and it is a complex technical problem. Fig 2 shows SMOS model, the model shows these security metrics objective segments:
Level 0 (Target): Security level of SUI
Level 1 (Main Viewpoints to Target): Three SUI security segments: (i) Security-enforcing mechanism or control, (ii) Security quality of the system, (iii) secure system lifecycle, project and business management.
Level 2 (Fundamental Measurement Objectives): Three fundamental objectives of security measurement: (i) Correctness, (ii) Effectiveness and (iii) Efficiency
Level 3 (Decomposition): the Basic Measurable Components (BMCs), the main point of collecting the security metrics such as: (i) requirement decomposition, (ii) design decomposition.
Level 4 (More Detailed Metrics Characteristics): this level presents more detailed metrics characteristics depending on the feasibility and use [4].

Fig. 3. Security Metrics Objective Segments (SMOS) model [4]

Risk Analysis
Risk analysis is another approach to determine the techniques which be applied in development of the software to assure it is fit for purpose. Three parameters are base of the risk analysis. Criticality of usage, Complexity of processing and Complexity of control are values for the risk analysis. Each parameter can take one of four values.
The values are:
Complexity of processing: very simple, simple, moderate and complex.
Complexity of control: very simple, simple, moderate and complex.
Any other consideration depends on any legal obligation that may have to be met. (1)

MODULAR REASONING
Modular reasoning is the process to understand each module of the system at a time. A language is called modular reasoning supported if the actions of module X can be understand solely based on the code in X with the specification of any modules referred to by X. For example, in Java language, a module is a single compilation unit or a file define a single top level class or interface. The specification of the module is the action of objects. Java code exclusively is one from specification. In more complex and expensive language such as Eiffel or Java annotated with JML, a specification can be given with previous or post status. Software assurance is closely related to modular reasoning, because the code of the module has a precise relation between action and specification of the method. Modular reasoning can also permit separation compilation and modular implementation. (15)

SEPARATION LOGIC
Separation logic is one of the rules of the local reasoning as the specifications of a component of a software mention only to code of that part and the resources which use by that portion of the software and it can not be evaluated with entire global system status. It helps developers and other third parties who are going to assure the software or certify it, to evaluate each part of the software separately. Separation logic help reasoning to clear any manipulation of data structures’ pointer including hiding information and monitor transferring ownership to avoid using semantic frames and also help for virtual separation for the reason of modular reasoning through running multi modules. (16)

CERTIFIED COMPILATION
To certify software in a high level language such as C, C++ or Java it is necessary to certify the source code of the programming language and also it needs to certify the compiler which compiles that source code to make an assembly executable program. For example CompCert is a certified compiler for language C programming who certifies this compiler is trustworthy and any certified source code can be certified assembly software by compiling with this certified compiler. (5)

LIGHTWEIGHT FORMAL METHOD
Certifying a large-scale software needs a large amount of money and time. But with Lightweight formal methods, it is possible to verify a large-scale software with a not heavyweight program and reduce usage of money and time. Most developers use modular components which are used in some classes and levels. In high level programming, components are running with restricted structures on well-defined interfaces, it helps to use lightweight formal methods to certified components detail based on certified compiler, and the whole software could be certified. Lightweight formal consider static-analyzer, error messages and decision procedure to analyze programming process. In these methods, programmer can generate some proof witness in the software and link them to certified low-level
kernel and other components to create end-to-end certificate. Lightweight formal methods dramatically reduce cost of certification.\(^{(5)}\)

**CONCLUSION**

Today, software is mixed with our life and it plays a major role in some critical activities. Accuracy and reliability are keys to make us secure to use software in critical situations. Software certification is a guarantee that software will be working as it should be. Connectivity and complexity are two parameters which make today’s software more complicated. Different programming language and development tools with complexity and complicating make software assurance is a real challenge that needs more money and time to certify big software.

EAL is an industrial level which shows in which level the software is tested and how we could trust it. Some metrics have been defined in reliability and security aspects which could help to define parameters in software assurance and certification.

Risk analysis is another method to help developers or assessors to evaluate the software that is fit to purpose and is as expected. Modular reasoning and separation logic are other methods to help third parties to assure the software with less effort. In these methods, the developers use each module as it is in code. And evaluation of each module is not related to whole project and software evaluation will need less effort.

Certified compilation is a method that could decrease the resources which need for certification of big projects. By using certified compiler, for certifying the software it needs only to certify the source code. Assuring the source code is much easier than whole software and needs less money and time. Lightweight formal is another method to decrease the efforts need for certifying a big software. It uses some technique based on components restriction on development tools to evaluate very big software.

**References**


