Evaluating the Relevance of Prevailing Software Metrics to Address Issue of Security Implementation in SDLC

C. Banerjee
Research Scholar,
Jagannath University, Jaipur, India

Arpita Banerjee
Assistant Professor,
St. Xavier's College, Jaipur, India

P. D. Murarka
Professor,
Arya College of Engg. & Tech., Jaipur

Abstract— **Security implementation in software during its early stages of development ensures fault free software. The security requirements are qualitative in nature therefore they should be converted into quantitative measure with the help of metrics. As per the statistics available, the issues of security are not very well addressed by already present traditional software metrics. Security becoming a very important requirement of most software systems, the existing software metrics or some newly developed security metrics have to be tailored considering the security aspect during the software engineering process. This research paper explores the role of existing software security and evaluates the same from security implementation point of view during software development process. The paper further shows the various finding and observation and recommends some solutions for comprehensive development of software security metrics.**

Keywords-software metrics; security metrics; software security; security in software; security implementation in SDLC

I. Introduction

Software, in this age of information, has an integral role to play in the world's economy and its success lies in its secured use. Security of software is the primary and major concern for industry, academic institution and research community. Security implementation in software enforces limits on the damages caused due to various attack triggered fault and provides quick recovery mechanism for the system. It ensures uninterrupted operation of the system under most adverse condition [1, 2].

Researchers suggest that security of software should not be considered as just another aspect of software quality. They advocated that the security aspect should be dealt separately and not merely as qualities feature due to the fact that implementation of good security in software cannot be considered as a byproduct of good quality [3]. In many organizations, less importance is given to implementation of security in software due to the pre assumption that it may delay the project and adds to time and cost factor [1]. As such, as an alternative, the organizations adopt the process of penetration testing which works on the concept of *build then break.* According to researchers, results have shown that penetration testing can only act as a security aid and does not reveal all security issues and problems in software [3].

Security requirement of software are considered as a set of nonfunctional attributes. These qualitative measure needs to be converted to quantitative measures or metrics which can then be used as indicators and estimators and helps the security analyst in measurement of degree of security implementation in software [4, 5]. From the beginning of software development process, these metrics can be useful in analysis of flaw in a software system suggesting its prevention or timely detection and correction [6].

Software Metrics are used by software development team to access project status, to gain insight into the efficiency of software process, to track potential risk, early problem area detection, adjust work flow or tasks, etc [7]. The security implementation during software development process are not properly addressed and since security is a very important requirement of a software systems, hence, the existing software metrics or some newly developed security metrics have to be extended / developed for comprehensive implementation of security during the software engineering process [8, 9].

Although researchers have done remarkable work in the field of software security metrics, still it lacks support and validation from the software organization and a major portion of work needs to be carried out for proper implement of security during the early development stages of software. In extension to the work carried out earlier, in this paper we intend to evaluate the relevance of prevailing metrics to address the issues of security implementation in SDLC. Rest of the paper is organized as follows: in section II we discuss about 'Software Metrics and Security Metrics', and in section III, 'Study of Existing Software Metrics' are given, section IV, throws light on 'Evaluation of Existing Metrics: Security Perspective' and Section V focuses on 'Findings and Observations' with 'Conclusion and Future Work' given in section VI.

II. SOFWTARE METRICS AND SECURITY METRICS

Measure is a quantitative indicator of the size of some process or process attribute. Measure of software engineering process can be characterized into direct measures and indirect measures. Direct measure includes lines of codes (LOC), size of memory, defects for each reporting time period, and execution speed. Indirect measure includes quality attributes like functionality, complexity, efficiency, reliability, maintainability, security, etc. Measurement is the process to obtain a measure. Metrics can be defined as a quantitative measure of the scale to which a system, component or process holds a known characteristic [7].

Software metrics can be defined as the measurement of software product and the process through which the software is developed. For modeling the software development process the measurement related to software process and products are developed by software development team. These measurements in the form of metrics are used by the software development team to estimate and predict various tangible and intangible factors like cost, schedule productivity, quality etc. The information thus collected from these metrics is used by the software development team to manage and control the development process which in turn results in improved software products [7, 8].

Security is characterized as a non-functional requirement and is a quality attribute which cannot be measured directly, hence indirect measures should be identified related with the security aspects and further should be converted into quantifiable measures using security metrics. Security metrics is defined as quantifiable measures which show how much security a product or process simply possess. Security metrics is normally built from the low level physical measures and at high level they can be considered as quantifiable measurements of some aspect of a system [7, 8].

III. STUDY OF EXISTING SOFTWARE METRICS

The growing use of applications mostly web based and distributed in nature has given rise to the concept of insecurity due to the presence of wide variety of vulnerabilities which can be exploited by the attacker to harm the system. Hence, building security during the software development process has become essential which can result in less threat and exploit posed to the system. The level of security incorporated into the software while developing it should also be measured to further improve or control it [8, 9]. It clearly advocates the importance of using security metrics in development process of software. The following section discusses the applicability of some established traditional metrics in measuring the security level of software.

Size oriented metrics are derived by normalizing any direct measure with the LOC. It considers the size of the project that has been used. For the elementary data contained for each project, a set of simple size-oriented metrics can be developed as follows i.e., Errors per KLOC (Kilo lines of code), Defect per KLOC, Cost per KLOC, Errors per person-month, Documentation per KLOC, LOC per person-month, Manpower hours per KLOC, Cost per page of documentation, and Functional Oriented Metrics [7].

Function oriented metrics proposed by Albrecht uses a measure called FP (Function Point) as a normalization value which is delivered by the application. The data is collected and with each count a complexity value is associated. To compute function point (FP), the relationship $FP = count\ total \times [0.65 + 0.01 \times \Sigma(F_i)]$ is used where count total is the addition of all entries of FP obtained. After the calculation of function point (FP) they are used similar to LOC to obtain the following metrics i.e., Errors per FP, Defects per FP, Cost per FP, Manpower hours per FP, Documentation per FP, and FP per person month. Since the basic function points are not sufficient for many organizations certain advancement were being proposed in the basic function point and it is known as 'Extended Function Point Metrics'. In this an extension of function point known as 'feature point' is considered [7].

DeMarco proposed Bang metrics which is an implementation independent indication of system size. It is evaluated using a set of primitives, developing counts for the following i.e., Functional primitives (FuP), Data elements (DE), Objects (OB), Relationships (RE), States (ST), Transitions (TR), Modified manual function primitives (FuPM), Input data elements (DEI), Output data elements (DEO), Retained data element (DER), Data tokens (TCi), and Relationship connections (REi) [7].

In Architectural design metrics, Card and Glass proposed three software design complexity metrics viz., structural complexity, data complexity, and system complexity. The structural complexity provides an indication of the complexity of module i and is defined as $S(i) = f^2\ out^{(i)}$. For a module i, data complexity is an indicator of the complexity in its internal interface and is defined as $D(i) = v(i)/[f_{out}^{(i)} + 1]$ where v(i) is the number of input variables passed to and output variables passed from module i. The system complexity is indication of sum of structural and data complexity and defined as $C(i) = S(i) + D(i)$ [7].

Component level design metrics comprises those measures which are related with module coupling, cohesion, and complexity. It focuses on the internal characteristics of software component. These measures are indicators of quality of a component level design. Bieman and Ott proposed Cohesion metrics which is an indication of five concepts and measure i.e., Data slice, Data tokens, Glue tokens, Superglue tokens, and Stickiness. These metrics were developed for string functional cohesion (SFC) defined as *SFC(i) = SG [S(i) / (tokens(i))]*, weak functional cohesion, and adhesiveness. Dhama has proposed a coupling metrics for module coupling that encompasses coupling of data and control flow, along with global coupling, and environmental coupling. The metrics is an indicator of module connectedness with that of other modules, global data, and with the outside environment.

McCabe proposed complexity metrics (cyclomatic complexity) which provides a quantitative measure of testing difficult and reliability. It also provides a quantitative indication of maximum module size. Experimental studies have also shown distinct relationship between the proposed metric and the no. of errors in source code and the time required to find and correct it. Henry and Kafura have also proposed information flow metric as a complexity metrics which provides the count of information related to enter and exit the procedure and accordingly the complexity is calculated [7].

In Interface design metrics, Sears has proposed layout appropriateness (LA) according to which, for a specific layout, cost can assigned to each sequence of actions and defined as *cost = ∑ [frequency of transition (k) x cost of transition (k)]* where 'k' is the specific transition from one layout entity to the next as a specific task is accomplished [7]. LA is defined as *LA = 100 x [(cost of LA – optimal layout) / (cost of proposed layout)]*. In respect to Source code Metrics, Halstead has identified the programs as a sequential token consisting of operators and operands. On the basis of these tokens the length of the program is calculated. He has made the use of primitive data as well as derived data or new data for his calculation.

Quality oriented metrics proposed by McCall and his colleagues is based on defect removal efficiency (DRE) as the normalization value. It is computed by the equation *DRE = E / (E + D)* where 'E' is the no of errors found beforehand the delivery of the software to the end-user and 'D' is the no of defects which are found after the software was delivered. Reliability metrics proposed by S. Henry and W. Li is based around the system failure probability. The reliability metrics are (i) *MTBF=MTTF+MTTR* where MTTF represents for Mean time to failure, MTTR represents Mean time to repair, and MTBF represents Mean time between failure, (ii) Probability of failure on demand (POFOD), (iii) Rate of failure occurrence (ROCOF), and (iv) *AVAIL=MTTF/(MTTF+MTTR)*100%*. This metrics can be used in the later phases of software development life cycle as it is based on the probability of system failure [7].

In Object oriented metrics, Chidamber and Kemerer proposed the CK metrics suite which consists of six class based design metrics meant for object oriented systems. It defines class-oriented software metrics whose focus is the class and its hierarchy using the follows Weighted Method per Class (WMC), Inheritance Tree Depth (DIT), No. of Children (NOC), Coupling between the Object Classes (CBO), Response for Class (RFC), and Lack of Cohesion in the Methods (LCOM). Lorenz and Kidd proposed a class-based metrics which was divided into four broad categories viz., size, inheritance, internals, & externals. It focuses on the counts of attributes and operations for an individual class and average values for the OO system as a whole i.e., Class Size (CS), No. of Operations Overridden by Subclass (NOO) , No. of Operations Added by Subclass (NOA), and Specialization Index (SI) [7, 10].

MOOD metrics suite was proposed by Harrison, Counsell, and Nithi for object oriented design. It consists of a set of six metrics as follows i.e., Method Hiding Factor (MHF) , Attribute Hiding Factor (AHF), Method Inheritance Factor (MIF), Attribute Inheritance Factor (AIF), Coupling Factor (CF), and Polymorphism Factor (PF). Operation oriented metrics was proposed by Lorenz and Kidd and provides three simple metrics focusing on individual operation's size & complexity as follows i.e., Size of Average Operation (OS_{avg}), Operation Complexity (OC), and Average no. of parameters for each operation (NP_{avg}) [7, 10].

Object oriented testing metrics was proposed by Binder who suggested a group of design metrics having a direct control upon the testability of object oriented system. The metrics are organized into categories as follows i.e., Lack of cohesion in the methods (LCOM), Percent public & protected (PAP), Public access to the data members (PAD), No. of Root classes (NOR), Fan in (FIN), No. of children (NOC), and Depth of inheritance tree (DIT) [7, 10].

The object oriented project metrics proposed by Lorenz and Kidd provides insight into the size of software using following set of project metrics i.e., Number of Scenario scripts (NSS) ,Number of Key Classes (NKC) , Number of Subsystems (NSUB) , Number of Support classes (NSC), and Average no. of support classes per key class [7, 10].

IV. Evaluation Of Existing Metrics For Security Implementation In Software

Traditional metrics lacked the security aspect because traditional software engineering approach are requirements driven and may depend upon available tools and the working environment and focus very little attention to software security. Among the metrics available in the market, many of them lack a sound theoretical base or a statistically significant experimental validation as they are generally based on empirical or historical data. The scope and limitations of the existing metrics from software security point of view are discussed below as follows:

Size oriented metrics possess restriction in terms of its validity and applicability which is a matter of global debate. According to researchers, their use in estimation is subjected to details which are not possible to achieve. The LOC measures used in size oriented metric are programming language dependent. Also, incorporating them in nonprocedural language is an issue [7]. Furthermore, the researcher, so far, have not been able to establish relationship between the software's LOC and its security. Implementation of security from the initial stages of software development is not possible using LOC as the value of LOC cannot be measured until the coding process and which happens at a very later stage of software development [8]. Based on these limitations, at present, it may be concluded that there is no scope left for using size oriented metrics for measuring the security aspect of a software system.

Determining the complexity of function oriented metrics is a difficult task as their complexity is subjective. Function points (FP) computation involves subjective evaluation at various points which may not provide unique functional points and hence dependency on analyst is needed. Both function point and feature point are language independent. It is considered as an estimation approach due to the fact that it is based on the data which should be known well in advance in the beginning of the project. It can be used in both conventional and nonprocedural languages. Although function oriented metrics is an indirect metrics still the scope of using it for measuring the security of software is limited [7]. The measurement parameters of function oriented metrics are very different from the parameters required for measuring the security of a software system hence not suitable to measure the security of software [11, 12].

The Bang metrics is used to suggest the domain of software i.e., function strong and data strong depending upon the ratio RE/FuP [7]. Although, it is a metrics for the analysis model and is used during the early stages of software development, yet, the metrics is not suitable for security aspect of software and only provides the indication of the size of the software [11, 12].

The software design complexity metrics is a metrics for design model and is applicable during the design stage of software development. The metrics suggests that as the complexity value increases the overall architectural complexity of the system also increases. It provides an indicative measure that there is likelihood that integration and testing efforts will also increase [7]. From security implementation point of view, nothing is suggested or proposed. Further, this metrics is relevant during the design stage and focus on the characteristics of the program architecture having emphasis on the module effectiveness and its architectural structure. Hence, this metrics has very little scope for implementation of security from the early stages of software development [11, 12].

Component level design metrics requires inner working of the module under construction. They may be applied once a procedural design has been developed. These metrics predicts and reveals the critical information about software's reliability and maintainability using automatic analysis of source code. They help control the design activities. During testing and maintenance stages, these metrics help in pinpointing areas of potential instability [7]. The complexity metrics provide very little about security. These types of metrics are mostly used in the debugging performance and maintenance efforts. Security is not the major issue for complexity metrics and hence this type of metrics does not serve as a security metrics [11, 12].

In relation to Interface design metrics, very little information is available and no sound metric has been published that would provide insight into the quality and usability of the interface. There is no mention on the security aspect of the software and hence interface design metrics is not deemed fit for security implementation in software development process [7]. Synchronization of source code metric proposed by Halstead with various security metrics is yet to be established [8, 11]. Again security has not been given due importance in this metrics.

As there are no efficient ways or mechanism to count the defects in software process using Defect Removal Efficiency Metrics so they are not widely used. Reliability metrics is also a quality metric and it can be implemented at the later stages to value how much reliable the security measures are within the software. This quality metric is secure as it is based on time constraint measuring failure at a particular time interval [7, 8]. DRE metrics and reliability metrics can only be judges in the later stages (i.e., implementation, performance reliability etc.) so they cannot be applied in the initial phase of software development and hence these type of quality metric are not considered as potential candidate for security metrics [11, 12].

The methods defined in CK Metrics suite does not state that whether they belong to one class which defines it or they belong to many classes which inherit it [7]. As the no of children will increase so the no. of class inheriting those methods will also increase resulting in more amount of time needed for testing security of the software. Lorenz and Kidd metrics has large number of overridden classes which create design problems [7]. This type of metric is difficult to test and modify so security measures cannot be updated as and when needed.

In MOOD metrics suite, as the CF increases the complexity of metrics also increases which result in poor applicability of qualities like understandability, maintainability and potential for reuse etc. [7]. Since the MOOD metrics suite is proposed to be used for object oriented design stage hence there is no scope of using this metrics suite for incorporating security '*right from the beginning*' i.e., from requirements engineering phase. In operation oriented metrics, the larger the number of operation the more complex the metric will be resulting in inaccurate results [7]. As there are large number of parameter, and judging the security of each and every parameter is not possible and cannot be valued before completion of software development process.

In object oriented testing metrics, as most of the methods are designed and inherited for public class only so the chances of secure testing decreases. This also leads to violation of encapsulation [7]. Adequate security measure cannot be implemented in the early stages of software development i.e., requirement stage. In object oriented project metrics, as size is directly proportional to the efforts and duration of so the product manager is unaware of implementation size of the metric. Since the implementation size is not known in advance so appropriate security measures cannot be undertaken before implementation of software.

V. Finding and Observation

Based on the study carried out in the field of software metrics and its relevance and scope in implementation of security in software development process following findings and observations are made:

A. *Issues and Challenges*

The complexity of security aspect is a very major factor which impacts the development and management of secure software. In the current scenario there are very few measures which can be potential candidate for measuring the security of software having a reliable, well-defined and precise evaluation mechanism. Therefore, at present, precise, accurate and effective estimation for planning and controlling the security aspects of software engineering is an issue with the research community which needs to be explored further. For incorporating security in the software development process and to measure its effectiveness we need a comprehensive software metrics whose goal is to (i) identify the essential security parameters that affect the secured software process, (ii) measure it, and, (iii) if required, to control it.

Moreover, from risk point of view, its evaluation may give rise to a number of causes resulting in security breach like threats, asset value, and vulnerabilities. It is very difficult to inculcate all these factors while designing security metrics as these factors have numerous dimensions of their own. Although, asset value is the easiest to measure among the three factors which are mentioned above but in some cases its measurement proved to contain flaws. Some of the research and academicians are of the opinion that threat and vulnerabilities detection is not possible in the early stage and some argue that it is possible. Another problem with the security metrics is the issue of its wide acceptance. The reason behind this belief is that the security metric is in the early stages of development & there are no framed terminologies which could be used in totality as shown in figure 1.

B. *Recommendations*

Following recommendations are made based on the study of existing software metrics and its evaluation in terms of implementation of security aspect in software:

- Software Security Metrics may be designed so that its implementation is not programming language dependent.
- Some mechanism may be proposed to establish the relationship between LOC and security of software.
- The measurement parameters of existing software metrics may be extended to align it with the parameters required for measuring the security of a software system so that the existing metrics becomes suitable for measuring the security of software.
- The existing software metrics may be extended to make it independent of software development phase so that the enhanced software metrics may be used for security 'right from the beginning' i.e., the requirements engineering phase.

C. *Proposed Solutions*

Based on the research carried out and issues and challenges discussed, following solutions are proposed which could be carried out in case of future research directions in the field of software security metrics:

1) *Solution 1:* An extension of existing metrics for measuring the security of software as shown in figure 2.

2) *Solution 2:* Development of Metrics suite initially applicable for individual stages for measurement of security and finally a cumulative measure of all stages may be calculated and achieved indicating the degree of overall security of software as shown in figure 3.

3) *Solution 3:* Development of comprehensive metrics suite for measuring the security of software with its applicability 'right from the beginning' i.e., the requirements engineering phase in a waterfall model approach as shown in figure 4.

4) *Solution 4:* Development of comprehensive metrics suite for measuring the security of software with its

applicability 'right from the beginning' i.e., the requirements engineering phase in a spiral model fashion as shown in figure 5.

VI. Conclusion And Futurework

In the present scenario where software security is a subject of major concern, it is very necessary to safeguard the software from malicious attacks. It is necessary that security should be incorporated '*right from the beginning*' i.e. the requirement stage so we can provide strong and secure software which may continue to work efficiently in malicious environment. Any process undertaken to improve security should be properly measured for its smooth functioning. The use of security metric has proved as a potential mechanism to measure the security of software systems. It is a quantifiable approach of measuring security.

All the metrics which are discussed in this paper are product metrics and pay less attention to the security of the software. These types of metrics are generally implemented in the later stages of the software development so it limits the applicability of its security aspects. Most of the software development models available are probabilistic and pragmatic so in this case the application of software metric may become difficult and costly.

But the applicability of existing metrics for implementing security in the software development life cycle has its limitations. One of the future works includes extending the existing metrics for implementation of security during the early development stages of software. A comprehensive set of metrics suite covering all the stages of software development can be a potential future work. Another future work may include a more comprehensive approach in development of software security metrics with a synchronized approach with existing traditional metrics which can address the issue of security implementation in SDLC '*right from the beginning*' to produce secure software.

References

[1] C. Banerjee, S. K. Pandey (2009): "Software Security Rules: SDLC Perspective", International Journal of Computer Science and Information Security, IJCSIS, USA, Vol. 6, No. 1, October 2009, pp. 123-128.

[2] Gary MaGraw: "Software Security – Building Security In", Addison-Wesley Software Security Series, 2006 ISBN 0321356705.

[3] Chess, Brian. "Metrics that matter: Quantifying software security risk." In Workshop on Software Security Assurance Tools, Techniques, and Metrics, pp. 500-265. 2006.,

[4] Lance Hyden, "IT Security Metrics: A Practical Framework for Measuring Security & Protecting Data", McGraw Hill, 2010

[5] Mukta Narang and Monica Mehrotra, "Security Issue – A Metrics Pespective", International Journal of Information Technology and Knowledge Management, Vol. 2(2), July-Dec 2010, pp 567-571.

[6] Smriti Jain, Maya Ingle, "Review of Security Metrics in Software Development Process", International Journal of Computer Science and Information Technologies, Vol. 2 (6), 2011, ISSN 0975-9646, pp 2627-2631.

[7] Roger S. Pressman, "Software Engineering a Practitioner's Approach", 7th Edition, McGraw Hill Publications, 2009, ISBN 978-0073375977.

[8] Sree Ram Kumar T, Sumithra A, Alagarsamy K, "The Applicability of Existing Metrics for Software Security", International Journal of Computer Applications, Volume 8– No.2, October 2010, ISSN 0975-8887, pp 29-33.

[9] Gurdev Singh, et al., "A Study of Security Metrics", IJCEM International Journal of Computational Engineering & Management, Vol. 11, January 2011, ISSN 2230-7893, pp 22-27.

[10] Amjan Shaik, et al., Statistical Analysis for Object Oriented Design Software Security Metrics, International Journal of Engineering Science and Technology, Vol 2(5), 2010, 1136-1142.

[11] Kan, Stephen H., "Metrics and models in software quality engineering", Addison-Wesley Longman Publishing Co., Inc., 2002.

[12] Klasky, Hilda B., "A Study of Software Metrics." PhD diss., Rutgers, The State University of New Jersey, 2003.

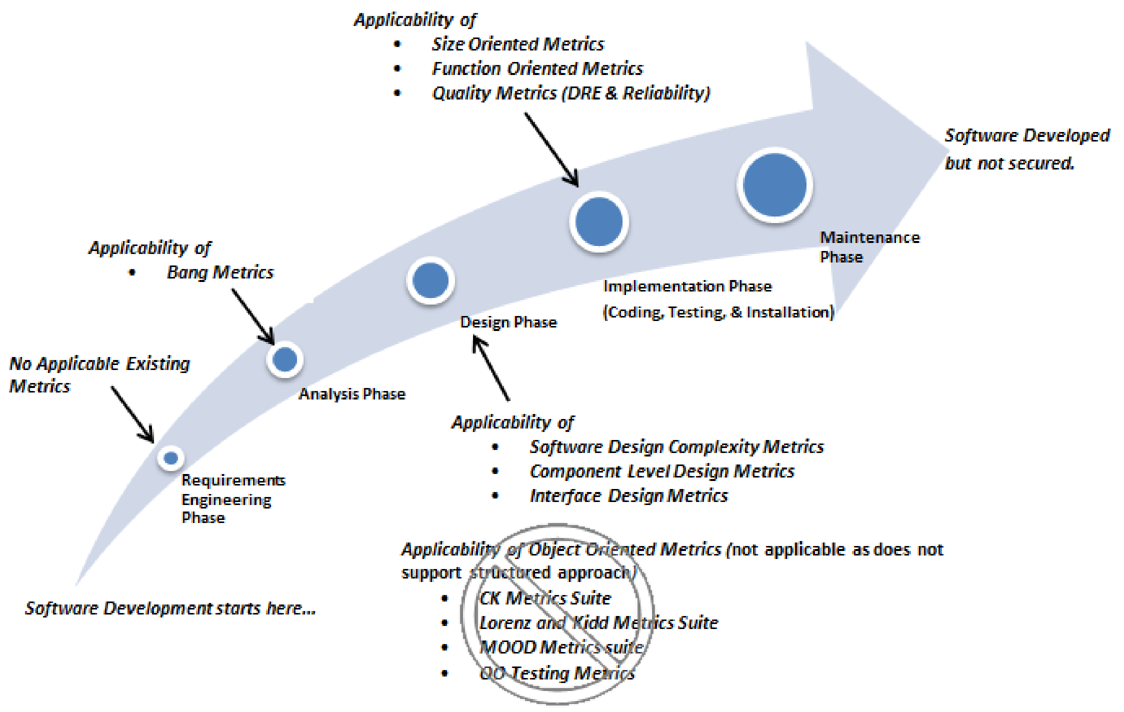
Applicability of
• Size Oriented Metrics
• Function Oriented Metrics
• Quality Metrics (DRE & Reliability)
Software Developed but not secured.
Maintenance Phase
Implementation Phase (Coding, Testing, & Installation)
Applicability of
• Bang Metrics
Design Phase
No Applicable Existing Metrics
Analysis Phase
Applicability of
• Software Design Complexity Metrics
• Component Level Design Metrics
• Interface Design Metrics
Requirements Engineering Phase
Applicability of Object Oriented Metrics (not applicable as does not support structured approach)
• CK Metrics Suite
• Lorenz and Kidd Metrics Suite
• MOOD Metrics suite
• OO Testing Metrics
Software Development starts here...

Figure 1. Present Scenario of Existing Metrics and its applicability from Security point of view in SDLC.

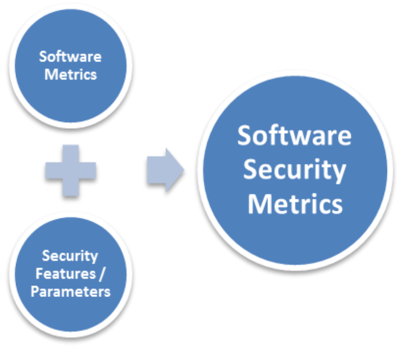

Figure 2. Solution 1: An extension of existing metrics for measuring the security of software

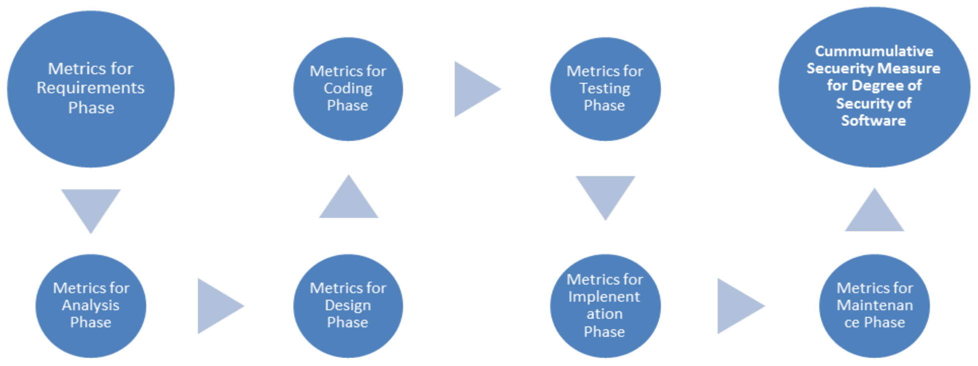
Metrics for Requirements Phase
Metrics for Coding Phase
Metrics for Testing Phase
Cummumulative Secuerity Measure for Degree of Security of Software
Metrics for Analysis Phase
Metrics for Design Phase
Metrics for Implentation Phase
Metrics for Maintenance Phase

Figure 3. Solution 2: Development of Metrics suite initially applicable for individual stages for measurement of security and finally a cumulative measure of all stages may be calculated and achieved indicating the degree of overall security of software.

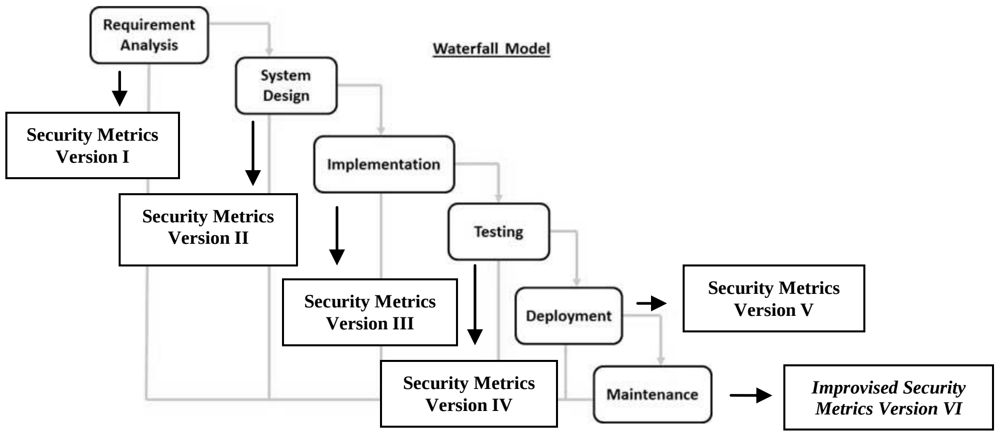
Waterfall Model
Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance
Security Metrics Version I
Security Metrics Version II
Security Metrics Version III
Security Metrics Version IV
Security Metrics Version V
Improvised Security Metrics Version VI

Figure 4. Solution 3: Development of Metrics suite initially applicable for individual stages for measurement of security and finally a cumulative measure of all stages may be calculated and achieved indicating the degree of overall security of software.

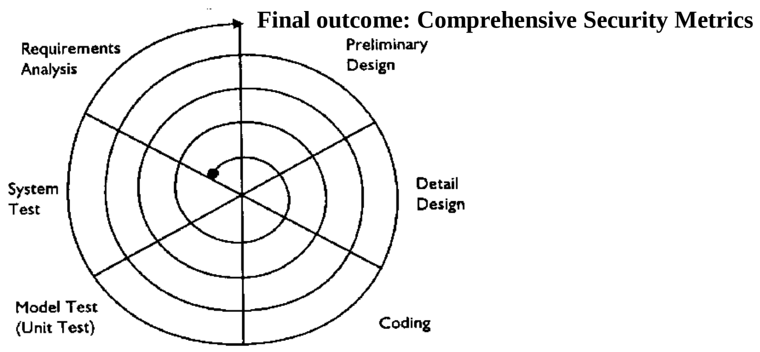

Figure 5. Solution 4: Development of Metrics suite initially applicable for individual stages for measurement of security and finally a cumulative measure of all stages may be calculated and achieved indicating the degree of overall security of software.