International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 12 | Dec -2016 www.irjet.net p-ISSN: 2395-0072

Software Security Metrics

Prajakta J. Jadhav, Prof. Torana N. Kamble

Computer Engineering Dept. Bharati vidyapeeth collage of Engineering, Navi Mumbai
Prof, Computer Engineering Dept. Bharati vidyapeeth collage of Engineering, Navi Mumbai

---

Abstract - *On security issues, measurement is very important for understanding and evaluating the performance and comparison. Many metrics have propose to measure various constructs of Object Oriented paradigm such as class, coupling cohesion, inheritance, information hiding and polymorphism and use for various aspects of software quality. The use of static metrics is insufficient for Object Oriented software due to presence of run time polymorphism, template class, methods, dynamic binding and some code left unexecuted due to specific input condition. For that use of dynamic metrics instead of static metrics to compute the software characteristics and deploy them for maintainability prediction.*

***Key Words*: Metrics, Software Security, Design level Metrics, Code level Metrics**

1.INTRODUCTION

Computer security the first step to take before measuring any security is to define what exactly is meant by 'security'. So what is computer security? Few actually attempt to define it, even though most agree that having it is good. In the scope of this thesis, computer security is defined as in, which starts with dividing computer security into software and application security. Other aspects of security engineering, such as physical security, are not considered while discussing computer security. The reason for this narrow scope is an important idea expressed in and : The central culprit of issues concerning computer security is actually software security.

Software security- means designing, building and testing software for its security. Software security should not be confused with security software. The point of software security is to ensure that people developing software do a better job in considering security as an integral part of the software. Software security takes into account both security mechanisms and design for security. Software engineers, built the software with security. Another problem in building secure software is how to measure the security of the software. A metric has multiple possible definitions depending on the chosen source and some sources even try to avoid using the term altogether . A useful metric is one that "quantitatively characterizes a property", implying that there has to be a property to characterize.

The measurement theory also defines the terms 'measure', 'measurement' and 'value'. A measure is something that a metric needs, such as an instrument or a formula that allow the metric to be applied to related objects under inspection. A measurement is a process to get the results with the measure. Finally, all measurements need to end up with a value. According to, whenever there is a need for a measurement, all measurements end up using these five critical elements:

- The property to be measured needs to be identified.
- A metric needs to be defined to quantitatively characterize the property.
- A measure needs to be developed that applies the metric to a target.
- A measurement process needs to be designed.
- Each measurement needs to have a value and an estimate of its accuracy.

In this thesis, the definition of metric is: "a consistent standard for measurement" as defined in. According to, a good metric should be:

- Consistently measured without subjective criteria.
- Cheap to gather, preferably in an automated way.
- Expressed as a cardinal number or percentage instead of qualitative labels.
- Expressed using at least one unit of measure, such as "defects", "hours" or "dollars".
- Ideally, it is contextually specific.

This thesis uses a simple measurement process as presented in:

- Metrics need to be available.
- A suitable metrics framework needs to be chosen and implemented.
- Measurements need to be interpreted.

Need-

Sometimes anything express in the number, you know something about it. But when you cannot measure it when you cannot express it for that we need metrics. We can not control things which we can not measure. Metrics are used to measure the quality of the project. Metrics is unit used for describing an attribute. Metric is scale for measurement. Before choosing suitable metrics frameworks, this chapter

explores and presents the currently available software security metrics.

1.1 Categories of Metrics:

Software security metrics can be categorized in multiple ways that represent viewpoints or abstractions within the metrics. The reason for different viewpoints is obvious: a manager has very different needs for the metrics from a software developer. The categories indicate the environment where the metric works well or is designed to work while showing where the metric is most likely to fail.

Here we are taking two categories:

I. Design Level Metrics (Static Metrics)
II. Code Level Metrics (Dynamic Metrics)

For design level metrics and code level metrics has same properties:

1. Coupling
2. Cohesion
3. Inheritance

Property	Design Level Metrics	Code Level Metrics
Coupling	DAC	MPC
	MOA	RFC
Cohesion	CAM	LOCM
Inheritance		DIT
		NOC

Fig.1 Properties based metrics

1. Coupling

Coupling means the degree of interaction an object has with other objects. Objects with high coupling are greater target for successful attacks than objects with small coupling.

1.1 Coupling support in design level metrics DAC-Data Abstraction Coupling, MOA-Measure of Aggregation.

DAC-number of abstract types defined in a class. Abstraction is a programmer hides all but relevant data about an object in order to reduce complexity and increase efficiency. It measure the number of object classes within the given class. Any data type with other data types as members or local variable that is an object of another class has data abstraction coupling. Higher Data Abstraction Coupling is more complex structure of the class.

MOA-Number of data declaration whose types are user defined classes. This metric measures the extent of the part-whole relationship, realized by using attributes. The metric is a count of the number of data declarations (class fields) whose types are user defined classes.

1.2 Coupling support in code level metrics MPC-Message Passing Coupling, RFC-Response For Class.

MPC-It measures the number of message passing among objects of the class. A large number indicates increased coupling between class and other classes in system. Classes are dependent on each other which increase the overall complexity of system

RFC- The metric called the response for a class measures the number of different methods that can be executed when an object of that class receives a message. Ideally, we would want to find for each method of the class, the methods that class will call, and repeat this for each called method, calculating what is called the transitive closure of the method's call graph. This process can however be both expensive and quite inaccurate. In ckjm, we calculate a rough approximation to the response set by simply inspecting method calls within the class's method bodies. The value of RFC is the sum of number of methods called within the class's method bodies and the number of class's methods.

2. Cohesion

It measure how well the methods of class are related to each other. It has low cohesion and high cohesion several describable including robustness, reusability, understandability. Low cohesion several undesirable test maintain reuse is very difficult.

2.1 Cohesion support in design level metrics CAM-Cohesion Among Methods of Class

The sum of intersection of method parameters with the maximum independent set of all parameter types in the class. This metric computes the relatedness among methods of a class based upon the parameter list of the methods. The metric is computed using the summation of number of different types of method parameters in every method divided by a multiplication of number of different method parameter types in whole class and number of methods. A metric value close to 1.0 is preferred.

2.2 Cohesion support in code level metrics LOCM-Lack of Cohesion Method

It indicates whether a class represents a single abstraction or multiple abstractions. If class represents more than one

abstraction, it should be refectories into more than one class each of represent the single abstraction. Aim of this is to detect problem of classes. If the LOCM value is high, it means low cohesion.

3. Inheritance

Inheritance allows to provide classes with generalizations and special relationships. Inheritance allows reuse. Inheritance could allow subclasses access to classified information. It measure various aspect of inheritance such as depth and breadth in hierarchy and overriding complexity.

3.1 Inheritance does not support in design level metrics.

3.2 Inheritance support in code level metrics DIT-Depth of Inheritance Tree, NOC-Number of Class.

DIT-Depth of Inheritance Tree

The (DIT) metric provides for each class a measure of the inheritance levels from the object hierarchy top. In Java where all classes inherit Object the minimum value of DIT is 1. And root class consider as an zero.

NOC-Number of class

It measure the total number of direct subclasses of a class at run time. Classes with large number of children are considered to be difficult to modify, so required testing because of the effect on change on all children. It is more complex because they have numerous children.

4. Comparing design and code level metrics

Boehm observed that fault removal is 50 to 100 times less costly when performed in the design phase rather than after the deployment. As a practical point of view, software engineers need to be aware that the metrics results are tool dependent, and that these differences change the advice the results imply. As a scientific point of view, validations of software metrics turn out to be even more difficult. Since metrics results are strongly dependent on the implementing tools, a validation only supports the applicability of some metrics as implemented by a certain tool. More effort would be needed in specifying the metrics and the measurement process to make the results comparable and generalizable.

5.CONCLUSIONS

Software metrics are usually used to measure some aspect associated with software development. These aspects may include estimation, detection and prevention of issues. The utilization within measurement framework and the use of automated tools can help towards development process control and higher quality software, so the design and code level metrics are not same in performance. Code level metrics is better than the design level metrics.

REFERENCES

[1] Hemlatasharma, Anuradha chug “Dynamic metrics are superior than static metrics in maintainability predication”,IEEE Transactions on 2015M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[2] Chidamber S.R., Kemerer, C.F.: "A metrics suite for object-oriented Design", IEEE Transactions on SW Engineering, Vol. 20, No.6, June 1994.K. Elissa, “Title of paper if known,” unpublished.

[3] C. B. Chowdhury, Istehad and M. Zulkernine, “Security metrics for source code structures,” in Proceedings of the Fourth International Workshop on Software Engineering For Secure Systems. Leipzig, Germany: ACM, 2008, pp. 57–64.

[4] Everald E. Mills : “ Software Metrics”, SEI Curriculum Module SEI-CM-12-1.1, Software Engineering Institute, Carnegie Mellon University, December 1988.

[5] K. Maruyama, “Secure refactoring - improving the security level of existing code,” in Proceedings of the Second International Conference Software and Data Technologies (ICSOFT 2007), Barcelona, Spain,2007, pp. 222–229.

[6] R. Malhotra, A. Chug, “An Empirical Study to Redefine the Relationship between Software Design Metrics and Maintainability in High Data Intensive Applications” , Lecture Notes in Engineering and Computer Science, Proceedings of The World Congress on Engineering and Computer Science, USA, pp. 61-66, 2013.

[7] J. Alghamdi, R. Rufai, and S. Khan. Oometer: A software quality assurance tool. *Software Maintenance and Reengineering, 2005. CSMR 2005. 9th European Conference on*, pages 190–191, 21-23 March 2005.

[8] Aqrissoftware. http://www.aqris.com/.