International Journal of Engineering & Technology, 7 (2.32) (2018) 367-370

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

SIS Framework for Risk Assessment Through Quantitative Analysis

[1]D.NagaMalleswari, [2]Dr.K.Subrahmanyam

[1,2]Dept of CSE , K L E F,Vaddeswaram, India
**Corresponding author E-mail: nagamalleswary@kluniversity.in*

Abstract

Now a days, risk management plays very important role in Information systems, currently there are various risk assessment techniques. When system is analysing the source code, automatically some disputes may arise which depends on various reasons. These disputes may arise some of the risks in information system which may leads to loss of some data. To avoid that, in this paper we are implementing a framework for source code analysis which is used for brief assessment of risk, which includes guidance to risk minimization. In this framework source based risk assessment is done through the source code analysis. In order to assess risk that arose from the source code, first we need to calculate complexity of a source code in Information System. Finally the complexity which is the result of this framework will indicates the risk intensity of the source code.

Keywords*:Risk Assessment, Risk Management, Source code analysis, Effort or Performance.*

1. Introduction

Software Product will be developed in order to meet the collection of conditions and requirements of a group of business people. The necessity for high software quality continues to raise because the software systems dependencies and destructing effects due to a failure or a software error can cause in terms of finance and life-time.[1]

Therefore, it is major and dominant that the software system works mainly according to its functional requirements, and also to the non-functional requirements which decides the software product's quality. If a software system is more complex, the mistakes will be more done by the programmers, introduction of faults that can lead to an erroneous output or an execution failure in an inconsistent component. The relationship between failure and a fault, a risk observed in software system can be seen as a potential issue. [7] To reduce software operation risks, code which has the issue to cause errors and risk has to be identified so that obligatory actions like performing a more testing on the code that can be taken to obviate the occurrence of such problems. Consequently, these things can make programmers or the developers to detect risks in the software before it is sited and reduction in code maintenance.

2. Literature Review

Risk Management can be defined as the measure of risk containment and mitigation. Good software helps an organization to grow better, where unproductive software may lead an organization go worse. The software failures are caused by the reasons are called Risk. [2] It is observed that risk has been arising majorly at five different areas during the development of an software application they are Application and system architecture, New and unproven technologies, Organizational, User and functional requirements, Performance

Assessing and analysing of risk are the keys ways to eradicate risk and make the project successful. Assessment of risk can be done in many ways and many more methods were proposed. Some of them are mainly assessed manually or traditionally. We are mainly concentrated on the assessment of source code analysis. We found few models assessed source code through gathering information from stakeholders and reviews conducted on the design of documents. These were considered to be the secondary fact retrieval. The system's source code is completely analysed and this may include the code which has been written in different languages for different interactions and subsystems and, this is considered as a primary fact retrieval. Finally, the obtained results from the analysis of source code are compared with obtained results from the reviews and opinions of stakeholders. Therefore this helps to find a way to validate the views and hence they decide whether it is a risk or a misunderstanding. [3] Few proposed models of risk as dynamic and static risk assessment models. The risk is assessed block wise i.e. methods, files and functions of an application of software. Static model deal with the code structure such as number of p-uses,c-uses , function calls, definitions etc. and other model which is dynamic describes the test coverage of the dynamic system of the calibrated metrics which are used in the model. In this we found mainly two principles. They are 1. The higher risk means the more complex in the coded structure 2. The less risk means more thoroughly the code is tested. [4]

3. Existing System

The traditional quantitative measurement by calculating the risk impact by collection of information from various available sources. This is the main method in monetary value calculation. EMV can be abbreviated as Expected Monetary Value. The EMV is calculated as given below:-

$$EMV = \text{Probability} \times \text{Impact} \quad (1)$$

$$\text{Impact} = \text{maximum Impact} \times Pi \quad (2)$$

$$EMV = Pe \times Pi \times \text{maximum Impact} \quad (3)$$

Pe represents the Probability of an event.
Pi represents the Probability of a maximum value. [5]

There is one more metric i.e. MR which means Management Reverse [2]. MR is given as

$$MR = \sum (\text{probability}_i \times \text{impact}_i)$$

This parameter is used to reduce the risk. [2]
From this existing system we also have two more approaches 1. Datrix approach 2. Risk Assessment Using Source Code Approach.

Datrix Approach: This Datrix approach which is identified at Bell Canada. In this approach source of an application is analysed. Its main objective is to attain maintainability of the application or software with respect to source code. Datrix approach related to the concept of Abstract Semantic Graph. To get such a graph we need to parse the source code. The Abstract syntax tree is obtained (AST). [2] The AST is implemented to extract semantic information like variable type, scope of the identifier etc. This information which is obtained is then added to the Abstract Syntax Tree as edges, node attributes or other kind of annotations, that gives the productions of Abstract Syntax Tree. To find out more risks, this graph must be more considerable [6].

Risk Assessment using Source Code Approach

This approach helps to gauge the risk by following process which divides the process, as two phases. The information came out during first phase is produced by the automatic analysis of the Project source code. This information called as primary information.

Secondary phase, in this secondary information taken from reviews of the software application developers. A gauge or tool in the JAVA based is introduced for the collection of primary data using the analysis of source code. The information which is gathered in the initial phase are to be placed in the corresponding tables in the database. After this stage the risk analysis is analysed by the core analysers to write necessary corrections to get the effective information.

4. Proposed System:

From the disadvantages of existing system to risk assessment like shortages of the determining tool for versions, shortage of global view, which may help for the representation of the sub graphs. There is tight dependency between the outcomes and the analyser's shortage of few metrics for assessment of risk. Our proposed system which is a framework, mainly overcomes the disadvantages of the existing system on assessing the source code risk in information systems with help of the Cyclomatic complexity. The framework is designed in such a way that, by applying various software metrics to a source code which can be an effective gauge of software Risk. Our framework follows an order as shown below.

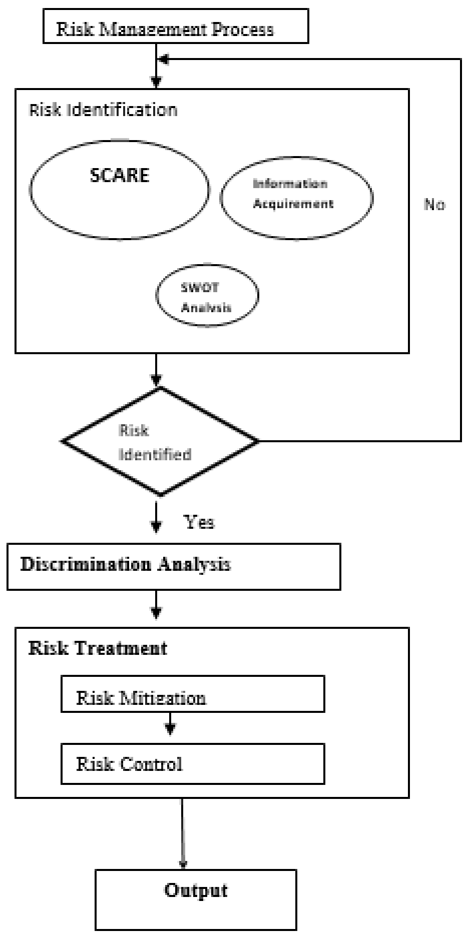


Fig: SIS Frame Work

From the above figure it is clear that any process related to risk first starts and comes under Risk Management Process. We are under the step of risk mitigation. Mitigation is nothing but an assessment. This framework well suits that. So, this framework follows the risk management process and finds out the intensity of the risk by using Cyclomatic complexity. This results the complexity. Based on the table provided in the cyclomatic complexity Table 1, we can decide how much risk does our source code has? Considering that complexity we can make more changes to the code and make that code more efficient. So we can say that this framework helps to make the source code more efficient by minimizing the risk.

Based on the complexity calculation, there is a way by which complexity can be calculated i.e. Cyclomatic complexity, can be helpful in assessing effort of the source code that used to prevent future maintenance issues and software Risk.

Cyclomatic complexity is used to estimate the total complexity of a real time application or specific methodology in it. The software source metric numerically measures a program's analytical strength based on flow and decision paths in the source code. it is measured from the control flow graph, where every individual node on the control flow graph which represents undividable groups or commands within the source code.

It is used to measure the performance of any type of source code. Self-determining path is a path which has minimum one edge which was not approved or visited before.

Cyclomatic complexity can be calculated with respect to procedures, macros, methods, modules, classes within a program [10].

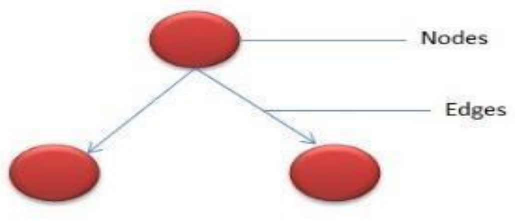


Figure 1: Representation of nodes and edges

Representation of Flow graph for a Source program:
Representation of Flow Graph for a program is defined as many nodes connected with the edges. Following are Flow graph figures for following statements like looping statements like while, if-else statement, until and normal sequence of flow.

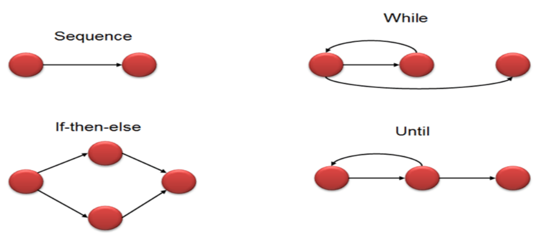


Figure2: Flow graph representation for statements.

Mathematical demonstration of Cyclomatic Complexity:

Mathematically, it is collection of self-determining paths through the flow graph diagram. The complexity of the Source code can be measured through the formula-

CV(G) = EN - NN + 2

Where,
NN - Count of Nodes
EN - Count of edges
CV (G) = PN + 1

Where PN = Count of predicate nodes (conditional nodes)

Cyclomatic complexity can be calculated traditionally by the above mathematical notation if the source code is little and minor. Computerised tools must use if the source code is compound or if the lines of code is as immense as this includes many control flow graphs. On Basis of complexity value, the coding team or the developing team can decide the necessary modifications can be taken for evaluation. Source code with a complexity value of below or equal to 10 can be treated as low complexity or complexity with considerable range. This effort value calculated can be used to assess the risk factors and can be helpful to the improvement of source code part. In Software system majority of the risks can be caused by the Source code of the project. Hence deducing the risks in the source code can help the system to be more risk free. Using the Cyclomatic complexity as a primary factor for an application or a system helps the organization to identify the major - risk parameters and helps to develop for the improvement or the adjustment approaches to reduce the threats or identified risks, repair time, productive issues, technical issues in the source code. Understanding the system's complexity provides clear analysis regarding where a developed source program needs additional improvement or effort to be kept in order to get the successful development in the multitier, multiple technology organization. [8]

The table gives layout on the complexity value and equivalent denotation of CV (G):

Table1: Complexity value equivalence meaning

Complexity value	Description
1-10	Cost and Effort is less High Testability Structured and well written code
11-20	Cost and effort is Medium Medium Testability Complex Code
21-40	Cost and Effort are high Low Testability Very complex code
>40	Very high cost and effort Not at all testable

Tools for calculation of Cyclomatic Complexity:

Different tools are obtainable for determining the complexity of the software system. There are defined tools are used for certain technologies and languages. Complexity has been observed by the total count of decision making nodes and number of individual functions in a Source code. The looping statements like for-each, while,for,do-while,catch,try,switch-case statements and decision making points are conditional statements like if-else in a source code.

Examples of tools for Cyclomatic complexity calculation are

- **CCCC** - C and C++ Code Counter
- **GMetrics** – Find complexity metrics in Java, JSP based applications
- **OCLint** - Analysis of Static code for C, JAVA and technical level Languages

In this paper we considered the CCCC tool to measure the Cyclomatic complexity

The CCCC is a code analyser and counter tool for the analysing the source code in multiple languages (mainly JAVA, CPP), which gives a detailed report in Table format in HTML form and XML report on different observations of the given source code. It is Free of Cost and compatible in all systems.

Depths of code of this kind are generally referred to as 'software source code metrics', or more precisely 'software risk metrics' (as the term 'software metrics` also covers measurements of the software process, which are called 'software process metrics'). There is a reasonable agreement among modern opinion privileged persons in the software engineering field that measurement of some kind is probably a Good Thing, although there is less agreement on what is worth measuring and what the measurements mean.

CCCC has been developed as free software, and given in command prompt model to input the source code. Users are supposed to compile the program themselves, and to modify the source to reflect their preferences and interests using the generated reports.

5. Result:

We have calculated the effort value for the Pharmacy management system source code in C++ using CCCC tool. The report generated consists of Lines of Code, Complexity Number, Lines of Comment, and Depth of Inheritance Tree with individual complexity values of independent functions.

From the report we can finally calculated the effort of the Information system of Pharmacy application to assess the risk in the source code of system with high efficiency effort estimating method.

Detailed Report on Module Anonymous

Metrics	Tag	Overall	Per function
Lines of Code	LOC	156	******
McCabe's Cyclomatic Number	MVG	26	******
Lines of Comment	COM	30	********
LOC/COM	L_C	5.200	
MVG/COM	M_C	0.867	
Weighted Methods per Class(weighting=unity)	WMC1	5	
Weighted Methods per class(weighing=visible)	WMCv	0	
Depth of Inheritance tree	DIT	0	
Number of Children	NOC	0	
Coupling between objects	CBO	0	
Information Flow Measure (inlusive)	IF4	0	********
Information Flow Measure (visible)	IF4v	0	
Information Flow Measure (concrete)	IF4c	0	********

Functions

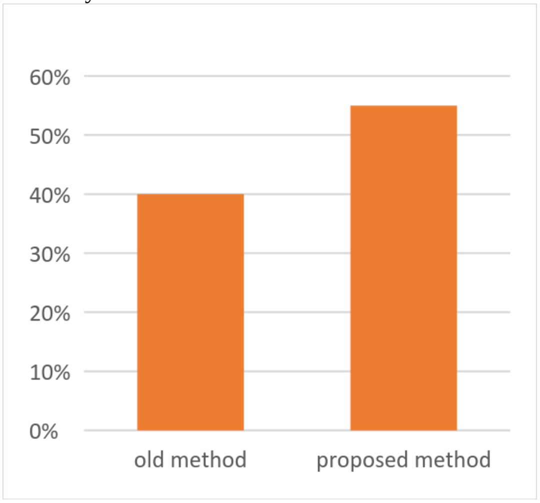
Function Prototype	LOC	MVG	COM	L_C	M_C
DepartmentMenu(queue*) definition D:\hosmng.cpp:200	97	14	13	7.462	1.077
InputPatient(void) definition D:\hosmng.cpp:152	17	4	4	-----	-----
OutputPatient(patient *) definition D:\hosmng.cpp:174	13	3	1	-------	------
ReadNumber() definition D:\hosmng.cpp:190	6	1	2	-------	------

Metrics	Tag	Overall	Per function
Lines of Code	LOC	156	******
McCabe's Cyclomatic Number	MVG	26	******
Lines of Comment	COM	30	********
LOC/COM	L_C	5.200	
MVG/COM	M_C	0.867	
Weighted Methods per Class(weighting=unity)	WMC1	5	
Weighted Methods per class(weighing=visible)	WMCv	0	
Depth of Inheritance tree	DIT	0	
Number of Children	NOC	0	
Coupling between objects	CBO	0	
Information Flow Measure (inlusive)	IF4	0	********
Information Flow Measure (visible)	IF4v	0	
Information Flow Measure (concrete)	IF4c	0	********

Result Analysis: The report generated above by the CCCC tool and calculated accurately the Cyclomatic Complexity which means the effort or performance as 26 and Lines of Code as 156 which mean it is a Low Testable Cost and Effort are high, Very complex and composite code. So, the code must be modified accordingly in order to get the low effort value.

6. Conclusion and Future Work

The research on risk management process helped us to propose a software Risk assessment Framework, which is cost effective and assessing the risk from Source code perspective. In this framework, the Lines of code will be measured and Performance of the source code is calculated in the form of complexity. One Information system was taken and the complexity value is calculated by the proposed assessment framework which produces accurate risk reports.
The efficiency of our proposed method is high while comparing to other methods with the real time projects calculated for Information systems.



Graph1: Effort analysis for Source code

Our current research assesses the performance of the source code using cyclomatic complexity. The performance of the source code can also be measured using essential complexity, module design complexity, global data complexity and specified data complexity.

References

[1] Kutay, C. and Babar, M. A. 2005. Teaching three quality assur-ance techniques in tandem-lessons learned. In Fifth International Confer-ence on Quality Software.QSIC’05. IEEE, 307–312
[2] Risks Armen Keshishian1, Hasan Rashidi21Computer science Dept, Qazvin Azad UniversityIran, Tehran 2Hasan Rashidi, Qazvin Azad UniversityIran, Tehran
[3] Arie van Deursen CWI and Delft University of Technolo-gy ,The Netherlands and Tobias Kuipers, Software Improvement Group ,The Netherlands
[4] W. Eric Wong, Yu Qi, and Kendra Cooper Department of Com-puter Science University of Texas at Dallas Richardson, TX 75083 {ewong, yxq014100, kcooper}@utdallas.edu
[5] J. Kontio, “The riskit method for software risk management”, Insti-tute for Advanced Computer Studies and Department of Com-puter science, University of Maryland, 1999.
[6] Lapierre, Sebastien, Lague, Bruno and Leduc, Charles. “Datrix Source Code Model and its Interchange Format: Lessons Learned and Considerations for Future Work. Montreal”, Canada: Bell Can-ada, Quality Engineering and Research, 2002.
[7] Lyu, M., Yu, J., Keramidas, E. and Dalal, S. ARMOR:Analyzer for Reducing Module Operational Risk. In Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Compu-ting (Pas-adena, California, June 1995). IEEE Computer Society, Washing-ton, DC, 137-142.
[8] http://sarnold.github.io/cccc/CCCC_User_Guide.html
[9] http://www.castsoftware.com/glossary/cyclomatic-complexity
[10] http://qafriend.com/software-metrics/cccc-tool-for-cyclomatic-complexity
[11] http://www.guru99.com/cyclomatic-complexity.html