Assessing the Software Complexity and Security metrics from UML Class diagram

[1] Prof.D.M.Thakore, [2]S.J.Sarde (M.Tech Student)
[1]Department of Computer Engineering
Bharati vidyapeeth Deemed University College of Engineering, Pune-43, Maharashtra, India
[2]Department of Computer Engineering
Bharati vidyapeeth Deemed University College of Engineering, Pune-43, Maharashtra, India

Abstract-As the standard definition of software engineering it is the development of sound engineering principle in order to achieve the software which is effective, efficient, understandable, and further it can be run on any real time machine. In software engineering, Software metrics are very useful for a forward engineering of re-engineering process of existing software system. Also, they are absolutely necessary in re-engineering process. They show exactness, clear picture and understanding of the existing software system. It should be the first step in effective re-engineering process.

Quality of software systems heavily depends on their structure, which affects maintainability and readability. However, the ability of humans to deal with the complexity and security of large software systems is limited. In this paper we are proposing system methodology approach, that should measure coupling, cohesion (Complexity), Data Access and Operation Access Metric (Security) which dependent on the assumption that the attributes, methods, relationship and classes of Object-Oriented systems are connected in more than one way.

Index term: Class diagram, object oriented language, software quality metrics-complexity and security, source code, Unified Modelling language etc.

I Introduction

Software engineers generally use indirect measures that lead to metrics which provide a quantitative basis for understanding the underlying information in software development processes. Software metrics have always been important for software developers to assure the quality of some representation of software and organizations are achieving promising results through their use. Therefore, to develop suitable software metrics models for user (developer) who urgently need them for re-engineering of existing software system. Therefore, this proposed methodology approach **"source code analysis for software quality metrics"** proposes that the evaluation of metrics as part of reverse engineering in re-engineering process i.e. Software Reverse Engineering Tool (SRET). SRET should be developed under two categories: 1) Complexity and 2) Security Source code requirement analysis is for extracting software quality metrics for re-engineering process is done manually or with some tool but it can take more time due to complexity and unambiguous nature of source code. If some user require particular metrics urgently to re-engineer the existing software system, because such metrics are vital or necessary for effective re-engineer process. If such metrics are extracted manually then it should take more time and also, budget. That is the derive force behind the development of such a system which does it automatically without any manual efforts

Consider e.g. Both users (developer and system analyzer) should review the source code for complexity and security metrics manually. It should be good for small software system, but if software system is large then it is quite cumbersome because large software system may contain large no. of source code (LOC) with more complexity and less security. Also, it should require more time for calculation of complexity and security and also wastage of resources.

Therefore, to develop such suitable Software Reverse Engineering Tool (SRET) which could help system analyzer and developer to review the source code and calculate the source code complexity and security with automatic framework using UML Class Diagram.

II Existing tools and their limitations

Software engineering developer and system analyst should be based on the tools for implementing these metrics to support them in quality evaluation and ensure tasks to allow to measure software quality and to deliver the information needed as input for their decision making and engineering processes. Currently a large body of software metrics tools exists. But these are not the tools which have been used to evaluate the software metrics.

1) Analyst4j : It is based on the Eclipse platform and available as a stand-alone Rich Client Application or as an Eclipse IDE plug-in. It features search, metrics, analyzing quality and report generation for Java programs [2].

2) VizzAnalyzer: It is a tool for quality analysis. It reads or parses software code and other design specifications as well as documentation and performs a number of quality analyses. The VizzAnalyzer is a framework or environment for analyses and visualizations of existing software system. The VizzAnalyzer is a framework designed to help programmers or developer or system analyst in software engineering activities like re-engineering [2,3].

Shortcomings with this technique:

1) It has high coupling values and less cohesion values.
2) Also it ignores security metrics such as Data and Operation Access Metrics.
3) It analysis only few source code language file such as for java.

III Literature Survey

Software metrics are very useful in a forward engineering of re-engineering process of existing software system. Also, they are absolutely credential in re-engineering process. They show exactness, clear picture and concise understanding of the existing software system. It should be the first step in effective re-engineering process.

Use of the powerful and practical metrics like cohesion, coupling of existing systems and develop or generate new metrics and then add them to the new complexity and security categories to enhance the quality of the software re-engineering process which leads to the enhance quality of the software which is under the process of re-engineering[4].

Software reengineering is an expensive process due to the complexity of the software, we cannot emphasise on the area where the re-engineering work is required,. Coupling and Cohesion metrics are complexity metrics out of particularly cohesion metrics have the potential to help in this identification and to measure progress. The most extensive work on such metrics is with cohesion metrics. It should use of dependence information that make them an excellent choice for cohesion measurement.

It should be raise the most important question such as does a software developer or analyst which could be access to complexity metric values for the program do a better job of restructuring the program? [5]

Security metric is not considered as much as other quality attributes such as complexity metrics. Also, most security studies concentrate on the level of individual program statements. Such type of approach makes it hard and expensive to discover and fix vulnerabilities caused by design errors in the existing system.

Therefore, in this paper we should also focus on the security design of an existing object oriented application and define security metrics. These metrics allow designers (developer or system analyzer) to find out and fix security vulnerabilities at an early stage of the re-engineering process which will help to reduce the cost of software reengineering as it reduces the rework and consumption of resources which helps the designer to review the security metrics to make particular decision about security into re-engineering approach. In particular, to propose security metrics to measure Data Encapsulation (accessibility) and Cohesion (interactions) of a given object-oriented class from the point of view of potential information such as source code. Defining another security metrics which cover the entire source code of existing software including coupling, inheritance, and cohesion [6].

IV Proposed Work and system architecture

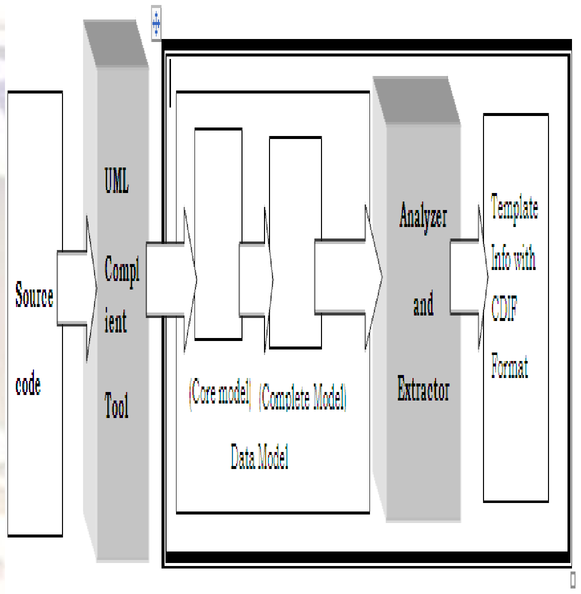

Figure 1: Proposed Architecture of System

Here take the input from user (developer), a document which contains source code. Convert this source code document into **Core Prototype Model** (class diagram) specifies the entities and relations that can and should be extracted immediately from source code. Then process this output for **Complete Model** as input specifies Object, Property, Entity and Association are made available to handle the extensibility requirement. On the output of Complete Model, then apply our rules on the basis of which Security Accessibility and complexity metrics are extracted.
In the case of in information extraction:

DAM = No. of private (protected) attributes/total no. of attributes in class
OPM = No. of public methods/ total no. of methods in a class
Cohesion = No. of methods interactions with attributes in the program code / maximum no. of methods interactions with attributes
Coupling = Access frequency of attributes of one class/sum of frequency of all attributes

1. Core Prototype Model

The Core Prototype Model states the entities and relations that can and should be extracted immediately from source code. The core model consists of the main Object Oriented entities such as Classes, Methods, Attributes and Inheritance Definition. For the reengineering we require the other two ideas namely the associations Invocation and Access. An Invocation presents the definition of a Method calling another Method and an Access presents a Method accessing an Attribute.

In automated software modelling, the Source Code as software requirement specification is translated to the formal specifications such as Template Information with CDIF Format.

2. Complete-Meta-Model

Objects, Property, Entity and Association are made accessible to handle the prerequisite. For specifying language plug-ins, it is authorized to define language precise classes and to add language precise attributes to existing Objects. Tool prototypes are more limited in extending the model: they can define tool precise properties for and can add attributes to existing Objects. They are, however, not authorized to extend the repertoire of entities and associations.

3. CDIF Information Exchange Format

CDIF is standard for transferring models or standard for information exchange with different tools.

Key issue in the reengineering of large scale object-oriented systems is due to the heterogeneity in today's object-oriented programming languages.
Proposed system also generates Template Information into CDIF Form for these programming constraints.

This is also added facility provided in the proposed system as compared to current existing work.

V Conclusion

In this paper we have emphasise on the software quality metrics complexity and security via analyzing UML class diagram which is obtained as an input from the source code and the document specification. The proposed work is fully automated eliminating the manual effort required from the developer and analyzer, further because of the elimination of manual work these system is effective, efficient for the reengineering of the software which already in existence with effective utilization of the key resources .

VI References

[1] "Beyond Language Independent ObjectOrientedMetrics:Model Independent Metrics" Michele Lanzalanza@iam.unibe.ch Software Composition Group Universit ´a di Berna, Svizzera and St´ephaneDucasseducasse@iam.unibe.ch Software Composition Group Universit ´e de Berne, Suisse

[2] "Comparing Software Metrics Tools" RudigerLincke, Jonas Lundberg and Welf Lowe Software Technology Group School of Mathematic Mathematics and Systems EngineeringVaxjoUniversity,Sweden{rudiger.lincke|jonas.lundberg|welf.lowe}@vxu.se

[3] A Qualitative Evaluation of a Software Development and Re-Engineering Project Thomas Panas,RudigerLincke, Jonas Lundberg,Welf Lowe Software Technology Group MSI, University of Vaxjo, Sweden

[4] "Development and Application of Reverse Engineering Measures" in a Re-engineering Tool S. Zhou, H. Yang and P. Luker William C. Chu Department of Computer Science Department of Information Engineering De Montfort University Feng Chia University England Taiwan

[5] "An Empirical Study of Slice-Based Cohesion and Coupling Metrics" Timothy M. Meyers and David Binkley Loyola College in Maryland Baltimore, Maryland 21210-2699, USA {tmeyers,binkley}@cs.loyola.edu

[6] Alshammari, Bandar and Fidge, Colin J. and Corney, Diane (2009) "Security metrics for object-oriented class designs". In: QSIC 2009 Proceedings of : Ninth International Conference on Quality Software , August 24-25, 2009, Jeju, Korea. (In Press)

[7] "New Conceptual Coupling and Cohesion Metrics for Object-Oriented Systems"BélaÚjházi, Rudolf Ferenc, TiborGyimóthy University of Szeged, Hungary Department of Software Engineering ujhazi.bela@stud.u-szeged.hu, {ferenc, gyimi}@inf.u-szeged.hu and Denys Poshyvanyk The College of William and Mary, USA Computer Science Department denys@cs.wm.edu

[8] "Reverse Engineering Component Models for Quality Predictions" Steffen Becker, Michael Hauck, and MirceaTrifu FZI Research Center Software Engineering Karlsruhe, Germany Klaus Krogmann Karlsruhe Institute of Technolgy Software Design and Quality Karlsruhe, Germany Jan Kofroˇn Charles University in Prague Distributed Systems Research Group Prague, Czech Republic

[9] "An Exchange Model for Reengineering Tools" Sander Tichelaar and Serge Demeyer, Software Composition Group, University of Berne, Switzerland, {demeyer,tichel}@iam.unibe.ch

[10] "A Visual Analysis and Design Tool for Planning Software Reengineerings" Martin Beck, Jonas Tr ¨umper and J¨urgenD¨ollner {martin.beck}, {jonas.truemper}, {juergen.doellner}@hpi.uni-potsdam.deHasso-Plattner-Institute – University of Potsdam, Germany