

Measuring the Quality of Software through Analytical Design by OOAD Metrics

S.Pasupathy
Associate Professor,
Dept.of CSE, FEAT,
Annamalai University,
Tamil Nadu, India,

R.Bhavani, PhD.
Professor,
Dept.of CSE, FEAT,
Annamalai University,
Tamil Nadu, India,

ABSTRACT

Software plays an important role in today's computerized world. The programmer can use different languages to develop the software. In order to develop software, it needs several phases such as Analysis, Design, Implementation, Testing and Maintenance. Of these several phases, the analysis and design becomes essential, since these are the most essential feature in the development of the software.

Now-a-days, most of the software is object-oriented, because the object-oriented languages provide easy way to develop and maintain the program. This object-oriented program consists of several divisions based upon the purpose. Each division performs some functions depending upon the code. All these divisions are then integrated to provide the single program. If any error occurred in any part of the program means, it is necessary to change the error in whole program. To avoid this kind of unnecessary change with long time duration, the developer has to overview and tests the initial phase such as analysis and design. These kinds of testing on analysis and design for an object-oriented program is carried out by a technology called **OOAD (Object Oriented Analysis and Design)**.

In this paper, a methodology has to be proposed to analysis the design to be carried out in the development of the program, before start to implement. The methodology also provides many essential features that are used to automate the process of testing on an object-oriented analysis and design. This can be done by implementing the configuration file for detecting the error rate. Thus this paper provides efficient strategy for OOAD.

Keywords

Analysis, Configuration File, Design, Error Rate, Implementation, Maintenance, OOAD, Object-Oriented, Phases, Software, Testing.

1. INTRODUCTION

Now-a-days, everything becomes computerized. In order to change the things computerized, all the things are converted into software format. This software can be developed by means of programming languages. Of these languages, many different kinds are exists based upon the type of complexity. Initially, the programming languages

did not follow any structure and thus it was termed as Unstructured Programming Languages. The next step is to overcome the unstructured drawback by means of introducing Structured Programming Language. In this structured programming language, as the length of the program increases, readability becomes difficult. That kind of languages is also termed as Procedure-Oriented Language (POL).

To overcome this readability problem, a new and emerging technology is introduced named as "Object-Oriented Programming (OOP)". In this object-oriented programming language, based upon the difficulty and functionality of the program, the program is divided into multiple sub-divisions and each performs the specified functions. All these things are grouped under a single object named as Classes. These kind of object-oriented programming languages are mostly used to develop the program successfully.

To develop the program or software, it is necessary to follow several steps as stated in the software engineering. Software Engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines. Software Engineering: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is applicable of engineering to software.

In order to develop software, the major essential steps are as follows:

- Analysis
- Design
- Implementation
- Testing
- Maintenance

Of these, the major care must be taken in the part of Analysis and Design. These are the initial steps to develop the software. If any changes held in these steps, the whole software might be affected. In order to measure the quality of the software from the initial phase and to validate the software to deduce the usefulness of the software based upon the error rate is carried out by using the new technique called OOAD (Object-Oriented Analysis and Design).

OOAD is a technique that consists of lot of metrics to validate and measure the quality of the software. This OOAD consists of several advantages over the quality measure of the software. The key idea about OO Design:

- Much of OO design is about managing dependencies
- It is very difficult to write OO code without creating a dependency on something
- 99.9% of lines of code contain at least one significant design decision
- Anyone who writes a line of code is defining the design

The OO Design goals are as follows:

- Make software easier to change when the programmer or user wants to change.
- The programmer or user might want to change a class or package to add new functionality, change business rules or improve the design
- The programmer or user might have to change a class or package because of a change to another class or package it depends on (e.g., a change to a method signature)
- Manage dependencies between classes and packages of classes to minimize impact of change on other parts of the software
- Minimize reasons that modules or packages might be forced to change because of a change in a module or package it depends upon.

Object-oriented technology is becoming increasingly popular in industrial software development environments. This technology helps in the development of a software quality of higher quality and lower maintenance costs. Since the traditional software metrics aims at the procedure-oriented software development so it cannot fulfill the requirement of the object-oriented software. As a result, a new set of object-oriented software metrics has been evolved.

Object-Oriented Metrics are the measurement tools adapted to the object-oriented paradigm to help manage and promote quality in software development.

Quality and Scope is the two essential features in each and every product. Also in the Software development environment, the developing software must be in good quality to lead in this computerized world. Only the best qualified software may exists in this world and can be used more popularly.

The design and development of software using object oriented paradigm is gaining popularity day by day. Object Oriented Analysis and Design of software provide many benefits to both the program designer and the user. This technology promises greater programmer productivity, better quality of software and lesser maintenance cost. Most commonly used Object-Oriented Paradigms are Java, C++, C sharp, and Vb.net. C sharp is Microsoft's new programming language for .net platform. It combines some

of the best features of modern programming language such as java, c++ or visual basic.

In this research work, many object-oriented metrics has been analyzed and develop a new technique to improve the quality and scope of the software. Also through this paper, the error rate is also defined in order to enhance the client to determine whether they have to use or left it. The proposed algorithm determines the features of the software and to evaluate the quality of the software. Thus the proposed strategy provides efficient methodology to implement the object-oriented metrics.

2. RELATED WORK

In paper [1], Claire Le Goues and Westley Weimer stated that Formal specifications could help with program testing, optimization, refactoring, documentation, and, most importantly, debugging and repair. However, they were difficult to write manually, and automatic mining techniques suffer from large amount of false positive rates. To address the problem, they proposed to augment a temporal-property miner by incorporating code quality metrics. They measured the code quality by extracting additional information from the software engineering process and using information from code that was more likely to be correct, as well as code that was less likely to be correct.

In paper [2], Aliza et al discussed about the problems that were faced in the IT industry to develop the best product at reduced rate. In order to determine quality and to improve the quality of services, the IT industry seems to be converging towards a set of commonly used metrics, including but not limited to, equipment availability, the time to resolve incidents, service providers need to adopt a consistent and continuous focus on their internal processes, etc. They studied the tradeoff between the cost of a service (as manifested by the staffing level) and the corresponding quality metrics (subject to the service level agreement). In particular, they studied the cost of service quality through the use of an optimization model that took into account the constraints and cost factors typically encountered in a service provider environment.

In paper [3], Bandar Alshammari et al presented a hierarchical model for assessing an object-oriented program's security. Security was quantified using structural properties of the program code to identify the ways in which 'classified' data values may be transferred between objects. The model began with a set of low-level security metrics based on traditional design characteristics of object-oriented classes, such as data encapsulation, cohesion and coupling. Finally, the entire program's security was summarized as a single security index value. The model was validated via an experiment involving five open source Java programs; using a static analysis tool they had developed to automatically extract the security metrics from compiled Java byte code.

In paper [4], Saad Alahmari et al discussed that Service-Oriented Architecture (SOA) was intended to improve software interoperability by exposing dynamic applications as services. To evaluate the design of services in service-based systems, quality measurements were essential to

decide tradeoffs between SOA quality attributes. In the paper they introduced the structural attribute of service granularity for the analysis of other internal structural software attributes: complexity, cohesion and coupling. Consequently, metrics are proposed for measuring SOA internal attributes using syntax code.

In paper [5], Vishwajit Joshi discussed about the use of performance measures in business was hardly new. Companies had been measuring costs, quality, quantity, cycle time, efficiency, productivity, etc., of products, services and processes. The elements of continuous improvement were built into the approach.

In Paper [6], Douglas et al stated that Open-source development processes had emerged as an effective approach to reduce cycle-time and decrease design, implementation, and quality assurance costs for certain types of software, particularly systems infrastructure software, such as operating systems, compilers and language processing tools, editors, and distribution middleware. In this paper, Douglas et al presented two contributions to the study of open-source software engineering. First, they described the key challenges of open-source software, such as controlling long-term maintenance and evolution costs, ensuring acceptable levels of quality, sustaining end-user confidence and good will, and ensuring the coherency of system-wide software and usability properties. Second, they presented the goals and methodology of the Skoll project, which focused on developing and empirically validating novel open-source software quality assurance and optimization techniques to resolve key open-source challenges.

In paper [7], Chen et al tested the hypothesis that generic recovery techniques, such as process pairs, can survive most application faults without using application-specific information. They examined in detail the faults that occur in three, large, open-source applications: the Apache web server, the GNOME desktop environment, and the My SQL database.

In paper [8], Nair et al. described a case study of combinatorial testing for a small subsystem of a screen-based administrative database. The system was designed to present users with input screens, accept data, then process it and store it in a database. The study was extremely limited in that only one screen of a subsystem with two known faults was involved, but pair wise testing was sufficient to detect both faults. In paper [9], Wallace and Kuhn reviewed 15 years of medical device recall data gathered by the US Food and Drug Administration (FDA) to characterize the types of faults that occur in the application domain. These applications include any devices under FDA authority, but are primarily small to medium sized embedded systems, and would range from roughly 104 to 105 lines of code.

In paper[10], Kuhn and Reilly analyzed reports in bug tracking databases for open source browser and server software, the Mozilla web browser and Apache server. Both were early releases that were undergoing incremental development. In paper [11], Richard stated that Exhaustive testing of computer software was intractable, but empirical

studies of software failures suggested that testing can in some cases be effectively exhaustive.

In paper[12], Edgar Gabriel et al pointed that a large number of MPI (Multiple Programming Interface- like Multitasking) implementations are currently available, each of which emphasize different aspects of high-performance computing or are intended to solve a specific research problem. It also presented a high-level overview the goals, design, and implementation of Open MPI.

In paper[13], Robyn stated that Open source software systems were becoming increasingly important these days. Many companies are investing in open source projects and lots of them were also using such software in their own work. This was also introduced the fact extraction process to show what logic drives the various tools of the Columbus framework and what steps need to be taken to obtain the desired facts.

In paper[14], Rudolf et al. presented that one area of the web services architecture yet to be standardized was that of fault tolerance for services. This paper investigated the feasibility of using WS-BPEL as an implementation technique for fault tolerant web services. The mapping of various fault tolerance patterns to WS-BPEL was presented.

In this paper, the proposed method has to be developed by consolidating the related papers and also improve with more special features.

3. METHODOLOGY

3.1 Proposed Work

The summary of the proposed work is discussed below: OOAD is the technique to measure the quality of the software using different OOAD metrics. OOAD metrics are of numerous to measure the software in different ways. In this proposed work, it is to be developed with new OOAD metrics to determine the quality and error rate of the software to enhance the quality of the software.

To automate the testing on the Object-Oriented Design by OOAD, OOAD Metrics have to be proposed in this paper. These metrics is constructed by taking or collecting large volume of data in order to provide the metrics suitable for all kinds of Object-Oriented languages such as Java, C#.Net, VB.Net, or C++. This OOAD metrics describes about the analysis design that is to be carried out in an object-oriented language.

Based upon the OOAD metrics, the quality of the software is evaluated. The proposed work provides the generalized methodology which will be termed as *QMGen (Generalized Quality Measurement) technique* on the object-oriented programming metrics and it is better to determine the program quality.

This OOAD metrics analysis the quality of the program by taking the program and then analysis the program by identifying the type of language used, number of classes used in the program and the type of the classes used in the program. All these things are provided in the form of Configuration files.

The configuration file is the file that contains all the necessary attributes about the metrics and it is most useful in order to determine the quality of the program. Almost all of the configurable items are provided in this configuration file. This configurable item includes the language used in the program, how many variables are used in the program and the type of the variable. Also the process of compilation is provided along with these configurable items.

Apart from these items, the configuration file also contains the information about the errors found in the program, type of the error and the error rate. The error is also classified into several categories based upon the type of the error. These items are stored as follows:

Class	a	b	c	...	x
Method	a1	b1	c1	...	x1
Identifiers	n1	n2	n3	...	Xn
Variables	v1	v2	v3	...	vn
.
.
.
Error type	e1	e2	e3	...	en

From this configuration files, almost everything about the program is stored by analysis the design. Based upon the error type, the error rate is determined and the quality of the program is calculated. The error rate is differentiated for different types of errors such as syntax error, run-time error and so on. Some of the error rates are defined below:

Syntax error	5%
Run-time error	7%
Data-type mismatch	3%
Compilation error	8%
Executable files	10%
Other errors	2%

The configuration file is built as follows with all the necessary attributes. These configuration file consists of three sections such as <compile>, <OOAD> and <error>.

- The <compile> section is used for compiling the programming file.
- The <OOAD> section is used to analysis the design of the program whereas;
- The <error> section is used to determine the error rate found in the program.

All these sections consist of generalized sub-tags to identify the quality of the program.

From this configuration file, the quality of the program is analysed based upon the OOAD metrics. When the program is inputted to this methodology, the information is retrieved from the configuration file and matches it with the inputted program.

All the metrics are fetching from the program and the level of inheritance is also determined. Upon completing these actions, the error rate is determined and the quality of the program is measured. The sample configuration file is shown below:

```

<configuration-file>
<config>
  <compile>
    <program-type> Java
      <cmd> javac </cmd>
    </program-type>
    <program-type> C#
      <cmd> csc </cmd>
    </program-type>
    <program-type> VB
      <cmd> vbc </cmd>
    </program-type>
  </compile>
  <OOAD>
    <Class-Number>.....</Class-Number>
    <Class-Type> .....</Class-Type>
    <Method-Count> .....</Method-Count>
    <Identifiers> .....</Identifiers>
    <Variables> .....</Variables>
    <Data-type> .....</Data-type>
    <Access-Specifiers> .....</Access-Specifiers>
  </OOAD>
  <error>
    <error-type> Syntax Error
      <error-rate> 5 </error-rate>
    </error-type>
    <error-type> Run-Time Error
      <error-rate> 7 </error-rate>
    </error-type>
    <error-type> Data type Mismatch
      <error-rate> 3 </error-rate>
    </error-type>
    <error-type> Compilation Error
      <error-rate> 8 </error-rate>
    </error-type>
    <error-type> Executable Error
      <error-rate> 10 </error-rate>
    </error-type>
    <error-type> Other Errors
      <error-rate> 2 </error-rate>
    </error-type>
  </error>
</config>
</configuration-file>

```

This proposed methodology also consists of an algorithm to determine the quality of the program. The algorithm is given below with proper explanation.

3.2 QMGen Algorithm

```

Create the Configuration file
Include the compilation procedure, OOAD Metrics,
and Error Detection
Get the Inputted Program
Analysis the Design of the program using OOAD
Metrics in the Configuration File
Detect the kind of program and compile the
program using Compilation Procedure
Get the results of the Compilation
Analysis the Error report using Configuration File
If error occurred then
Identify the error type and deduce the error rate
End if
If no error || error < threshold value then
Quality = "Good"
Else
Based upon the error rate, the quality of the
program is determined
End if
    
```

3.3 Algorithm Explanation

The algorithm process as follows: the initial step of this methodology is to create the configuration file with all the necessary attributes including Compilation Procedure, OOAD Metrics and Error Information. The next step is to get the inputted program from the user and then to analysis the quality of the program using OOAD Metrics. The quality of the program can be determined by compiling the program using the compilation procedure. The result of the compilation procedure is analyzed based upon the compilation result. If the compilation result shows "successfully compiled", then there is no error occurred and then the quality of the program is determined to be good. Otherwise, based upon the error rate occurred, the quality of the program is analyzed. Thus the OOAD Metrics is used to determine the program quality.

4. EXPERIMENTAL SETUP

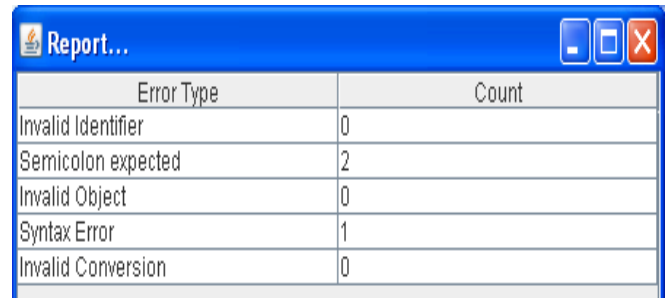
The Proposed methodology is very efficient that is used to evaluate the quality of the software. To test the efficiency of the methodology, various experimental setups are constructed and the result is analyzed. The experimental setup is made by taking the set of users with two categories such as, with programming skills and without programming skills. The user with programming skills can able to analyze the type of program and to compile the program. Then they use the configuration file to analysis the OOAD Metrics and to deduce the error rate.

Also, for the users with no programming skills, can execute the process by automate the process of analyzing the quality of software by using the configuration file.

Finally, the result is submitted to the user announcing the quality of the software and the usability of the software is determined by the user based upon the error rate.

Thus this proposed methodology provides better experimental results when it is implemented on any type of programming software. Thus it is very useful for all kinds of users to analysis the quality of software. The following table-1 shows the output of the sample program developed using this proposed methodology to evaluate the error rate in a given program.

Table-1: Sample Output



Error Type	Count
Invalid Identifier	0
Semicolon expected	2
Invalid Object	0
Syntax Error	1
Invalid Conversion	0

5. RESULTS AND DISCUSSION

Based on the experimental setup, various results have been identified. Also to ensure the quality of the proposed methodology, many comparisons have been undertaken with the existing techniques to analysis the rate of performance. Some of the comparison result is discussed below in table-2 and the comparison chart is shown in figure-1:

Table-2: Performance Comparison

Quality Measurement Methods	Rate of Performance (%)
CMMI (Capability Maturity Model Integrated)	85
TQM (Total Quality Management)	89
Liskov Substitution	80
DFT (Design For Test)	78
QMGen (Generalized Quality Measurement)	95

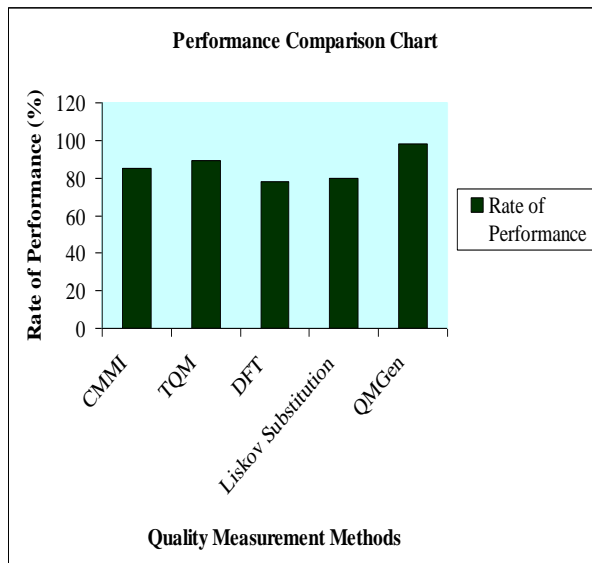


Figure-1: Performance comparison Chart

6. CONCLUSION

Thus the quality of the software is measured and the result is automated by the proposed methodology of this research. This methodology provides an efficient and well-suited mechanism to measure the quality of the software. This measurement is made by analysing the design using the configuration file provided by the research work. Various OOAD metrics are proposed along with the compilation procedure for different kind of languages and the various error mechanisms. Thus this research work completed successfully with efficient methodology proposed in this work.

7. REFERENCES

- [1] Claire Le Goues and Westley Weimer “Measuring Code Quality to Improve Specification Mining”, IEEE Transactions on Software Engineering, vol.38,no.1,pp. 175-190, January 2012.
- [2] Nikos Anerousis, Yixin Diao, Aliza Heching, “The Cost of Service Quality in IT Outsourcing”, IEEE IM 2011 , pp. 773-784, May 2011.
- [3] Bandar Alshammari, Colin Fidge and Diane Corney, “A Hierarchical Security Assessment Model for Object-Oriented Programs”, Faculty of Science and Technology, Queensland University of Technology, Australia, 11th International Conference on Software Quality, pp. 218-227, May 2011.
- [4] Saad Alahmari, Ed Zaluska, David C De Roure, School of Electronics and Computer Science University Southampton, UK, “A Metrics Framework for Evaluating SOA Service Granularity”, International Conference on Service Computing, pp. 512-519, Jul 2011.
- [5] Vishwajit Joshi, Consultant, Process and Innovation Performance, Accenture Management Consulting, Mumbai, India, “Metrics Center of Excellence, From idea to implementation of a “meaningful” measurement and analysis process”, IEEE on Joint Conference, pp. 133-141, Nov 2011.
- [6] Douglas C. Schmidt, Adam Porter, “Leveraging Open-Source Communities To Improve the Quality & Performance of Open-Source Software”, Electrical & Computer Engineering Department Computer Science Department, University of California, Irvine University of Maryland, May 2011
- [7] Subhachandra Chandra and Peter, M. Chen, Computer Science and Engineering Division, Department of Electrical Engineering and Computer Science, University of Michigan, “Whither Generic Recovery from Application Faults? A Fault Study using Open-Source Software”, pp. 97-106, June 2000.
- [8] V.N. Nair, D.A. James, W.K. Erlich, and J. Zevallos, “A Statistical Assessment of Some Software Testing Strategies and Application of Experimental Design Techniques,” Statistica Sinica, vol. 8, no. 1, pp. 165-184, 1998.
- [9] D.R. Wallace and D.R. Kuhn, “Failure Modes in Medical Device Software: An Analysis of 15 Years of Recall Data,” Int’l J. Reliability, Quality and Safety Eng., vol. 8, no. 4, August 2001.
- [10] D.R. Kuhn and M.J. Reilly, “An Investigation of the Applicability of Design of Experiments to Software Testing,” Proc. 27th NASA/IEEE Software Eng. Workshop, Dec. 2002.
- [11] D. Richard Kuhn, Senior Member, IEEE, Dolores R. Wallace, Member, IEEE Computer Society, and Albert M. Gallo Jr., “Software Fault Interactions and Implications for Software Testing” June 2004.
- [12] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, Timothy S. Woodall Innovative Computing Laboratory, University of Tennessee, Open System Laboratory, Indiana University, “Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation”. Sep 2004.
- [13] Robyn R. Lutz, Jet Propulsion Laboratory, “Software Engineering for Safety: A Roadmap”, pp.213-226 June 2000.
- [14] Rudolf Ferenc, István Siket and Tibor Gyimóthy, University of Szeged, Department of Software Engineering, “Extracting Facts from Open Source Software”, pp 60-69, September 2004.