TECHNIUM

Technium Vol. 3, Issue 3 pp.82-99 (2021)
ISSN: 2668-778X
www.techniumscience.com

Quality Metrics measurement for Hybrid Systems (Aspect Oriented Programming – Object Oriented Programming)

Mazen Ismaeel Ghareb[1,*], Gary Allen[2]

[1]College of Science and Technology, University of Human Development, Iraq - E-mail: mazen.ismaeel@uhd.edu.iq

[1]College of Computing and Engineering, University of Huddersfiled, UK - E-mail: mazen.ismaeel@hud.ac.uk

[2]College of Computing and Engineering, University of Huddersfiled, UK - E-mail: g.allen @hud.ac.uk

* Corresponding author: College of Science and Engineering , University of Human development , Kurdistan Region, Iraq. Tel: +9647705310098.

Abstract.
This paper explores a new framework for calculating hybrid system metrics using so ftware quality metrics aspect-oriented and object-oriented programming. Software m etrics for qualitative and quantitative measurement is a mix of static and dynamic sof tware metrics. It is noticed from the literature survey that to date, most of the archite cture considered only the evaluation focused on static metrics for aspect-oriented app lications. In our work, we mainly discussed the collection of static parameters , long with AspectJ-specific dynamic software metrics.The structure may provide a new di rection for research while predicting software attributes because earlier dynamic met rics were ignored when evaluating quality attributes such as maintainability, reliabilit y, and understandability of Asepect Oriented software. Dynamic metrics based on th e fundamentals of software engineering are equally crucial for software analysis as are static metrics. A similar concept is borrowed with the introduction of dynamic so ftware metrics to implement aspect-oriented software development.Currently, we onl y propose a structure and model using static and dynamic parameters to test the aspe ct-oriented method, but we still need to validate the proposed approach.

Keywords: Aspect-Oriented Programming Metrics, Object-Oriented Metrics, Static Metrics, Dynamic Metrics, Software Quality metrics, Quality framework Measurement, Hybrid system metrics, hybrid application.

1. INTRODUCTION

Aspect-oriented programming (AOP) may be a fairly new method that has been argued for enabling higher modularization of crosscutting concerns [1] and thus hastening the event process. The findings are that well-defined problems are easier to manage, Modified, and established that the total programmer's work time should be shorter than the analog system's event time, realized without the mechanisms provided by AOP Validating such theories calls for observational studies. This paper presents the results of preliminary empirical evaluation of the software quality AOP metrics on software AO/OO hybrid software quality. Three experienced programmers, with three open AO/OO projects, using a script to extract AOP and the Object-Oriented Programming (OOP) metric approach and reflect on new proposal quality framework. The paper includes a comparison of the developed systems AOP and OOP,Chidamber and Kemerer (hereafter CK)[2], Distance from the most sequence metric proposed by Chidamber and Kemerer Martin[3], out-of-code standard metric (defined as acceptance variety Tests passed)[4,5,6], and productivity metrics for programmers. CK Technology metrics[2] adapted to new aspect-oriented properties [7][8][9][10].

The research [9] stated that metrics for object-oriented software development is limited, and empirical Evidence, linking the object-oriented methodology and venture consequences is scarce [11]. Even scarcer is anecdotal Evidence of the effect of aspect-oriented programming on software format quality, or development efficiency metrics. Therefore, the purpose of this paper is to this gap and supplies empirically Evidence of the impact of aspect-oriented programming on software development affectivity and graph exceptional metrics, as plan aspects are

Extremely essential to produce remarkable software program [11]. The hypothesis hat graph excellent metrics are proper predictors of the fault proneness are supported in [12] and [13]. According to [14], amongst the claimed benefits of this new technology is the reduction in the amount of code written and greater cohesion. As with any new technology, aspect-oriented technology has both advantages and costs. Quality of the AO system could also be measured with the help of internal characteristics like brotherly love [15], coupling [16] and complexity [17]. Researchers and practitioners have suggested measures and templates to live the consistency of the operating systems. Software metrics may be a qualitative indicator of any package attribute, and mod-el specifies the relationship between those metrics. McCall et al. in the late 1970s [17] and in the late 1980s Boehm [18] introduced quality models for software that developed the concept for project managers to identify risks and determine quality. As object-oriented programming achieved a reasonable degree of sophistication, ISO / IEC 9126[19], software design requirements were introduced in the early nineties, which promoted maintenance. The inheritance property of object-oriented programming put the focus on reuse, Dromey [20] introduced reusability to the six features as present in ISO / IEC 9126. IEEE also took the lead of releasing the Software Development Knowledge Body[11], which is the first significant effort to compile, define, and standardize the terminology models, methods and techniques. Here are many other scholars and practitioners [21] [22] [23][24] who have issued recommendations for the living standard of information systems of various varieties.

The paper starts by introducing the relevant area add in Section 2. Section 3 includes an overview of the framework being studied, including consumer specifications. Section 4 specifies the research aims, the inquiries to be addressed and the metrics to be obtained in accordance with the goal-question-metric (GQM) approach [25]. Research findings are summarized in Section 5, with a conclusion Section 6. The paper's conclusions are laid out in Section 7.

2. RELATED WORK

No associated research focuses on the systematic assessment of aspect-oriented solutions[26]. In addition to discussing the impact of aspects on object-oriented programming activities, Kersten and Murphy[27] defined other laws and policies used to ensure consistency and modifiability. Walker u.a. Provided initial observations into how aspect-oriented programming is useful and accessible. Soares et al.[28] stated that the Web-based system's implementation of AspectJ has significant advantages over the corresponding pure Java implementation. Garcia et al.[29] requested a quantitative report, conceived Comparing the management and reuse of pattern-oriented solution assistance versus an aspect-oriented multi-agent framework method. It contends that the aspect-oriented approach facilitated the creation of the investigated framework with better crosscutting modularization unique Concerns. The usage of aspects have culminated in more excellent isolation of crosscutting problems, a reduced (though less cohesive) coupling, and fewer lines of code. Tsang et al. [30] evaluated the effectiveness of AOP for the separation of concerns. They applied the CK metrics suite to assess and compare an aspect-oriented and object-oriented real-time system in terms of system properties. They found improved modularity of aspect-oriented system over the object-oriented method, indicated by the reduction in coupling and lack of cohesion values of the CK metrics.

Garcia et al. [31], have developed systematic studies that investigated the employment of aspect-oriented programming to implement classical design patterns. It's worth mentioning that Tonella and Ceccato [32] performed an empirical assessment of refactoring the aspectizable interfaces. This study shows that the Aspect able moves Interfaces have a small effect on the major scale of the decomposition, but they generate an increase in the modularity of the application at the same time. Modularization of the design of the crosscutting interfaces simply simplifies the interpretation of the source code from the standpoint of established consistency attributes [33][34]. ISO / IEC 9126 discusses three facets of software consistency: I consistency of systems, (ii) content of goods, and (iii) quality of products in operation. ISO / IEC 9126 series specifications implemented a hierarchical paradigm of six main consistency features. Such design attributes are grouped into 21 sub-characteristics that relate to the nature of the interior. ISO / IEC 9126-1 is mainly concerned with

identifying standard characteristics and sub-characteristics within the end goods. ISO / IEC 9126-2 includes external measures described in ISO / IEC 9126-1 for the calculation of attributes of six external quality assets.

ISO / IEC 9126-2 identify external measurements; ISO / IEC 9126-3 defines internal measures and ISO / IEC 9126-4 describes standard indicators in use for consistency or sub-characteristic calculation. Internal metrics measure software itself; external metrics measure computer-based system behavior that includes software and software quality measurement results of using the software in a particular context of use [18].

Quality Type	Characteristics	Sub-characteristics
Software Product Quality	Functionality	Suitability
		Accuracy
		Interoperability
		Compliance
		Security
	Reliability	Maturity
		Fault tolerance
		Recoverability
	Usability	Understandability
		Learn-ability
		Operability
	Efficiency	Time behavior
		Resource behavior
	Maintainability	Analyzability
		Changeability
		Stability
		Testability
	Portability	Adaptability
		Install-ability
		Replace-ability
		Conformance

Figure 1 Internal metrics measure software ISO / IEC 9126

Empirical study

Three hybrid projects that have **b**een developed using AO/OO and a java program to extract the static metrics for Aspect oriented programming and object-oriented programming. an plug CodeMR in used to extract Object-Oriented metrics than a combination of two metrics reflected for hybrid application system.

2.1 AO/OO open source projects

2.1.1 GTalkWAP This application is used for accessing Google chat service using WAP enable access. The application programmed by java, AspectJ languages [35]

2.1.2 AspectJ Hot Draw project is an open-source project developed with Aspect-Oriented refactoring of the two-dimensional graphics system within JHotDraw. TestJHotDraw is built as a subproject for research designed to ensure behavior maintenance between the two solution

These two projects have been an Object-Oriented Programming/Object Oriented Programming AO/OO hybrid system, which developed with both language techniques. Table 1 below shows the AO/OO properties of these projects, how many java files and aspect files it contains. These values indicate that AOP affects all software development processes, the quality metrics, and other related metrics of these projects.

Table 1 AO/OO project Properties

No.	Project Name	*.java Line of Code (Java File) wc -l `find . -name *.java -print`	*.aj (Aspect file) wc -l `find . -name *.aj -print`
1	Google Talk	25462	1958
2	Ajhotdraw	41594	2579

2.2 Restrictions

There is not any plug-in that can measure both AO/OO metrics together and draw the metrics calculation for them as one package in the project. There is difficulty in reading from the file extension. AJ aspect file in the projects.

3. Measurements

The analytical analysis was conducted with the mistreatment of the GQM technique, defined in [36], to illustrate measurements on the package that comes in. There had been defined goals. The initial one relates to the quality of the code (in terms of its modularity and Size), and hence the different considerations proposing new quality framework for the hybrid application system.

Goal 1: The AOP impact measurement on application quality.

26 Question 1: How does AOP make the modularity of a program an object?

Question 2: Which impact does AOP have on the scale of a system?

Goal 2:The AOP metrics impact on AO/OO software quality ?

Question 1: can the proposing quality metrics derived from ISO/9126 software will promising new era of measuring hybrid application software.

Goal 1 question 1 :Metrics: metrics Dn (Distance) have the answer to this problem CBM (Coupling Between Modules), in the central chain, RFM (Modul Response) and LCO. (Failure to Cohere in Operations).

Goal 1 Question 2 : The LCO. parameter shows modules which are not Cohesive, that is, they know a feature that should be separated, and thus the LCO. Metric is an approximate measure of modularity.

Metrics: Metrics that indicate a system's Size are: NCLOC (Non-Comment Lines Of Code), NOM (Number Of Modules), and WOM (Weighted Operations in Module).If the system is smaller (in terms of number of source code lines and modules), it is easier to grasp the programmer's intentions and comprehend the full functionality. The WOM metric counts operations in a module (Class, interface or Aspect) and together with the NOM metric indicates its Size.

3.1.1 GTalkWAP

This application is utilized for getting to Google chat benefit using WAP empower to. The application modified by java, AspectJ dialects [34]. CodeMr Plug-in has been utilized as an open-source instrument to extricate OOP measurements. AOP measurements it has been gotten using shell script and specific program created by java dialect. The CodeMR has mostly gotten AOP measurements, for occurrence, complexity, measure, coupling, and cohesion. The plug-in CodeMr has been graphical sees of all highlights of OOP measurements subtle elements because it has appeared in Figure 1 to 4. The Green Color meaning this metric is having moo values; the light green implies low-medium, Orange color meaning Tall values, and Ruddy color, meaning exceptionally high. Figure1 appears the values of Complexity, coupling, Need of Cohesion, and Estimate. The measure is %47 low-medium, and the complexity is %18 is low-medium, and other values are moo. The other two measurements have lower values, and the device has the correct quality measurements. The impediments are not perusing.

Technium Vol. 3, Issue 3 pp.82-99 (2021)
ISSN: 2668-778X
www.techniumscience.com

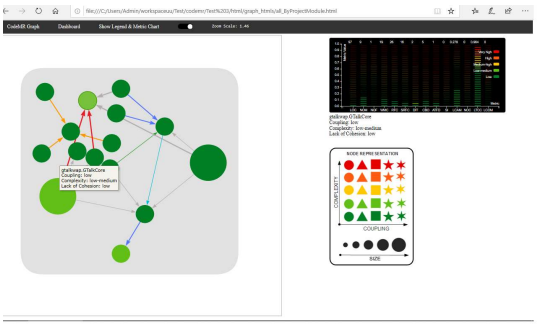

Figure 2 General Information about Metrics for Google Talk project

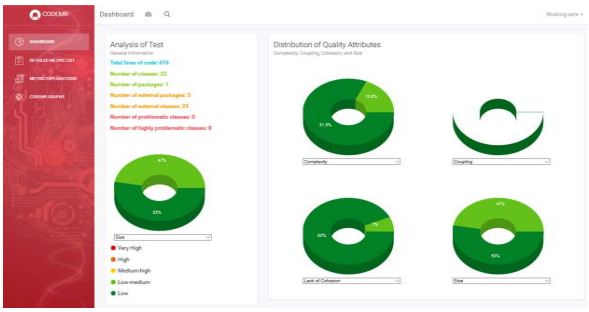

Figure3 coupling and Cohesion values regarding packages Gtalk Project

Figure 2 appears all the values of measurements employing a bar chart and appears the connections between all bundles. It seems the estimate of the bundle spoken to by circle measure, and the colors speak to the need of cohesion and coupling because it is performed within the hub representation charts. The CodeMr plug-in shows the module relationships between all modules. The values of metrics have represented by different colors from low to high (Dark green, Light green, yellow, orange, and red) colors, which are shown in the bar chart, as shown in Figure 4. Figure 4 displays all the information for metric values for each Java class in the project. The findings indicate the benefits of the project have standard benefits for a Google talk projct. Addingmore, the effects show all the Classes except the aspect .aj extension.

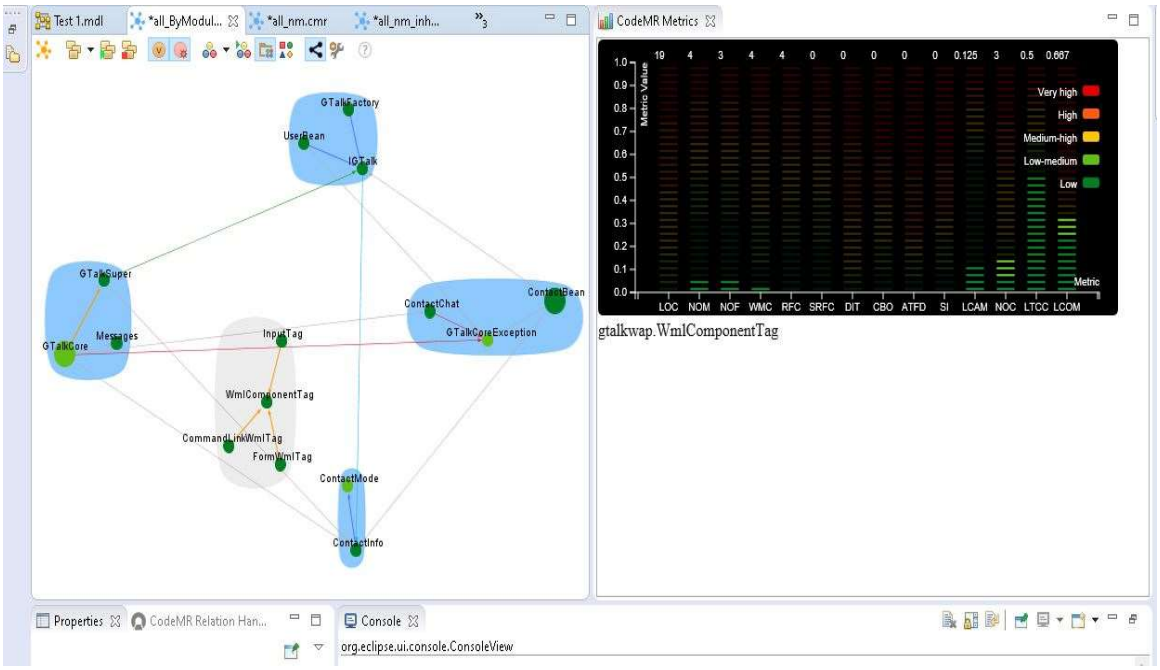

Figure 4 Metrics, values regarding Modularity Gtalk Project

List of all classes (#22)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
1	GTalkCore					97	low-medium	low	low	low-medium
2	ContactMode					18	low-medium	low	low	low
3	GTalkCoreException					10	low-medium	low	low	low
4	CommandLinkWmlRen...					81	low	low	low	low-medium
5	Tags					76	low	low	low	low-medium
6	ContactBean					64	low	low	low	low-medium
7	ContactInfo					47	low	low	low-medium	low
8	UserBean					43	low	low	low	low
9	RosterRenderer					42	low	low	low	low
10	GTalkSuper					33	low	low	low	low
11	InputRenderer					27	low	low	low	low
12	InputTag					24	low	low	low	low
13	ContactChat					21	low	low	low	low
14	DataRosterTag					20	low	low	low	low
15	WmlComponentTag					19	low	low	low	low
16	IGTalk					14	low	low	low	low
17	CommandLinkWmlTag					12	low	low	low	low
18	ActionMethodBinding					11	low	low	low	low
19	Messages					10	low	low	low	low
20	FormWmlRenderer					9	low	low	low	low
21	FormWmlTag					5	low	low	low	low
22	GTalkFactory					4	low	low	low	low

Figure 5 classes Metrics Details Gtalk Project

This project has twenty-four java and 4 aspect files with extension .java and .aj consequently. In Section 3 it mentioned how extracting the road of code metrics for this project employing a bash script. This project considers a little scheme because it contains a twenty-four class file and four aspect files. Line of code metrics shows the share of Aspect is 0.07% from the entire Line of Code because it shows in table 1. The portion of the road of the Code of Aspect is little. However, it's a significant impact on modularity metrics and other metrics, as explained in table 2 and table 3.

Table 2 OOP Metrics and typical AOP Metrics GTalkWAP project

Project Name												
	Complexity				Coupling			Cohesion				
GTalkWAP												
	W.M.C. Weighted Method Count	D.I.T. Depth of Inheritance Tree	RFC. Response For a Class	SI Specialization Index	NOC (Number of Children)	CBO. (Coupling Between Object Classes)	ATFD (Access to Foreign Data)	LCOM (Lack of Cohesion of Methods)	LCAM (Lack of Cohesion Among Methods(1-CAM))	LTCC (Lack Of Tight Class Cohesion (1-TCC))	LOC. Line Of Code	NOF(Number of Fields)
Average Values	6.5	0.909	3.90909 1	0.0	0.181	0.090	0	0.225	0.157	0.315	42.875	1.136
Total Values for each factors	11.318				0.271			0.697				

Technium Vol. 3, Issue 3 pp.82-99 (2021)
ISSN: 2668-778X
www.techniumscience.com

Table 3 AOP Metrics GTalkWAP project

	Size						Complexity				Coupling			Cohesion		Modularity	
No.	Project Name	Number of Class files	Number of Aspects files	Line of code aspect	Number pointcuts	No. Of Aspects	WMC: Weighted methods per Class	DIT: Depth of inheritance Tree	RFC: Response for a Class	Cyclomatic Complexity of Aspect CCA (Advice, Pointcut,	CBO: Coupling between object classes	NOC: Number of Children	Coupling on Advice Execution (CAE)	LCOM: Lack of cohesion in methods	Lack of Concern-based Cohesion (LCC)	Concern Diffusion over Components (CDC)	Crosscutting Degree of an Aspect (CDA)
1	Google Talk	24	4	69.928	8	2	6.5	1.7	3.9	26	0.09	0.18	0	0.22	2	6	5
Total Values for each factor	107.928						38.1				0.27			2.22		11	
Normalize values for each factor [0-1] X new = (X – X min) / (X max – X min)	1						0.351				0.0			0.02		0.10	

3.1.1.1 AOP metrics extraction :

Crosscutting Degree of an Aspect (CDA) metrics: In the project Gtalk, it has two aspects that contain eight pointcuts, which affects 40 methods in 4 modules.
CDA = Total of Effected Methods / Total of the pointcut
= 40/8 = 5
The values of CDA are five, which indicates the high values of CDA in the Gtalk project. Another indication is calculating CDA/WMC =5/6.5, which is equal to 0.76 and indicates advice effect several modules.

Table 4 java program AOP metrics Extraction for GtalkWAP project

Coupling on Advice Execution (CAE) Metrics: In the project Gtalk, it has a 0 joinpoint defines in Aspects. CAE indicates that the modules not effected by Aspects regarding coupling. The second indication is calculating CAE/WOM =0/6.5, which is equal to 0, which means there are high values of calling methods. Still, there is no joint point and Advice defined in the Aspect to demonstrate the effect of coupling on the modules.Concern Diffusion over Components (CDC) metrics: the following equation can calculate CDA metrics of Gtakproject

C.D.C. = number of components that assist in the implementation of a concern + number of components that access the components. The value of CDC of this project is six which means is have high values of CDC among the project, and there are 24 java classes and four aspect files if we take an average effect overall the project this will be

CDC Effect = CDC values /Total number of Classes and Aspect

=6/28

=0.214

Lack of Concern-based Cohesion (LCC) metrics: The project Gtalk has two nun functional concerns of Lack of Concern-based Cohesion (LCC) in this project. It contains four aspect files. These projects have high values of LC C; this indication means this project has a cohesive solution in concerns implementation.

The Aspect complexity (AC) metrics: The Cyclomatic complexity of the project Gtalk has eight pointcuts 4 of them have the complexity of 3 because we have count one for the name of pointcuts and one for the wildcard and one for calling the function (3*4). The other four pointcuts are more complex, and the Cyclomatic complexity will be more complex because it contains wildcard in their signature and || operator, which counted as four for each calling specific function (4*4) so the total complexity of this project equal to 26.

In table 3 below, it shows all the details of AOP metrics for project Gtalk. The normalization values range from 0 to 1. Using these formula X new = (X – X min) / (X max – X min). The maximum values are Size, which is equal to 1. AOP metrics for complexity are equal to 0.351, which indicates the small value; however, compared to the Size of the AOP is consider a complex project. The project has low values of coupling; this means the good structure of this project. The low cohesion of this project showed the project support high readability and maintainability. The modularity values are 0.1, which indicates the system has a profound connection between modules in the system. Modularity metrics values indicate how many modules have independent on another module in the system, as it shows in table 3.

3.1.2 AspectJ HotDraw

It is an open-source code project created with Aspect-Oriented refactoring of the framework of the two-dimensional design inside JHotDraw. TestJHotDraw is worked as a subproject for research intended to guarantee conduct upkeep between the two arrangements [37]. The codeMR has in part separated AOP measurements. For example, multifaceted nature, Size, coupling, and attachment. The module CodeMr has been graphical shows include all OOP measurement subtleties, as it shows in Figure 5 to 8. The Green Color meaning this measurement is having low qualities; the light green methods low-medium, Orange shading indicating High qualities, and Red shading meaning high. In the venture, AjHotdraw shows a typical appropriation for the majority of bundles and classes. Figure 5 shows the floating over the Class that has high attachment and low-medium qualities. Different circles speak to the class size and associations with different Classes in the task.

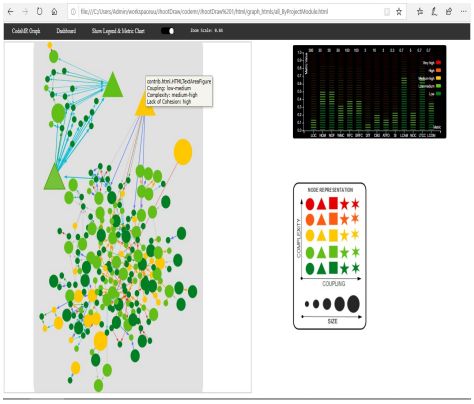

Figure 5 AjHotdraw project metrics

Figure 6 shows the overall quality metric dispersion spoke to by pie graphs. Figure 19 shows % 20.9 medium-high complexities, %19 of High Attachment, and 12.6% gigantic size. Quality measurements show that there are a few issues like this venture.

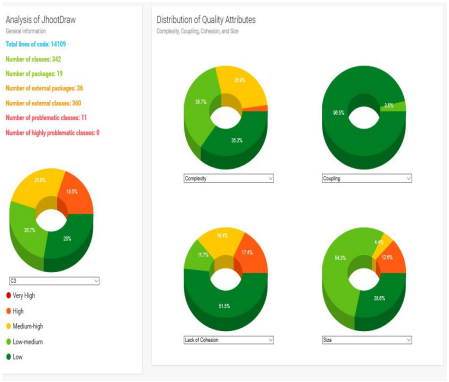

Figure 6 General Quality Metrics Distribution AjHotdraw project

Figure 7 shows the estimations of attachment with high sums conveyed among all bundles in the task. Union Metrics demonstrates this task needs to improve in a portion of their bundles to diminish the top qualities.

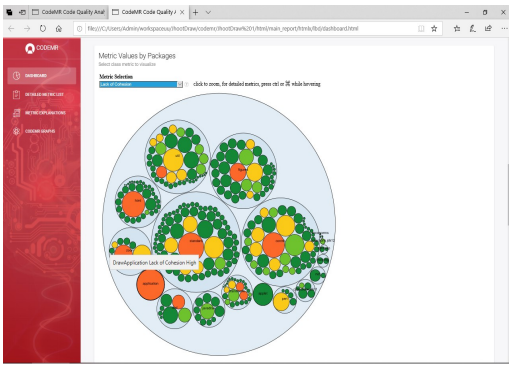

Figure 7 Quality Metrics by Packages AjHotdraw project

Technium Vol. 3, Issue 3 pp.82-99 (2021)
ISSN: 2668-778X
www.techniumscience.com

Table 5 Table 4 OOP Metrics and some AOP Metrics using CodeMr

	Complexity				Coupling			Cohesion			Size					
AjHotdraw																
	WMC Weighted Method Count	DIT Depth of Inheritance Tree	RFC Response For a Class	SI Specialization Index	NOC (Number of Children)	CBO (Coupling Between Object Classes)	ATFD (Access to Foreign Data)	LCOM (Lack of Cohesion of Methods)	LCAM (Lack of Cohesion Among Methods(1-CAM))	LTCC (Lack Of Tight Class Cohesion (1-TCC))	LOC Line Of Code	NOF (Number of Fields)	NOSM (Number of Static Methods)	NOM (Number of Methods)	NOSF (Number of Static Fields)	NORM (Number of Overridden Methods)
Average Values	22.274	1.532	11.913	0.172	0.419	0.852	0.017	0.229	0.345	0.358	97.245	1.142	0.216	6.497	0.593	0.324
Total Values for each factor	35.891				1.288			0.932			106.017					
Normalize values for each factor [0-1] X new = (X – X min) / (X max – X min)	0.33				0			0.00			1					

This task AjHotdraw has 291 java records, and 31 angles document with expansion .java and .aj like this. In Section 3.1 it referenced how separating the line of code measurements for this task utilizing a slam content. This venture considers a medium task size since it contains a 291 class document and 31 viewpoint records. Line of code measurements shows the level of perspective is 0.3% from the total Line of Code, as it shows in table 4. The part of the line of code of Aspect has a little rate. Be that as it may, it significantly affects measured quality measurements and different measurements, as clarified in table 5.

Technium Vol. 3, Issue 3 pp.82-99 (2021)
ISSN: 2668-778X
www.techniumscience.com

Table 6 OOP Metrics and some AOP Metrics using CodeMr plug-in for AjHotdraw project

	Size						Complexity				Coupling			Cohesion		Modularity	
No.	Project Name	Number of Class files	Number of Aspects files	Line of code aspect	Number Pointcut	No. Of Aspects	WMC: Weighted methods per class	DIT: Depth of Inheritance Tree	RFC: Response for a Class	Cyclomatic Complexity of Aspect CCA (Advice, Pointcut, JoinPoint)	CBO: Coupling between object classes	NOC: Number of Children	Coupling on Advice	LCOM: Lack of cohesion in methods	Lack of Concern-based Cohesion (LCC)	Concern Diffusion over Components (CDC)	Crosscutting Degree of an Aspect (CDA)
2	AjHotdraw	291	31	83.193	37	53	22.27	1.53	11.93	282	0.85	0.4	12	0.23	10	150	4.86
Total Values for each factor	495.193						317.73				13.25			10.23		154.86	
Normalize values for each factor [0-1]	1						0.634				0.006			0		0.298	

3.1.2.1 **AOP metrics extraction:**

Crosscutting Degree of an Aspect (CDA) metrics: In an AjHotdraw project, many Aspects is 53, number of pointcut 37 that have called 180 methods, so the CDA is CDA = Total of Effected Methods / Total of the pointcut

=180/37

=4.86

Another indication is calculating CDA/WMC =4.86/22.7, which is equal to 0.21 and indicates advice effect several modules. AjHotdraw shows high values of the CDA, which indicate many modules affected by aspects of pointcut and intertype declaration.

Coupling on Advice Execution (CAE) Metrics: in the AjHotdraw project, many Advice is 12; this Advice will locking for joinpoint and run it, adding to that it depends on the type of execution of the method before, after or around Advice. These results show that there are nine joint points. Another indication is calculating CAE/WOM =12/22.7, which is equal to 0.52, which means there are high values of calling methods with calling nine joinpoints and 12 advice defined in Aspect. This demonstrates the effect of coupling on the base code for project modules.

Concern Diffusion over Components (CDC) Metric: The project AjHotdraw, we can calculate

the CDC = number of components that assist in the implementation of a concern + number of components that access the components. The value of CDC of this project is 150 which means is have high values of CDC among the project; there are 291 java classes and 31 aspect files if we take an average effect overall the project this will be

CDC Effect = CDC values /Total number of Classes and Aspect

=150/322

=0.46

Lack of Concern-based Cohesion (LCC) Metrics: LCC. Metrics is calculated by counting all Concerns. LCC values are five for the Commands package. These concerns are three non-functional and two concerns with an interface which responsible for add and Remove Commands in GUI. Figure package has five functional Concerns to read and write on Figure attributes and adding other functionalities. An LCC value of AspectJHotDraw project is equal to ten .LCC metrics for this project is considered high amounts, which is means the project has cohesive aspect concerns

Aspect complexity of (AC) Metrics: The Aspect complexity AC of the project AjHotdraw has 37 pointcuts. The package figures it has the complexity of 46. AC includes the complexity of Aspect and pointcuts the have more complex pointcuts because it uses in their signature wildcards and && operator, nested pointcuts, and keyword before, after, and around the execution of the methods. The package Undo 10 CCA metrics value. The package commands have 235 CCA values, which considered very involved in pointcuts and intertype declaration and Advice. As it mansions in the section above, the Advice can be measured the same as methods and contains contralto statements and iteration, which make it complicated. The pointcuts are more involved in all other packages because they have very complex signature with wild cards, ||, &&, methods calls. So the total values of CCA are 282, which means this project has more significant benefits of the Cyclomatic complexity of Aspect.

n table 6 below, it shows all the details of AOP metrics for project AjHotdraw. The normalization values range is between 0 to 1. Using these formula X new = (X – X min) / (X max – X min). The maximum values are Size, which is equal to 1. AOP metrics for complexity is equal to 0.634, which indicates the high value, which is considered a complex project. The project has low values of coupling 0.006; this means the good structure of this project. The average cohesion values of this project showed the project has less supporting high readability and maintainability. The modularity values are 0.298, which indicates the system has a good connection between modules in the system. It shows how many modules have independent of another module in the system

4. Summary

4.1 **Gtalk application analysis** :

To sum up the outcome, the separated measurements were done to extricate the remaining AOP measurements through two CodeMr procedures to expel OOP and some of AOP measurements and java programs. Therefore, Tables 2 and 3 have specific outcomes. The mix of these tests generally demonstrated a similar result. Table 6 shows the mix esteems normal execution for project Gtalk.

	Size	Complexity	Coupling	Cohesion
Average Values	0.56	0.30	0.0	0.015

In table 6 the typical qualities for OOP/AOP between the various measurements were explained. Of the considerable number of measures, the two tables have comparative outcomes. The measurement scale is comparable to 0.56, which speaks to 50 percent. These discoveries propose that Gtalkproject is a difficult task. The all-out qualities for intricacy measurements are equivalent to 30 percent. These outcomes showed that normal unpredictability of the source code OOP and AOP is available in this task. The lower multifaceted nature measurements help bring down the upkeep and testing costs. The coupling esteems incredibly frail, and the coupling for OOP/AOP measurements is higher. These discoveries recommend that low coupling suggests. The attachment implies that a piece of the source

code at one point in the code base is associated with one another. The particularity esteems 0.10, proposing a profound connection between the modules in the system. Estimations of particularity measurements show what number of modules in the framework have autonomous on another module.

4.2 **Ajhotdraw application analysis** :

The extricated measurements were done to separate the remaining AOP measurements through two CodeMr procedures to evacuate OOP and some of AOP measurements and java programs. This way, Tables 4 and 5 have specific outcomes. The blend of these tests generally indicated a similar result. Table 7 shows the mix esteems normal execution.

Table 7OOP/AOP metrics values for AjHotdraw project

	Size	Complexity	Coupling	Cohesion
Average Values	1	0.48	0.002	0.0

In table 7 the ordinary qualities for OOP/AOP between the various measurements were explained. Of the considerable number of measures, the two tables have comparable outcomes. The measurement Size is equal to 1, which speaks to the most extreme estimations of measurements. These discoveries recommend that AjHotdraw is a perplexing task as indicated by Size of base code. The complete qualities for unpredictability measurements are equivalent to 0.48. These outcomes exhibited that typical intricacy of the source code OOP and AOP is available in this venture. The lower multifaceted nature measurements help bring down the upkeep and testing costs. The coupling esteems small, which 0.02, and the coupling for OOP/AOP measurements is higher. These discoveries recommend that low coupling suggests the further division of the inconsequential source code. More top attachment esteems that piece of the source code at one point in the codebase are associated with one another; anyway, the union is not as much as a coupling in this undertaking, which was not a suggestion. The measured quality qualities are 0.29, recommending a profound connection between the modules in the system. Estimations of calculated quality measurements show what number of modules in the framework have free on another module. The AjHotdraw have less ordinary of seclusion measurements.

4.3 **Proposing Metrics for AO/OO:**

According to the two project metrics extraction and extraction, it is possible to propose new sub characteristics to the ISO/9126 quality model to measure the AOP metrics part. In table 8 contain these sub characteristics. It is essential to have these metrics and need to be evaluated. The statistics priorities of these sub characteristics have been calculated in sections 5.1 and 5.2 for quality metrics—however, more need to be assessed regarding the dynamic parameters for the sub characteristics.

Technium Vol. 3, Issue 3 pp.82-99 (2021)
ISSN: 2668-778X
www.techniumscience.com

Table 8 Proposed AOP Software Quality Model

Quality Type	Characteristics	Sub-C
Software Product Quality	Functionality	
		A
		Int
		C
		E
	Reliability	
		Fa
		Re
		C
	Usability	Und
		L
		C
	Efficiency	Ti
		Resc
		Service
		A
	Portability	In
		C
		Re
		C

The proposing perspective situated item quality structure will be concentrated on AOP part and AOP new measurements. These unique measurements concentrated on static estimation, not dynamic quality item measurements, since dynamic evaluation, it needs a group of engineers and item directors and clients. This examination focuses on 5 different ventures created with OO/AO code. The primary center was to show the impact on AOP measurements on programming item quality utilizing the proposed system.

5. Conclusion:

This paper explains the importance of measuring the AOP metrics in the OO/AOP application. Statistical measurement has been conducted regarding measuring statistics AO/OO metrics. The results showed there is a positive and significant impact on AOP metrics for hybrid application systems. Several parameters have been standard between these two techniques AOP/OOP. However, regarding dynamic metrics such as efficiency and other dynamic metrics, there is many works need to

be done with a large group of developer and using several projects. AOP ISO/9126 Framework has been adapted with proposing sub characteristics to the framework that needs to be added to rich the measurement of AO/OO hybrid applications from AOP metrics perspectives. More studies and experiments need to be conducted to find all metrics for these proposed characteristics.

References

[1] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M. and Irwin, J., 1997, June. Aspect-oriented programming. In European conference on object-oriented programming (pp. 220-242). Springer, Berlin, Heidelberg.

[2]Chidamber, S.R. and Kemerer, C.F., 1994. A metrics suite for object oriented design. IEEE Transactions on software engineering, 20(6), pp.476-493.

[3] Ghareb, M. I., & Allen, G. (2015). Identifying similar pattern of potential Aspect oriented functionalities in software development life cycle. Journal of Theoretical and Applied Information Technology, 80(3), 491-499.

[4] George, B. and Williams, L., 2003, March. An initial investigation of test driven development in industry. In Proceedings of the 2003 ACM symposium on Applied computing (pp. 1135-1139).

[5] Ghareb, M. and Allen, G., 2019. An empirical evaluation of metrics on aspect-oriented programs. UHD Journal of Science and Technology, 3(2), pp.74-86.

[6]George, B. and Williams, L., 2004. A structured experiment of test-driven development. Information and software Technology, 46(5), pp.337-342.

[7] Ghareb, M.I. and Allen, G., 2018, April. State of the art metrics for Aspect oriented programming. In AIP Conference Proceedings (Vol. 1952, No. 1, p. 020107). AIP Publishing LLC.

[8] Dhambri, Karim, Jean-Franois Gélinas, Salima Hassaine, and Guillaume Langelier. "Visualization-based Analysis of Quality for Aspect-Oriented Systems." In ACM international Conference. 2005.

[9] Laddad, R., 2003. Aspect-oriented programming will improve quality. IEEE software, 20(6), pp.90-91.

[10] LI, YY. and ZHANG, XF, 2011. Study on the Improvement of Code Quality Using AOP Journal of Taiyuan University of Technology, (6), p.3.

[11] Subramanyam, R. and Krishnan, M.S., 2003. Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects. IEEE Transactions on software engineering, 29(4), pp.297-310.

[12]Briand, L.C., Wüst, J., Ikonomovski, S.V. and Lounis, H., 1999, May. Investigating quality factors in object-oriented designs: an industrial case study. In Proceedings of the 21st international conference on Software engineering (pp. 345-354).

[13]El Emam, K., Melo, W. and Machado, J.C., 2001. The prediction of faulty classes using object-oriented design metrics. Journal of systems and software, 56(1), pp.63-75.

[14] Kumar, A., Kumar, R. and Grover, P.S., 2008, March. Notice of Violation of IEEE Publication Principles Towards a Unified Framework for Cohesion Measurement in Aspect-Oriented Systems. In 19th Australian Conference on Software Engineering (aswec 2008) (pp. 57-65). IEEE.

[15] Kumar, A., Grover, P.S. and Kumar, R., 2009. A quantitative evaluation of aspect-oriented software quality model (AOSQUAMO). ACM SIGSOFT Software Engineering Notes, 34(5), pp.1-9.

[16] Kumar, A., Kumar, R. and Grover, P.S., 2008, December. Towards a unified framework for complexity measurement in aspect-oriented systems. In Proceedings of the 2008 International

Conference on Computer Science and Software Engineering-Volume 02 (pp. 98-103).
[17] McCall, J.A., Richards, P.K. and Walters, G.F., 1977. Factors in software quality. volume i. concepts and definitions of software quality. GENERAL ELECTRIC CO SUNNYVALE CA.
[18] Kumar, A., Grover, P.S. and Kumar, R., 2009. A quantitative evaluation of aspect-oriented software quality model (AOSQUAMO). ACM SIGSOFT Software Engineering Notes, 34(5), pp.1-9.
[19] Kumar, A., Grover, P.S. and Kumar, R., 2009. A quantitative evaluation of aspect-oriented software quality model (AOSQUAMO). ACM SIGSOFT Software Engineering Notes, 34(5), pp.1-9.
[20] Dromey, R.G., 1995. A model for software product quality. IEEE Transactions on software engineering, 21(2), pp.146-162.
[21] Bourque, P. and Fairley, R.E., 2014. Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.
[22] Gill, N.S., 2005. Factors affecting effective software quality management revisited. ACM SIGSOFT Software Engineering Notes, 30(2), pp.1-4.
[23] Mili, H., Mcheick, H. and Sadou, S., JOURNALOFOBJECT TECHNOLOGY. JOURNAL OF OBJECT TECHNOLOGY, 1(1).
[24] Singh, P. and Singh, H., 2008. DynaMetrics: a runtime metric-based analysis tool for object-oriented software systems. ACM SIGSOFT Software Engineering Notes, 33(6), pp.1-6.
[25] Van Solingen, R. and Berghout, E., 2001, April. Integrating goal-oriented measurement in industrial software engineering: industrial experiences with and additions to the Goal/Question/Metric method (GQM). In Proceedings Seventh International Software Metrics Symposium (pp. 246-258). IEEE.
[26] Garcia, A., Sant'Anna, C., Figueiredo, E., Kulesza, U., Lucena, C. and von Staa, A., 2006. Modularizing design patterns with aspects: a quantitative study. In Transactions on Aspect-Oriented Software Development I (pp. 36-74). Springer, Berlin, Heidelberg.
[27] Kersten, M. and Murphy, G.C., 1999. Atlas: a case study in building a web-based learning environment using aspect-oriented programming. ACM SIGPLAN Notices, 34(10), pp.340-352.
[28] Sirbi, K. and Kulkarni, P.J., 2010. Impact of Aspect Oriented Programming on Software Development Quality Metrics. Global Journal of Computer Science and Technology.
[29] Soares, S., Laureano, E. and Borba, P., 2002. Implementing distribution and persistence aspects with AspectJ. A.C.M. Sigplan Notices, 37(11), pp.174-190.
[30] Garcia, A., Sant'Anna, C., Chavez, C., da Silva, V.T., de Lucena, C.J. and von Staa, A., 2003, May. Separation of concerns in multi-agent systems: An empirical study. In International Workshop on Software Engineering for Large-Scale Multi-agent Systems (pp. 49-72). Springer, Berlin, Heidelberg.
[31] Tsang, S.L., Clarke, S. and Baniassad, E., 2004, May. An evaluation of aspect-oriented programming for java-based real-time systems development. In Seventh IEEE International Symposium onObject-Oriented Real-Time Distributed Computing, 2004. Proceedings. (pp. 291-300). IEEE.
[32] Hannemann, J. and Kiczales, G., 2002, November. Design pattern implementation in Java and AspectJ. In Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (pp. 161-173).
[33] García, F., Bertoa, M.F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M. and Genero, M., 2006. Towards a consistent terminology for software measurement. Information and Software Technology, 48(8), pp.631-644.
[34] Tonella, P. and Ceccato, M., 2005. Refactoring the aspectizable interfaces: An empirical

assessment. IEEE Transactions on Software Engineering, 31(10), pp.819-832.
[35] Lamping, J.O., Xerox Corp, 2006. Aspect-oriented programming with multiple semantic levels. US Patent 7,140,007.
[36] Caldiera, VRBG and Rombach, H.D., 1994. The goal question metric approach. Encyclopedia of software engineering, pp.528-532.
[37] Ortu, M., Orrú, M. and Destefanis, G., 2019, February. On Comparing Software Quality Metrics of Traditional vs Blockchain-Oriented Software: An Empirical Study. In 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE) (pp. 32-37). IEEE.