Comparing Hybrid Tool for Static and Dynamic Object-Oriented Metrics

Babur Hayat Malik[1], Javaria Khalid [2], Hafsa Arif [3], Ayesha Sadiqa[4] ,Amara Tanveer[5], Asia mumtaz[6]
Zartashiya Afzal[7], Samreen Azhar[8], Muhammad Numan Ali[9]
Department of Computer Science and Information Technology
University of Lahore, Chenab Campus, Gujrat Pakistan

***Abstract*—Software metrics are created and used by the distinctive programming associations intended for assessing, guaranteeing program excellence, activity, and software recovery. Software metrics have turned into a basic part of programming growth and are utilized in each period of the product development life cycle. Software metrics essentially measure programming items like plan source code and help us in taking technical and administrative choices. The desire of this examination is to play out the relative investigation of static and dynamic metrics. In any case, software quality characteristics, for example, performance, execution time and dependability rely upon the dynamic exercises of the product artifact. Due to every one of these variables, we favor dynamic metrics instead of customary static metrics. With the assistance of customary static metrics, we are not capable to analyze different actualities of programming. There are various types of this OO static and dynamic equipments. In this paper we have played out a similar investigation of different OO static and dynamic metrics tools and find out the hybrid too is counted as best one extraction of both, static and dynamic characteristics from mobile Android applications. The source code and a Docker compartment is utilized by open source tool in only three phases pre-static, static and dynamic examination.**

Keywords—*Software metrics; static metrics; dynamic metrics; Object Oriented (OO)*

I. Introduction

A software metric is fundamentally a software engineering track which relates to the various software developments and dimensions. One effective tool used for software product analysis is software metrics [1] [2] [3]. It plays a major role in the analysis and improvement of software quality along with measurement of software complexities [4]. An appropriate software model is required for the development of reliable software. ISO 9126 is one of the quality models that uses software metrics [5] [6]. Several tools are required for making of software quality models which intends to do metrics calculations. Though, these tools are also required to produce accurate data [7]. Software metrics are categorized into three parts: product metrics, process metrics, and project metrics, as shown in Fig. 1.

Results are specified by a standard unit known as "Metric". It is used for evaluation of software processes, products, and services. Different authors have proposed several object-oriented (OO) metrics which are quite famous in the present software development environment [9]. These are different from standard metrics as they use objects instead of algorithms as a key object [10]. Traditional metrics are not eligible in determining the quality as intricate projects are enforced through OOD design practices, so they are required [11]. Somerville [12] described metrics in two types known as static and dynamic. Static metrics analyze code before executing it whereas dynamic metrics analyze code during code execution. In this research, static metrics is more focused on the understanding of procedural and object-oriented programming languages [4]. In this paper comparison of Static and dynamic OO tools are proposed. They are more emphasized for finding object-oriented metric tools on the basis of several parameters.

This paper is written in several sections. Firstly, Section II describes the literature work of various Object-oriented Static and dynamic metrics tools. Then, in Sections III is discussed the differentiation between Static and Dynamic Metrics. Various types of object-oriented Static and dynamic Metrics are presented in Section IV. In Section V, the comparative study of OO Metric Tools is performed. Lastly, Section VI, presents the conclusion of this article.

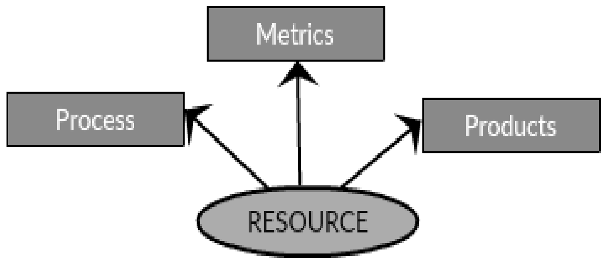

Fig. 1. Software Metrics [8]

II. Literature Review

Various OO metrics are developed until now which differ in their properties and features. The main purpose of this paper is to find out huge OO metric computational tools on the basis of their properties. Complex metrics to be resolved are still an issue whereas in traditional OO some metrics like CK and MOOD are considered quite helpful in the development of software [13].

Munson and Hall [14] identified the program complexity level along with three processes of functional, fractional, and operational complexity. Mayo et al. [15] discussed the quality attribute of the interface which calculates modules complexity and dynamic metrics when it's executed.

Honglei et al. [16] presented metrics definition, types, and history. Measurement of software complexity is one important factor and it's also related to software development price factor.

Hassoun et al. [17] proposed Dynamic Coupling Metric (DCM) for object level coupling that considers program execution as it is used to measure objects coupling during runtime. Though it also estimates the runtime complexity and system comparison at meta-level along with those systems which have no reflective features.

Singh and Singh [18] presented four class-level dynamic couplings for identifying object-oriented systems quality. They are more determined in finding key coupled classes consisting of most active classes during runtime. Gupta [19] presented three dynamic coupling metrics which consists of foremost relations between objects during runtime, i.e. aggregation, inheritance, etc.

Mayo et al. [20] defined both automated Interface and Dynamic Metrics. The first one is used for identifying modules complexity whereas dynamic metric calculates quality factor during execution. Hays in [21] identified OO systems testing and compared them with conventional programming language testing.

Mohsin, Shaikh, and Zeeshan Kaleem [22] presented the idea of code comprehension with a combination of Software metrics and techniques called Program Slicing. It is basically coded automation analysis for coupling, cohesion, and complexity.

Debbarma, Mrinal Kanti et al. [23] described the comparison of static and dynamic metrics and analyzed them in terms of regression testing that helps in effort and time estimation used during testing.

III. Types of Metrics

In various real-time applications, there is a small number of the most eminent metrics that are analyzed. There are different categories of metrics that are presented below:

A. *Traditional Metrics*

In an object-oriented system, traditional metrics are commonly applied to the methods that include the class operation. "A method is a component of an object that operates on data in response to a message and is defined as part of the declaration of a class". Methods reveal how a problem is fragmented into different sections. Two traditional metrics are Cyclomatic complexity and size (line counts) [24].

B. *Object-Oriented Metrics*

Object-oriented software metrics emphasis on measurements that are functional to the conceptions of classes, coupling, and inheritance. Encapsulation metrics are applied for classes, not for modules. Information Hiding is measured & enhanced due to Inheritance complexity is additional, the level of abstraction can be measured by Object Abstraction metrics. These are as follows:

- Metrics correlated with Class
- Metrics associated with Methods
- Metrics Encapsulation
- Measurement of Cyclomatic complication
- Metrics used for Inheritance [25, 26].

1) Static metrics: This Metric is the outcome of non-executable code. Static metrics describe system features from design through maintenance. Earliest Metric used for Static is [27] (LOC/KLOC) examine the throughput of a software package. In earlier 1990, McCabe was the most powerful metric for examining the intricacy of cyclomatic [28] complexity. Complexity is evaluated from the graphical representation and various mathematical equalities. In 1976 McCabe [29] demarcated the cyclomatic complexity metric. It measures the total numbers of independent routes over a software component.

2) Dynamic metrics: These are resultant of source code investigation. When code is running it evaluates what is really happening. Dynamic metrics comprise complication events and processes beneficial in consistency demonstrating at the same time [30]. When software is executing its values are reliant on the involvement or experimental information. From coding to maintenance system aspects are classified by dynamic metrics [8]. The comparison of static and dynamic metrics with its merits, demerits are shown in Tables I and II.

TABLE I. Static vs. Dynamic Metrics

Static Metrics	***Dynamic Metrics***
1. Its nature is always static.	1. Its nature is always dynamic.
2. It is simpler and easier to collect.	2. It is difficult and tough to gather.
3. OO software attributes are difficult to examine.	3. Different characteristics are easy to inspect like Inheritance, polymorphism, coupling, cohesion, and difficulty.
4. It takes less time as compared to dynamic analysis of software.	4. It takes more time to perform dynamic analysis of a program.
5. It is available at the early stages of the software development life cycle.	5. It is accessible late in the software development life cycle.
6. For software quality prediction its results are less accurate.	6. For software quality prediction its results are more accurate.
7. More Tools are effortlessly available to accomplish this examination.	7. Only a few tools are available for this analysis.
8. Its implementation is done on the code.	8. Its implementation is performed while code is being run.
9. It deals with structural aspects of the system.	9. It deals with the behavioral aspects of the system.
10. It identifies vulnerabilities in a runtime environment.	10. It can find weaknesses in the code at the exact location.

TABLE II. COMPARISON OF STATIC VS. DYNAMIC METRICS

Serial No.	Static Software Metric	Description	Merits	Demerits	Equations
1	SLOC (Source lines of code) [4]	It evaluates total lines in the program to measures its size. When software is developed it determines the productivity of the program.	Measuring automation possibilities	Inaccuracy in Accountability.	For (i = 0; i < 100; i++) printf("hello"); /* How many lines of code is this? */ Above case illustrate the following information: • 1(LOC), • 2(SLOC) (for statement and printf statement), • 1 comment line.
2	LOC (Line of Code)[4]	It consists of any number of lines, consist of source, whitespace, and comments.	Universal measure.	• Several languages issues • GUI tools Starter	1 (LOC) as stated in the above example
3	AMLOC(Average lines per method) [32]	It defines the average size of the method.	Method Size can detect simply.	Less clear and additional code statement.	Average Method Size= (The Total number of LOC) / (Number of Methods)
4	ACLOC(Average lines per class) [32]	It determines the moderate size of class according to LOC.	It is simple to define the number of code lines for each class therefore accurately determine the size of the class	More code lines can't be verified and can't be altered safely.	Average Class Size= (The Total number of LOC) / (Number of Methods)
5	NCLASS [32]	These metrics calculate the number of classes in the project.	Main Characteristics are undone or round-trip engineering.	In general UML figure categories, it supports class diagram.	----
6	Cyclomatic complexity [33]	Indicate the program difficulty areas.	• It assesses AI semantic complexity. • Useful in geographical and landscape environmental inquiry.	Positive correlation among cyclomatic complexity and defects. More errors in maximum complexity functions and methods.	$M = E - N + 2P$, E = Graph edges. N = Graph nodes. P =Connected components.
7	Function point [34]	It is a measurement element to examine the business functionality that delivers to a customer.	• An end-user business function maps to functional consumer requests like data entry. • Function points plot easily into user-oriented requests.	lbrecht perceived in his research that Function Points were extremely associated with code lines and increase complexity.	• Define the number of data functions (ILFs and EIFs) • indicative size (fp) = 35 x number of ILFs + 15 x number of EIFs.

8	Bug Counting [34]	Program inaccuracy results in improper or unpredicted result act in unintentional ways.	• Failure count models
			• Error seeding models
				• Involved more in program performance, does not concentrate on a number of program bugs.
				• Most requests of customers define according to functional reliability and not in terms of errors.
					Bugs.Count
					Bugs.SUM(Effort)
					Bugs.SUM(CustomValues.Number("Cost"))
					
					UserStories.SUM(CustomValues.Number("Bugs Count")) + Bugs.Where(UserStory.Feature == null || Feature.Id != UserStory.Feature.Id).Count

9	Halstead complexity [4][34]	Recognize computable software properties and the associations between them.	• These are traditional metrics but they can evaluate projects like C, C++, and Java. • It calculates the bugs, project length, size, and validity period.	• Modularity • All-Depth • Operator Type • Database Impact and Declaration	Program Vocabulary: $N=n_1 + n_2$ Program Length: $N=N_1+N_2$ Calculated Program Length: $N= n_1 \log2 n_1+ n_2 \log2 n_2$
10	Continuous Value Metrics[24]	In numerous circumstances it innate incorrectness: A straight line in a diagram can have the equivalent general average as a slanting line.	• Define a best, fewer bugs metric where single value metrics is possibly imprecise. • Value Metrics extension	Secondary metrics are frequently insufficient to actually define the dissimilarity in performance, demanding further tertiary metrics.	----

IV. Current Issues and Challenges

- After negotiations upon dynamic metrics, it has definitely perceived that currently not any metrics available for testability at execution time of the software systems.
- Its benefit includes accuracy and precision; however, they are more difficult in evaluation to static ones. Therefore, a good hybrid approach is required.
- For the analysis of different software aspects pseudo dynamic metrics is another auspicious research prospect readily accessible to researchers.
- It can be certainly observed from the survey of many research studies conducted by different authors that dynamic metrics are examined and tested using a project that is not bulky [31].

V. Results

We have to concern together static along with Dynamic Metrics to realize the deviation. After comparing both of these metrics we concluded that dynamic metrics analysis gives result at execution time of programs whereas static analysis at rest of the SDLC process. So, for dynamic analysis data is collected with the help of tool based on either Java or C++ based application, then apply a statistical tool to measure the quality of the product. Dynamic analysis can give a better result than static analysis.

AndroPyTool executes different tools in order to extract wide-ranging features from an input set of Android samples. All these features and the evidence that they symbolized are organized in three dissimilar classes (pre-static, static and dynamic), both the features and how they are extracted.

In Pre Static it comprises extracting information without inspection of code and permits to categorize and to track the sample. It also includes the package name and the main activity name, which are found with Andro-guard. In Static analysis, it contains those features that are regained by analyzing the application at the code level. In this category, features such as API calls, activities, opcodes or permissions can be originated. In Dynamic Analysis, it includes Droid Box tool for this purpose, which allows to dynamically find dissimilar information in real time. The information gathered by the Droid Box tool includes: the use of cryptographic functions, loaded DEX classes in run time and the kind of operation, network connections, SMS, phone calls, started services, enforced permissions and information leaks detected. The detail diagram of AndroPyTool is shown in Fig. 2 [38].

VI. Discussion

Various OO Metrics tools their description, merits and demerits are studied in this research paper. These tools are tabulated under various attributes that would be of interest to developers and researchers using the tools as elaborated in Table III. Our study has further pointed out the work and research findings that has been done till now to use of hybrid approach of static as well as dynamic metrics, although they have tremendous scope. Based on the analysis of existing dynamic metrics, we have tried to reveal potential research challenges and opportunities existing in the field of dynamic metrics. Best methodology that is suitable for pre-static, static and dynamic metrics is hybrid approach and its tool that is AndroPyTool.

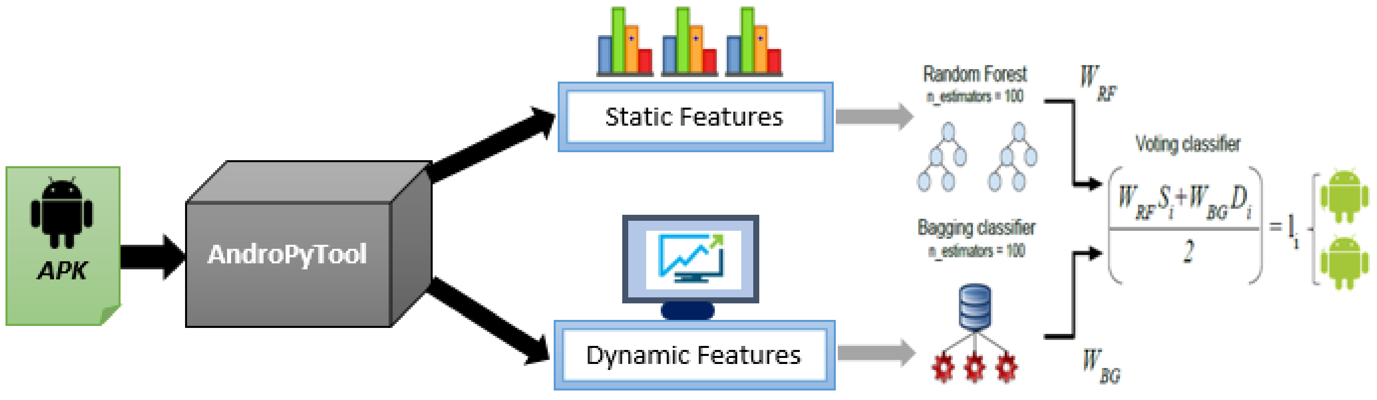

Fig. 2. AndroPyTool [38].

TABLE III. COMPARISON OF STATIC VS. DYNAMIC METRICS TOOLS

Tool Name	**Description**	**Language**	**Availability**	**Authors**	**Tool Type**
CheckStyle [35]	Java Checkstyle is an improvement tool to enable designers to compose Java code that clings to a coding standard. Presently Checkstyle gives checks that discover class plan issues, copy code, or bug designs like twofold checked to bolt.	Java	Open source	Oliver Burn	Static
FindBugs [35]	This is the Static Analysis tool and is open source that checks and study class files or JAR libraries for probable problems adjacent to a list of bug patterns by matching the byte code [5].	Java	Open source	David Hovemeyer and William Pugh	Static
StyleCop [35]	For plugins and customs rules, *StyleCop* provides an extensible framework to write down custom rules which match up to our requirements.	C#	Free	Andy Reeves, Chris Dahlberg	Static
JMT [36]	It only associates the Metrics with Java language.	Java	Free	Politecnico di Milano and Imperial College London	Dynamic
QMOOD++ [37]	QMOOD++ is easy and free of cost accessible in the runnable application and source code form. It handles the 30+ Metrics. QMOOD++ is an inclusive, multi-handler, multiprocessing, incorporated software tool.	C++	Free	Bansiya, Jagdish, and Carl Davi,	Dynamic
JMetric [11]	JMetric only works with Java. Its information is presented through tables and charts.	Java	Free	Commercial Tool	Dynamic
AndroPyTool [38][39]	AndroPyTool incorporate different analysis tools and Android applications Processing tools, in order to convey fine-grained reports drawing their individual performance and features.	Python	----	Alejandro Mart, Raul Lara-Cabrera, David Camacho	Hybrid

VII. CONCLUSION

A correlation of diverse software metrics and its major tools are presented in this comparative study. On the base of their major types like static and dynamic metrics, these are differentiated. At early stages of software development life cycle (SDLC), Static metrics are reachable easily. These metrics manage the overall structural qualities of the product framework and very simple to assemble. The unpredictability of static metrics has calculated the measure of exertion expected to create and keep up the code. In the latter stage of the software development life cycle, dynamic metrics are easily reachable.

These metrics confine the dynamic conduct of the framework and difficult to acquire and got from hints of code. After a virtual study of various static and dynamic tools are performed and broke down that hybrid tool is best in the greater part of the android applications. AndroPyTool, the primary objective is to furnish scientists and malware examiners with an incredible and coordinated device for extracting multi-source highlights from Android applications. In future work, more tools and features can be add on into AndroPyTool tool for better analysis and to improve the data analysis stages, in order to give more functionalities to the users.

REFERENCES

[1] M. Sharma, Dr. G. Singh, "Analysis of Static and Dynamic Metrics for Productivity and Time Complexity," IJCA, vol. 30, issue.31, September 2011.

[2] H. F Li and W. K Cheung, "An Empirical Study of Software Metrics," Software Engineering IEEE Transactions,vol.13, issue. 6, pp. 697-708, 1987.

[3] N. E Fenton "Software Metrics," Conference Proceedings of on the future of Software engineering ICSE vol. 8, issue: 2,2000.

[4] M. Sharma, A. Bhardwaj, L. Singh, N.Singh and C. Sharma, "Comparative study of static metrics of procedural and object oriented programming languages," International Journal of Computers & Technology, Volume 2 No.1 February 2012.

[5] ISO ISO/IEC 9126-1, Software engineering–Product Quality - Part 1: Quality model., 2001.

[6] ISO. ISO/IEC 9126-3, Software engineering–Product Quality - Part 3: Internal metrics., 2003.

[7] J. Novak and G. Rakić, "Comparison of software metrics tools for .net," University of Novi Sad, Faculty of Sciences, Department of Mathematics and Informatics, 2011.

[8] BM. Goel and S. Bal Gupta, "A Comparative Study of Static and Dynamic Object Oriented Metrics," International Journal of Information Technology & Systems, vol. 5, issue. 1, 2016.

[9] J. Chawla and A. Agarwal, "Object-Oriented Design Metrics to Predict Fault Proneness of Software Applications," (IJCSIT) International Journal of Computer Science and Information Technologies, vol. 5 (3), 2014.

[10] A. Albrecht and J. Gaffney: Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation; in IEEE Trans. Software Eng., pp. 639-648,2008.

[11] Kayarvizhy N, “Systematic Review of Object Oriented Metric Tools,” International Journal of Computer Applications vol. 135 issue.2, February 2016.

[12] Somerville “Software Engineering”, 6th Edition, Editor: Addison Wesley.

[13] Y. Ma, K. He, D. Du, J. Liu, and Y. Yan , “A Complexity Metrics Set for Large-scale Object-oriented Software Systems,” IEEE International Conference on Computer and Information Technology (CIT'06).

[14] John C. Munson and Gregory A. Hall, “Estimating test effectiveness with dynamic complexity measurement,” Empirical Software Engineering Journal.

[15] Kevin A. Mayo, Steven A. Wake and Sallie M. Henry, “ Static and Dynamic Software Quality Metric Tools,” Department of computer Science, Virginia Tech, Blacksburg.

[16] B. Mohan Goel and S. Bal Gupta, “Dynamic Coupling Based Performance Analysis of Object Oriented Systems,” International Journal of Advanced Research in Computer Science, vol. 8, issue. 5, May-June 2017.

[17] Y. Hassoun, R. Johnson and S. Counsell, “A Dynamic Runtime CouplingMetric for Meta Level Architectures,” In Proceedings of Eighth EuromicroWorking Conference on Software Maintenance and Reengineering, pp. 339, 2004.

[18] P. Singh, H. Singh, “Class-level Dynamic Coupling Metrics for Static and Dynamic Analysis of Object-Oriented Systems,” International Journal of Information and Telecommunication Technology, pp. 16-28, 2010.

[19] V. Gupta, “Validation of Dynamic Coupling Metrics for Object-Oriented Software.” ACM SIGSOFT Software Engineering Notes,vol.36(5), 2011.

[20] Kevin A. Mayo, Steven A. Wake, Sallie M. Henry, “Static and Dynamic Software Quality Metric Tools,” Department of computer Science, Virginia Tech, Blacksburg.

[21] J. Huffman Hayes, “Testing of Object-Oriented Programming Systems (OOPS): A Fault-Based-Approach,” Science Applications International Corporation, 1213 Jefferson-Davis Highway, Suite 1300, 22202 Arlington, Virginia.

[22] M. Shaikh, and Z. Kaleem. "Program Slicing Based Software Metrics towards Code Restructuring," In Computer Research and Development, Second International Conference on, pp. 738-741. IEEE, 2010.

[23] Debbarma, M. Kanti, N. Kar, and A. Saha, "Static and dynamic software metrics complexity analysis in regression testing," In Computer Communication and Informatics, International Conference on, pp. 1-6. IEEE, 2012.

[24] S. Pasupathy and R. Bhavani, “Object Oriented Metrics Evaluation,” International Journal of Computer Applications (0975 – 8887) vol.78,issue.1, September 2013.

[25] S. Morasca, “Software Measurement: State of the Art and Related Issues, slides from the School of the Italian Group of Informatics Engineering,” Rovereto, Italy, September 2008.

[26] J. Alghamdi, R. Rufai, and S. Khan, “Oometer: A software quality assurance tool. Software Maintenance and Reengineering 2009,” 9th European Conference on, pp. 190, March 2010.

[27] Li, Cheung, W.K, “An Experimental investigation of software metric and their relationship to software development effort,” IEEE Transaction on software engineering 649-653, Piscataway, NJ, USA.

[28] Thomas J. McCabe, “A Complexity Measure, IEEE Transaction on Software Engineering,” vol.2 issue. 4, pp. 308-320.

[29] S. Singh, K.S. Kahlon, “Static Analysis to Model & Measure OO Paradigms,” SAC, ACM.

[30] K. Kaur, K. Minhas, N. Mehan, and N. Kakkar, “Static and Dynamic Complexity Analysis of Software Metrics,” World Academy of Science, Engineering and Technology International Journal of Computer and Systems Engineering vol.3, issue.8, 2009.

[31] Chhabra JK, Gupta V, “A survey of dynamic software metrics. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY,” vol.25(5),pp.1016–1029 Sept. 2010.

[32] G. Singh, M. Sharma, “A Comparative Study of Static Object Oriented Metrics,” International Journal of Advancements in Technology, 21 January 2018.

[33] G. K. Gill, C. F. Kemerer, “Cyclomatic Complexity Density and Software Maintenance Productivity”, IEEE Transactions on Software Engineering, 1981, pp. 1284-1288.

[34] A. Versa, Rahul, “A Study of Various Static and Dynamic Metrics for Open Source Software,” International Journal of Computer Applications (0975 – 8887) vol. 122, 10 July 2015

[35] J. Novak, A. Krajnc and R. Zontar, “Taxonomy of Static Code Analysis Tools,” 16 March 2015.

[36] M. Bertoli G. Casale and G. Serazzi, “JMT: Performance engineering tools for system modelling,” Giuliano Casale, 04 June 2014.

[37] J. Bansiya, C. Davis, Using QMOOD++ for object-oriented metrics, Dr. Dobb’s Journal , 1997.

[38] A. Martin, R. lara-cabrera and D. Camacho, “A new tool for static and dynamic Android malware analysis,” 24 September 2018.

[39] A. Martin, R. LaraCabrera and D. Camacho, “Android malware detection through hybrid features fusion and ensemble classifers:the AndroPyTool framework and the OmniDroid dataset,” 05 February 2019.