

Object Oriented Metrics to measure the quality of software upon PHP source code with PHP_depend

Study case Request online System application

Sarwo¹, Harco Leslie Hendric Spits Warnars², Ford Lumban Gaol³
 Doctor of Computer Science
 Bina Nusantara University
 Jakarta, Indonesia
 sarwo@binus.ac.id¹, shendric@binus.edu², fgaol@binus.edu³

Richard Randriatoamanana
 Institut de Calcul Intensif
 Ecole Centrale de Nantes
 Nantes, France
 richard.randriatoamanana@ec-nantes.fr

Abstract—Nowadays, a company should be equipped with information technology and one of them is business application either web based or non web based which is built with object oriented programming in order to help the management to run their daily business processes. In order to supports a company with best excellent software applications then we need to make sure the quality of software application in term of easy to maintenance, understandability, reusability and so on. In term to measure the quality of software, there are many software metrics applications either free or proprietary which can be downloaded on Internet and one of them is PHP_depend. PHP_depend is software metrics which measure the quality of software based on PHP source code, by taking the PHP source code and parses it into Abstract Syntax Tree (AST). This paper will investigate how to measure quality of software based on PHP source code with PHP_depend software metrics. The investigation will examine codes from Request online system application which supports improvement or update request upon SAP application in Astra Graphia Information Technology PT. The measurement will be assessed based on metrics which is categorized into project, package, file, class and methods.

Keywords—*Object Oriented Metrics; PHP Metrics; Software Metrics; Software Quality; Software Measurement.*

I. INTRODUCTION

Astra Graphia Information Technology PT. is an Information Technology (IT) consultant company which sale, distributed, and maintain both hardware and software for all level customer with either in-house or reseller software. Astra Graphia run SAP application from www.SAP.com which supports their daily business activity processes. However, there are some improvement and update on implemented SAP application on daily basis, which is needed by departments in order to supports their business activity processes. Request online system is a web based application which is built with Personal Home Page (PHP) language programming in order to record and control all improvement or update request in SAP application. For future implementation, the intelligent of this application will be extended with such as technology such as Data Warehouse [19,20,21,22,23] or Data Mining with some options algorithms such as Attribute Oriented Induction(AOI)[24,25,26,27,28,29,30,31] or Attribute

Oriented Induction High level Emerging Patterns (AOI-HEP)[32,33,34,35,36].

Meanwhile, Open Source Software has impacted software industry and recently became extremely popular such as Personal Home Pages (PHP), Java Server Pages (JSP), Java and so on. This paper will investigate how to measure PHP source code with PHP_depend. The investigation will examine 32 PHP source codes from Request online system application which supports improvement or update request upon SAP application, PHP_Depend is a small program that performs static code analysis on a given source base , According to G. Kour and S. Evolution PHP_Depend can generate a large set of software metrics from a given code base and identify parts of an application where refactoring should be applied [1].

Object Oriented that makes designs more powerfull, more maintainable, and more reusable for design system. Recently, almost all system design already uses a technique Object Oriented, in the design of a system designed to ensure software quality meet the standard Object Oriented Programming (OOP) need be tested to detect and subsequently handle all errors in it. A number of schemes are used for testing purpose and this measurement will make the result software which easy maintenance, understandability and reusability. This paper present how to perform a software measurement and presents the results in the a report that is easily understood by management, with this report can be used to predict the level of error and how much cost to spend on developing the program

II. RELATED WORK IN THE LITERATURE

Measuring the discriminative power of object-oriented class cohesion metrics [2], this paper obtain the same cohesion values for different classes that have the same number of methods and attributes but different CPCIs. Software Quality Estimation through Object Oriented Design Metrics [3], this paper obtain how these metrics are useful in determining the quality of any software designed by using object oriented paradigm. Critical Analysis of Object Oriented Metrics in Software Development [4], this paper obtain to a review and analysis of object oriented metrics is presented for identification and validation of object oriented metrics and out of various metrics. Evaluating the impact of different types of

inheritance on the object oriented software metrics [5], this paper discuss focuses on effects of inheritance on object oriented metrics. Implementation of ISO 9126-1 quality model for asset inventory information system by utilizing object oriented metrics[6], this paper aim Proposed ISO 9126-1 quality model has been internally evaluated by object oriented metric using case study on Politeknik Caltex Riau (PCR) which is one of the organization that engaged in academic area. Software Product Quality[7], this chapter by Martin Glinz discus about software product quality. Effectiveness of encapsulation and object-oriented metrics to refactor code and identify error prone classes using bad smells [8], this research develop a metrics model to identify smelly classes to improve Encapsulation and Object-oriented Metrics. Applying the ISO 9126 quality model to test specifications [9], this paper apply ISO 9126 for model to test specifications. An ISO 9126-based Quality Model to Assess the Quality of TTCN-3 Test Specifications 2 Software Quality Models [10], this chapter apply ISO 9126 to Assess the Quality of TTCN-3 Test Specifications, TTCN-3 is ETSI Centre for Testing and Interoperability.

III. PHP_DEPEND SOFTWARE METRIC MEASUREMENT EXPERIMENTS

The experiments used source codes of Request online system application (ROLIS) which was allocated in sub directory C:\PEAR\projektoPHP\rolisdev as PHP project. The experiments were carried on windows 7.0 operating system and used PHP version 5.2.3, including PEAR (PHP Extension and Application Repository) package which provide library PHP open source code. This PHP project will be measured with PHP-depend software metrics with such as next steps[13]:

Install PHP minimum version of 5.2.3 by downloading XAMPP for windows v.5.6.23(PHP 5.6.23) from <https://www.apachefriends.org/index.html>

Download PEAR package manager go-pear.phar from <https://pear.php.net/manual/en/installation.getting.php> and save to computer.

Running go-pear.phar with PHP and typed PHP statement to run file go-pear.phar. Running this file will download needed files in drive C:\PEAR.

In order to check if PHP_depend is working in your machine then type statement: "pdepend -version" with command prompt in directory C:/PEAR. Figure 1 shows the statement process.



```

Administrator: Command Prompt
c:\PEAR>
c:\PEAR>php pdepend --version
PHP_Depend 1.1.4 by Manuel Pichler
  
```

Fig.1 Check version Software

There are 3 types PHP_depend experiments which will be carried on such as summary XML, Pyramid report and Charts.

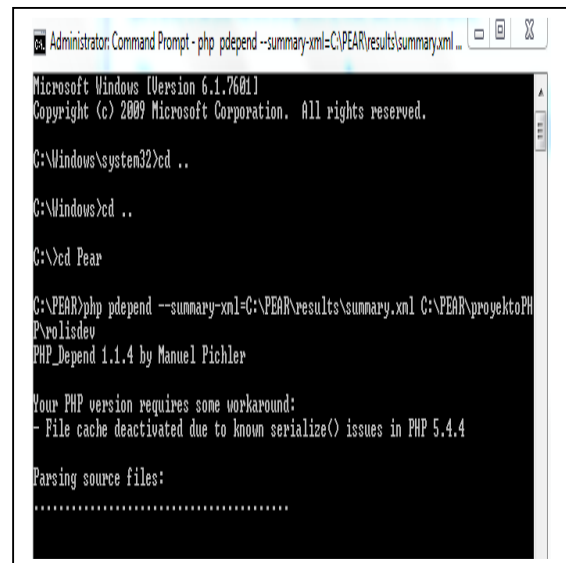
The execution time for each experiment will be different which depend on your computer size and allocation process in your computer. Next is explanation for each PHP_depend experiment and they are:

A. Summary XML.

This PHP_depend experiment will be executed in sub directory PEAR by giving statement: `php pdepend --summary-xml=C:\PEAR\results\summary.xml C:\PEAR\projektoPHP\rolisdev`. Figure 2 shows the statement execution and was executed in 3.22 minutes with memory allocation 94.50 MB and the result shows:

- Parsing source files : 657
- Executing Cyclomatic Complexity-Analyzer:5629
- Executing Class Level-Analyzer: 6240
- Executing Code Rank-Analyzer: 1410
- Executing Cohesion-Analyzer:10716
- Executing Coupling-Analyzer: 5639
- Executing Hierarchy-Analyzer: 5375
- Executing Inheritance-Analyzer: 1405
- Executing NPatch Complexity-Analyzer:5685
- Executing Node Count-Analyzer : 3692
- Executing Node Loc-Anlyzer: 3947

Moreover, file log summary.xml will be generated based on this statement execution and saved in sub directory C:\PEAR\results and consist of 21 metrics in PHP_depend with each score and they are : AHH=0.242, ANDC=0.588, CALLS=9412, CCN=16010, CCN2=17457, CLOC=114, CLSA=0, CLSC=254, ELOC=59994, FANOUT=346, LEAFS=213, LLOC=49466, LOC=86583, MAXDIT=3, NCLOC=70100, NOC=254, NOF=281, NOI=0, NOM=3152, NOP=5, ROOTS=3.



```

Administrator: Command Prompt - php pdepend --summary-xml=C:\PEAR\results\summary.xml ...
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd ..
C:\Windows>cd ..
C:\>cd Pear
C:\PEAR>php pdepend --summary-xml=C:\PEAR\results\summary.xml C:\PEAR\projektoPH
P\rolisdev
PHP_Depend 1.1.4 by Manuel Pichler

Your PHP version requires some workaround:
- File cache deactivated due to known serialize() issues in PHP 5.4.4

Parsing source files:
.....
  
```

Fig.2 generate Summary.xml

```
C:\PEAR>php pdepend --overview-pyramid=c:\PEAR\results\pyramid.svg c:\PEAR\projektoPHP\rolisdev
PHP_Depend 1.1.4 by Manuel Pichler

Your PHP version requires some workaround:
- File cache deactivated due to known serialize() issues in PHP 5.4.4

Parsing source files:
..... 60
..... 120
..... 180
..... 240
..... 300
..... 360
..... 420
..... 480
..... 540
..... 600
..... 657

Executing Coupling-Analyzer:
..... 1200
..... 2400
..... 3600
```

Fig.3 generate Pyramid Report

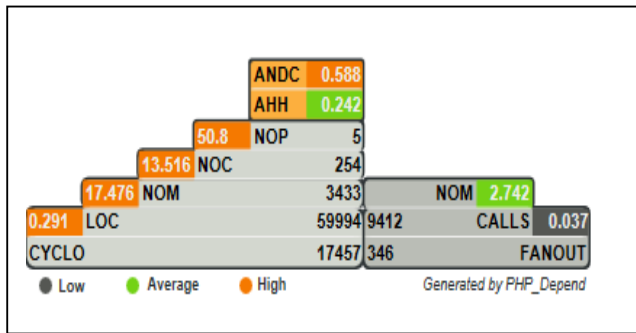


Fig.4 Pyramid Report

B. Pyramid report.

This PHP_depend experiment will be executed in sub directory PEAR as well and by typing statement:

Phpdepend--overview-pyramid=c:\PEAR\results\pyramid.svg :\PEAR\projektoPHP\rolisdev. Figure 3 shows the statement execution and was executed in 2.19 minutes with memory allocation 93.50 MB and the result will shows:

- Parsing source files : 657
- Executing Coupling-Analyzer: 5629
- Executing Cyclomatic Complexity -Analyzer:5681
- Executing Inheritance-Analyzer: 1405
- Executing Node Count-Analyzer : 3692
- Executing Node Loc-Anlyzer: 3947

File log pyramid.svg will be generated based on this statement execution and saved in sub directory C:\PEAR\results and will be showed with browser as shown in figure 4.

C. Charts.

This PHP_depend experiment will be executed in sub directory PEAR as well by giving statement: php pdepend --jdepend-chart = \ PEAR \ results \ jdepend.svg : \ PEAR \ proyektoPHP \ rolisdev. Figure 5 shows the statement execution and was executed in 00.15 minutes with memory allocation 88.25 MB and the result will shows:

- Parsing source files : 657
- Executing Dependency-Analyzer: 3411

File log jdepend.svg will be generated based on this statement execution and saved in sub directory C:\PEAR\results and will be showed with browser as shown in figure 6.

```
Administrator: Command Prompt
C:\PEAR>php pdepend --jdepend-chart=c:\PEAR\results\jdepend.svg c:\PEAR\projektoPHP\rolisdev
PHP_Depend 1.1.4 by Manuel Pichler

Your PHP version requires some workaround:
- File cache deactivated due to known serialize() issues in PHP 5.4.4

Parsing source files:
..... 60
..... 120
..... 180
..... 240
..... 300
..... 360
```

Fig.5 generate Charts Report

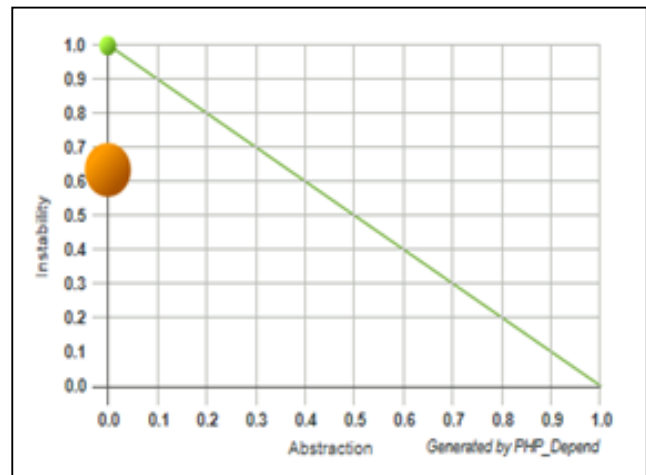


Fig. 6 Chart Report

IV. INTERPRETATION OF PHP_DEPEND SOFTWARE METRICS MEASUREMENT EXPERIMENTS

After running the PHP_depend software metric then we need to interpret the result as the conclusion of software metric measurement upon Request online system application as PHP

project which was allocated in sub directory C:\PEAR\proyektoPHP\rolisdev will generate Pyramid Report shown in Figure 4 and Chart report shown in Figure 6.

- $LOC = CYCLO/LOC$ (1)
- $NOM = LOC/NOM$ (2)
- $NOC = NOM/NOC$ (3)
- $NOP = NOC/NOP$ (4)
- $CALLS = FANOUT/CALLS$ (5)
- $NOM = CALLS/NOM$ (6)

The score for each of metrics in these 3 categorization of metric are shown in figure 4 as result of running pyramid report experiment and they are ANDC=0.588, AHH=0.242, NOP=5, NOC=254, NOM=3433, LOC=59994, CYCLO=17457, CALLS=9412 and FANOUT=346 as shown in figure 4. In order to find the software size then we need to score all these metrics with equations (1) to (6) where the number of dividend and divisor in each equation will refer to number its metric as shown in figures 4 or 8. The scoring only applied to size and complexity, and coupling categorization metrics as shown in figure 7, where inheritance categorization metrics such as ANDC and AHH didn't include since they have already their scores. As shown in figures 4 or 8, metrics CYCLO and LOC in equation (1) will have number 17457 and 59994 respectively, metrics LOC and NOM in equation (2) have number 59994 and 3433 respectively. Metrics NOM and NOC in equation (3) have number 3433 and 254 respectively, metrics NOC and NOP in equation (4) have number 254 and 5 respectively. Meanwhile, Metrics FANOUT and CALLS in equation (5) have number 346 and 9412 respectively, metrics CALLS and NOM in equation (6) have number 9412 and 3433 respectively.

The metrics LOC, NOM, NOC and NOP in left side in figure 8 as size and complexity categorization metric as shown in figure 7 with equations between (1) and (4). The score of metric LOC is executed with equation (1) = $CYCLO/LOC=17457/59994 = 0.291$. The score of metric NOM is executed with equation (2) = $LOC/NOM=59994/3433=17.476$. The score of metric NOC is executed with equation (3) = $NOM/NOC=3433/254=13.516$ and the score of metric NOP is executed with equation (4) = $NOC/NOP=254/5=50.8$. Figure 4 shows these 4 size and complexity categorization metrics' score such as LOC=0.291, NOM=17.467, NOC=13.516 and NOP=50.8 in the left side.

Meanwhile, the score for metrics CALLS and NOM in right side in figure 8 as coupling categorization metric as shown in figure 7 will be executed with equations (5) and (6). The score of metric CALLS is executed with equation (5) = $FANOUT/CALLS=346/9412=0.037$ and the score of metric NOM is executed with equation (6) = $CALLS/NOM=412/3433=2.742$. Figure 4 shows these 2 coupling categorization metrics score such as CALLS=0.037 and NOM=2.742 in the right side.

Moreover, in order to give easy understanding the software size measurement based on PHP_depend software metrics as shown in pyramid report in figure 4, then the metrics scores will be categorized into 3 different colour such as black, green

and orange colours. The black, green and orange colours refer to low, average and high scores based on reference value on table 1, where each number as minimum score. For example, AHH metric has minimum low score 0.09 with range score between 0.09-0.209, minimum average score 0.21 with range score between 0.21-0.319 and minimum high score 0.32 with range score start from 0.32.

REFERENCE METRIC VALUES [16]

| Metric | Low | Average | High |
|--------------------|------|---------|------|
| LOC=CYCLO/LOC | 0.16 | 0.20 | 0.24 |
| NOM=LOC/NOM | 7 | 10 | 13 |
| NOC=NOM/NOC | 4 | 7 | 10 |
| NOP=NOC/NOP | 6 | 17 | 26 |
| CALLS=FANOUT/CALLS | 0.56 | 0.62 | 0.68 |
| NOM=CALLS / NOM | 2.01 | 2.62 | 3.2 |
| ANDC | 0.25 | 0.41 | 0.57 |
| AHH | 0.09 | 0.21 | 0.32 |

The colouring metrics scores as shown in figure 4 are mapping based on metric categorization as shown in figure 7 where there are 3 metric categorization such as inheritance, size and complexity, and coupling categorization metric. Next are the details.

- Firstly, based on table 1, then Inheritance categorization metric such as ANDC=0.588 and AHH=0.242 as shown in figure 4 or 8 are categorized as high and average scores then they are coloured with orange and green respectively as shown in figure 4.
- Secondly, figure 4 mapping the metrics' scores of size and complexity categorization metric as results from equations (1) to (4) and they are LOC=0.291, NOM=17.467, NOC=13.516 and NOP=50.8 in the left side of figure 4. Based on table 1, these 4 size and complexity categorization metrics are categorized as high score, then all of them have orange colour as shown in figure 4.
- Thirdly, metrics scores of coupling categorization metric as results from equations (5) and (6) and they are CALLS=0.037 and NOM=2.742 in the right side of figure 4. Based on table 1, these 2 coupling categorization metrics such as CALLS=0.037 and NOM=2.742 are categorized as low and average scores, then they have black and green colour respectively, as shown in figure 4.

CONCLUSION

This paper presents how to perform measurements based on Object Oriented Metrics for PHP programming language, many tools that can be used to measurements for this paper used PHP_depend and source codes of Request online system application for testing, PHP_depend experiments generate summary XML, Pyramid report and Report Charts. The conclusion of after testing we suggest Request online

application system developed using PHP framework such as Codeigniter Framework, Laravel Framework or Zend framework, so we get a system that is more reliable and easier to develop. For PHP_Depend we suggest for Pyramid report and Chart Report continue to be developed to make it more easier to understand for user.

REFERENCES

- [1] G. Kour, A. Laws, and S. Evolution, "Using Lehman ' s Laws to Validate the Software Evolution of Agile Projects," pp. 716–722, 2016.
- [2] J. Al Dallal, "Measuring the discriminative power of object-oriented class cohesion metrics," *IEEE Trans. Softw. Eng.*, vol. 37, no. 6, pp. 788–804, 2011.
- [3] D. Arora, P. Khanna, A. Tripathi, S. Sharma, and S. Shukla, "Software Quality Estimation through Object Oriented Design Metrics," *Int. J. Comput. Sci. Netw. Secur.*, vol. 11, no. 4, pp. 100–104, 2011.
- [4] M. Bansal and C. P. Agrawal, "Critical Analysis of Object Oriented Metrics in Software Development," 2014 Fourth Int. Conf. Adv. Comput. Commun. Technol., pp. 197–201, 2014.
- [5] A. Chhikara, R. S. Chhillar, and S. Khatri, "Evaluating the impact of different types of inheritance on the object oriented software metrics," *Int. J. Enterp. Comput. Bus. Syst.*, vol. 1, no. 2, 2011.
- [6] Y. Fitriasia and B. Hendradjaya, "Implementation of ISO 9126-1 quality model for asset inventory information system by utilizing object oriented metrics," *Proc. 2014 Int. Conf. Electr. Eng. Comput. Sci. ICEECS 2014*, no. November, pp. 229–234, 2015.
- [7] M. Glinz, "Software Product Quality," 2014.
- [8] S. Singh and K. S. Kahlon, "Effectiveness of encapsulation and object-oriented metrics to refactor code and identify error prone classes using bad smells," *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 5, p. 1, 2011.
- [9] B. Zeiss and D. Vega, "Applying the ISO 9126 quality model to test specifications," *Softw. Eng.* 2007, vol. 105, pp. 231–244, 2007.
- [10] B. Zeiss, D. Vega, I. Schieferdecker, H. Neukirchen, and J. Grabowski, "An ISO 9126-based Quality Model to Assess the Quality of TTCN-3 Test Specifications 2 . Software Quality Models," www.ttcn-3.org, pp. 1–6.
- [11] J. McCabe, "THOMAS J. McCABE," no. 4, pp. 308–320, 1976.
- [12] G. Kaur and D. Sharma, "A Study on Robert C . Martin ' s Metrics for Packet Categorization Using Fuzzy Logic," vol. 8, no. 12, pp. 215–224, 2015.
- [13] M. Pichler, "Getting started PHPDepend," 2016. accessed from : <https://pdepend.org/documentation/getting-started.html> on 17 Agt 2016.
- [14] M. Pichler, "Overview Pyramid," 2016. accessed from : <https://pdepend.org/documentation/handbook/reports/overview-pyramid.html> on 17 Agt 2016.
- [15] M. Pichler, "Cyclomatic Complexity," 2016. accessed from : <https://pdepend.org/documentation/software-metrics/cyclomatic-complexity.html> on 17 Agt 2016.
- [16] Springer-Verlag Berlin Heidelberg; ISBN 978-3-540-24429-5; Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems; Michele Lanza, Radu Marinescu; 2006
- [17] M. Pichler, "Software metrics supported by PHP_Depend", 2016. accessed from : <https://pdepend.org/documentation/software-metrics/index.html> on 17 Agt 2016.
- [18] M. Pichler, "PHP_Depend", 2016. Accessed from: <https://pdepend.org/documentation/handbook/reports/abstraction-instability-chart.html> on 17 Agt 2016.
- [19] H.L.H.S. Warnars, "Desain model data warehouse dengan contoh kasus perguruan tinggi", *Journal of Industrial Engineering and Management System (JIEMS)*, Vol. 3, No. 1, pp. 9-20, February 2010.
- [20] H.L.H.S. Warnars, "Tata Kelola Database Perguruan Tinggi Yang Optimal Dengan Data Warehouse", *Journal Telkomnika*, Vol. 8 No. 1, pp. 25-34, April 2010.
- [21] H.L.H.S. Warnars, "Perbandingan penggunaan Database OLTP (Online Transactional Processing) dan Data Warehouse", *Creative Communication and Innovative Technology (CCIT) journal*, Vol. 8 No. 1, pp. 83-100, September 2014.
- [22] H.L.H.S. Warnars, E. Suria, D.K. Jeremy, "Pemahaman teori Data Warehouse bagi Mahasiswa tahun awal jenjang strata satu bidang ilmu komputer", *jurnal informatika*, Vol. 13, No.1, Mei 2015, pp. 20-24.
- [23] H.L.H.S. Warnars, R. Randriatoamanana. Datawarehouse: A Data Warehouse artist who have ability to understand data warehouse schema pictures. *IEEE International Conference TENCON 2016 (Technologies for Smart Nation)*, pp. 2207-2210, 22-25 Nov 2016, Singapore
- [24] H.L.H.S. Warnars. Attribute Oriented Induction with simple select SQL Statement. *The 1st Int. Conf. on Computation for Science and Technology (ICCST-1)*, Chiang Mai, Thailand, 4-6 August 2010.
- [25] H.L.H.S. Warnars. Measuring Interesting rules in characteristic rule. *The 2nd International Conference on Soft Computing, Intelligent System and Information Technology (ICSIT)*, pp. 152-156, Bali, Indonesia, 1-2 July 2010.
- [26] H.L.H.S. Warnars. Classification rule with simple select SQL statement. *National seminar Budi Luhur University 2010*, Budi Luhur University, Jakarta, 5 August 2010.
- [27] H.L.H.S. Warnars. Attribute oriented induction with star schema. *Int. Journal of Database Management system (IJDBMS)*, 2(2), 20-42, 2010.
- [28] H.L.H.S. Warnars. Star Schema Design for Concept Hierarchy in Attribute Oriented Induction. *Internetworking Indonesia Journal*, 2(2), 33-39, 2010.
- [29] H.L.H.S. Warnars. Mining Patterns with Attribute Oriented Induction. *The Int. Conf. on Database, Data Warehouse, Data Mining and Big Data (DDMBD2015)*, Tangerang, Indonesia, pp. 11-21, 2015.
- [30] Warnars, H.L.H.S., M.I. Wijaya, H.B. Tjung, D.F. Xavierius, D.V. Hauten, Sasmoko. Easy understanding of Attribute Oriented Induction (AOI) characteristic rule algorithm. *International journal of Applied Engineering Research (IJAER)*, 11(8), 5369-5375, 2016.
- [31] A. Wibowo, H.L.H.S. Warnars. Pengembangan learning characteristic rule pada algoritma data mining attribute oriented induction. *Jurnal Sistem Komputer*, 6(1), 17-29, 2016.
- [32] H.L.H.S. Warnars. Attribute Oriented Induction of High-level Emerging Patterns. *International Symposium on Foundations and Frontiers of Data Mining in conjunction with IEEE Int. Conf. on Granular Computing (IEEE GRC2012)*, Hangzhou, China, 11-13 August 2012.
- [33] H.L.H.S. Warnars. Attribute Oriented Induction High Level Emerging Pattern (AOI-HEP) future research. *The 8th Int. Conf. on Information & Communication Technology and Systems (ICTS)*, Surabaya, Indonesia, pp. 13-18, 24-25 September 2014.
- [34] H.L.H.S. Warnars. Mining Frequent Pattern with Attribute Oriented Induction High level Emerging Pattern (AOI-HEP). *The 2nd Int. Conf. on Information and Communication Technology (IEEE ICICT 2014)*, Bandung, Indonesia, pp. 144-149, 28-30 May 2014.
- [35] H.L.H.S. Warnars. Mining Frequent and Similar Patterns with Attribute Oriented Induction High Level Emerging Pattern (AOI-HEP) Data Mining Technique. *Int. Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)*, 3(11), 266-276, 2014.
- [36] H.L.H.S. Warnars. Using Attribute Oriented Induction High level Emerging Pattern (AOI-HEP) to mine frequent patterns. *Int. Journal of Electrical and Computer Engineering (IJECE)*, 6(6), 3037-3046, 2016.