A Cohesion and Coupling Driven Ontology for Measuring Software Quality at Code Levels

Ezekiel U Okike
University of Botswana, Gaborone, 267, BOTSWANA

Summary
The need for quality and reliable software systems has led to the development of rigorous software measures of quality at code level. One of the basic problems in software measurement is that many of the existing measures do not measure what they claim to be measuring. Chidamber and Kemerer metric suites were used to measure cohesion and coupling in six industrial systems. Findings of the study indicated that cohesion and coupling measured quality at code level in four of the systems in terms of the systems being highly cohesive and low in coupling. Cohesive systems 1, 2, 4, 6 had median values [0,1]. In these systems the level of coupling is acceptably low. In terms of correlations cohesion and coupling, cohesion and size, coupling and size all correlated significantly. The study concludes that cohesion and coupling were useful quality software measures in the studied systems.

. Key words:
Ontology, cohesion, coupling, software quality, measurement.

1. Introduction

Ontologies are models that represent concepts of interest in a domain using acceptable formalism [13,26]. There has been a growing interest in the use of ontologies due mainly to their possibilities in using them to represent knowledge in a structured manner [17]. Apart from knowledge representation and structuring, ontologies have useful applications in computational models, definition of objects and their functions [9]; Browsing and searching of semantic contents, construction of models of theories of domains; organizing contents in digital libraries, databanks, data marts, data warehouses, dictionaries and thesaurus systems, and relational data bases [24]. Furthermore, the role of ontologies in information systems research has been discussed in [20]. Figure 1 illustrates ontological approach in research.

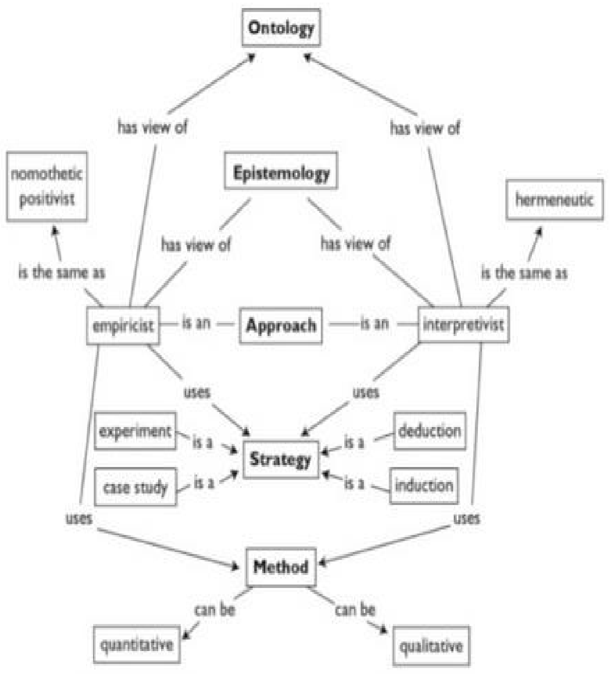

Fig.1. Ontology research method. Source: http://research methology.net

From figure 1, research method can be quantitative or qualitative or mixed; a research strategy can be case study, experiment or deductive, inductive; a research approach can be empiricist or interpretivist either of which has a view of Ontology or epistemology.
One area of interest in the application of ontologies is the measurement and evaluation of software quality at code level. This study is concerned with the application of an ontological approaches in software quality evaluation at code level using cohesion and coupling measures.

Manuscript received December 5, 2020
Manuscript revised December 25, 2020
https://doi.org/10.22937/IJCSNS.2020.20.12.27

1.1 Understanding Software Complexity and Measurement

Generally, software is a computer program written using appropriate programming language such as JAVA, C++, PYTHON, etc. There exist hundreds of programming languages to date as computer programming languages are dynamically evolving with the application and the technology of the day. Despite this development, software complexity remains the same as any other object in real life. As shown in figure 2, there are seven sources of software complexity. Of these components, cohesion and coupling have been identified as two dominant dimensions of software complexity [8].

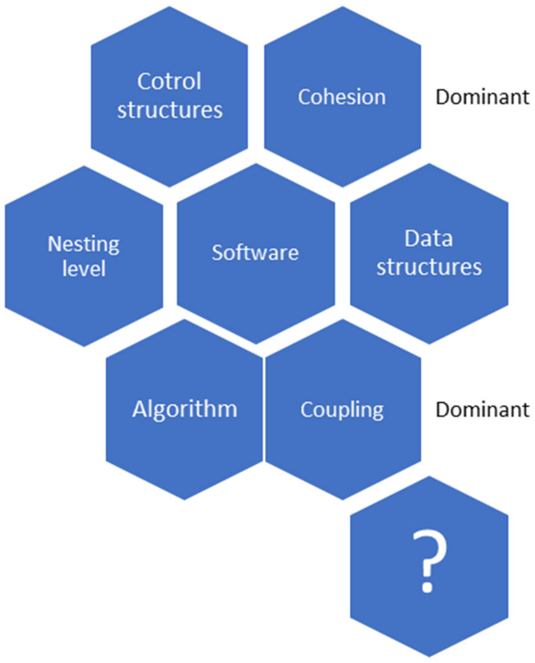

Fig. 2. Dimensions of software complexity

From figure 2, these dimension are internal attributes of software, usually measured as code level attributes. The internal quality of software products have no inherent, practical meaning within themselves except they are characterized in terms of the external quality of a software product. This is shown in the software quality model represented in figure 3.

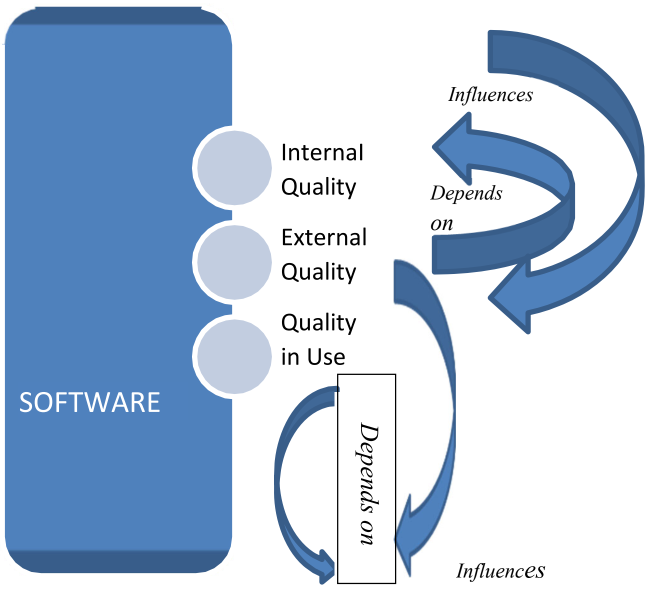

Fig. 3. Software Quality

From figure 3, the internal quality of software (Software, Cohesion, Coupling, Data Structures, Algorithms, Control structures, Nesting Level) influence the external quality which in turn depends on the internal quality. It is also evaluated by the internal/external complexity of a design. Similarly, the external quality of software influences the quality in use which in turn depends on the external quality. It is evaluated by maintainability, testability, reusability of design.

This study is concerned with measuring software quality at code level using Cohesion and Coupling. The empirical study is based on six industrial systems developed using Java programming language.

1.1 Statement of the Problem.

One of the basic problems in software measurement is that many of the existing measures do not measure what they claim to be measuring [22]. This situation is due largely to a poorly intuitive understanding of the concept of the software attributes being measured as well as the general lack of proper application of a rigorous approach to software measurement based on sound measurement theory.

1.2 Objective.

The objective of this paper is to apply a cohesion and coupling based ontological models in the measurement of module cohesion and module coupling in order to determine software quality at code level in six studied industrial systems. The term module refers to a class in the object oriented paradigm.

The specific objectives of this paper are

i. To demonstrate how cohesion and coupling metrics measure software quality at code level.
ii. To demonstrate the relationship between cohesion and coupling as attributes of software quality at code level.
iii. To investigate the relationship between cohesion, coupling and size measures in software quality

1.3 Research Questions

i. Does cohesion and coupling measure software quality at code level?
ii. Do cohesion and coupling correlate?
iii. Do cohesion, coupling and size measures correlate?

2. Measuring Software Complexity in terms of Cohesion and Coupling

Measurement is a process of assigning numbers or symbols to attributes of entities in the real world so as to describe them clearly.

2.1 Module cohesion

According to [26], Cohesion refers to the functional relatedness in software modules. Other terms sometimes used to denote the same concept are “module strength”, “binding”, and “functionality” [19]. Therefore, Cohesion as a software attribute captures the binding strength of elements in a module, or class.

2.2 Module coupling

In structural design, a large software system is usually partitioned into manageable units (modules) to make each small unit independent. Hence, coupling refers to the degree of independence between two modules. A desirable attribute of software is low coupling between modules which indicate a well partitioned system [22]

2.3 Measuring cohesion and coupling in object-oriented systems

This study is concerned with the measurement of class cohesion and coupling in the object oriented paradigm. A number of metrics for example [1, 2,3,4,5,6,7,8,12,13,15, 18,24,26] measuring cohesion and coupling exist in the literature. The major existing coupling Object oriented metrics are Coupling Between Objects (CBO), Response for a class (RFC) [7,8,15], Afferent coupling (CA). Other coupling measures are variants of CBO, RFC and CA. These include Conceptual Coupling Between Classes (CCBC), Conceptual Coupling between Methods (CCM), Coupling Between Methods and a Class (CCMC), Structural Coupling, Conceptual coupling [6]. All of these studies were inspired by the initial work of Chidamber and Kemerer) who defined the Lack of Cohesion in Methods (LCOM) metric for OO systems . To date the most acknowledged Object Oriented specific metrics are the Chidamber and Kemerer suite of metrics [7,8].

3.0 The Lack of Cohesion in Methods (LCOM) metric

Definition1.

Given a class C, with methods M, and instance variables I (∀ M,I= 1:n),
Then

Let P = $\{(Ii.\, I_j|\ Ii \cap Ij = \Phi\}$ and Q = $\{(Ii.\, Ij\ )|\ Ii \cap Ij \neq \Phi\}$
LCOM= |P|-|Q| If |P| > |Q|

$= 0, otherwise$ (1)

Practically, the metric identifies the number of method pairs in a class with zero (0) or null minus the number of methods pairs whose similarity is not zero.

Let C2 be a class with two methods M1 and M2, with instance variable I1 and I2 then degree of similarity of method β;

$$\beta = \{I_1\} \cap \{I_2\} .$$

This definition of LCOM was refined in LCOM2 to include inherited methods and attributes as follows [14]:

Definition 2.

$$Let\ P = \Phi,\ if\ AR(M) = \Phi\ \not\forall\ \forall \wedge \vee \cup \in \forall m \in Mi\,(\mathrm{c}\,)$$
$$= \{\,(m1.m2j\,) \dashv |\ m1, m2 \in Mi\ \wedge m1 \neq m2 \wedge AR(m1) \cap AR(m2)\ \cap Ai(c\,) = \Phi\}$$

else

$$Let\ Q = \{\,(m1.m2j\,)\,|\,m1, m2 \in Mi(c\,) \wedge m1 \neq m2 \wedge AR(m1) \cap AR(m2) \quad \cap Ai(c\,) \neq \Phi\}$$

Then

$$LCOM2 = \{|P| - |Q| \text{ If } |P| > |Q|$$
$$= 0, otherwise \qquad (2)$$

Note: Mi (c) are the methods in class c, Ai (c) are the attributes (instance variables) in class c, and AR represent attribute reference. There is enough evidence which indicates that LCOM = [0,1] indicates a cohesive class [3,2,20]. In addition, designing classes with less than 5 methods have proved to yield cohesive values, while classes with up to 5 or more methods needs to be split into 2 or more lasses to make them cohesive and hence achieve a well-designed class [12].

Cohesiveness and coupling are interconnected such that as one increases, the other decreases. Therefore, as a way of evaluating the effective use of software, designers generally try to achieve high cohesion and low coupling. A highly cohesive module is one which has a single basic function and is difficult to split. Seven levels of cohesion were discussed in [20] from the least functionally cohesive to the best functionally cohesive.

4. The Empirical Study

4.1 Methodology

Six different Java based programs developed for industrial application were used in this study. The Java codes contained 3254 classes, 503986 attributes, 249179 methods and 15476 public methods. The codes were developed by different people in different places and domains. Chidember and Kemerer metric tools were used to measure cohesion and coupling.

Accordingly, LCOM and NLCOM were defined for this study as follows:

$$LCOM = \{|P| - |Q| \text{ If } |P| > |Q|$$
$$= 0, otherwise$$

$$NLCOM = \frac{1}{LCOM}, Otherwise \qquad (3)$$

where LCOM is as defined in LCOM2 (equation 2 above), and NLCOM the normalized measure of LCOM as defined in equation 3. The variables used in this study are shown in Table 1, while the characteristics of the systems are shown in Table 2. The metric calculation process is shown in figure 2.

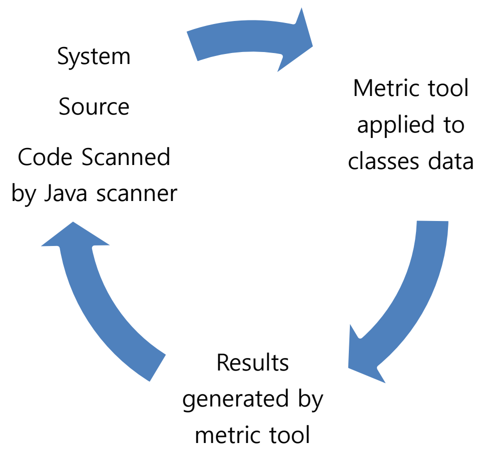

Fig. 2 Metric calculation process

Table 1. Metric variables used in this study

Metric	Meaning	Attribute Measured	Source
LCOM	Lack of Cohesion in Methods	Cohesion	Chidamber & Kemerer (1994)
NLCOM	Normalized LCOM	Cohesion	Okike (2007)
CBO	Coupling between Objects	Coupling	Chidamber & Kemerer (1994)
RFC	Response for a Class	Coupling	Chidamber & Kemerer (1994)
CA	Afferent Coupling		Spinellis (2005)
WMC	Weighted Methods per class	Size	Chidamber & Kemerer (1994)
NOC	Number of Children	Size	Chidamber & Kemerer (1994)
DIT	Depth of Inheritance		Chidamber & Kemerer (1994)

Table 2. Characteristics of the selected systems

Systems	1	2	3	4	5	6	Total
Classes	34	4	318	383	1055	1460	3254
Attrib.	4	4	2414	233	231110	270221	503986
Methods	30	10	3388	685	22000	223066	249179
NPM	21	7	2536	266	6232	6414	15476
Size(KB)	3.9	1.1	172.9	180	770	1030	2157.9

4. Results and Discussion

Using descriptive statistics and correlation analysis the following findings emerged from this study:

4.1 Descriptive statistics for the test systems

From Table 3, well designed systems are highly cohesive and have median values [0,1]. Similarly, systems exhibiting low coupling imply good design, and hence, quality software. Systems 1,2,4, and 6 satisfy these conditions. A close observation shows that for these systems, cohesion (LCOM) range [0,1]. This means that cohesion measures software quality (research question 1). In terms of coupling, systems 1,2,4 and 6 CBO values range between 3 and 5; while CA values range between 0 and 3. Hence, coupling is low in the systems.

4.2 Correlation Analysis for the test systems

Using systems 3, 4, 5, and 6 correlation analysis was performed in order to verify the relationship between cohesion and coupling research question 2 and 3:
Tables 4, 5, 6, and 7 present the results of correlation analysis for systems 5, 6, 4 and 3 respectively. Correlation is significant at 0.01 level or 0.05 level. This study adopted the 0-01 level of significance as appropriate for software systems involving human activity as developers [20]. In all the systems, cohesion and coupling are significantly correlated (research question 2]. This implies that high cohesion implies low coupling (see also Table 3). This agrees with previous studies such as [3,2]. In terms of size measures (WMC, NPM), cohesion and size are significantly correlated in all the systems. There is also significant correlation between coupling (CBO, RFC, CA) and size (WMC, NPM) in all the systems

Manuscript received December 5, 2020
Manuscript revised December 25, 2020
https://doi.org/10.22937/IJCSNS.2020.20.12.27

Table 3. System Statistics

Systems	No of classes	Stat.	LCOM	N LC M	CBO	RFC	CA	W M C	D I T	N O C
Sys 1	34	Min	0		0	0	0	0	1	0
		Max	2534		31	129	22	74	2	8
		Mean	79.0		5.6	22.2	4.2	7.9	1.4	.4
		Med	0	0	5.0	18.5	2.0	4.0	1.0	.0
		StD	440.2		5.9	24.2	4.9	13.0	.5	1.5
Sys2	4	Min	0		0	0	I	0	1	0
		Max	8		5	71	5	11	4	0
		Mean	1.8		2.0	34.2	2.3	4.8	1.8	.0
		Med	1.0		1.5	33.0	1.5	4.0	1.0	.0
		StD	2.4		2.5	36.0	1.9	4.7	1.5	.0
Sys3	318	Min	0		0	0	0	0	1	0
		Max	6075		16	182	10	118	41	3
		Mean	93.9		0.8	18.2	.6	9.7	1.2	4.0
		Med	3.0		0.0	7.0	.0	5.0	1.0	.0
		StD	490.0		2.2	28.1	1.5	12.5	2.3	.25
Sys4	383	Min	0		0	0	0	0	1	0
		Max	16290		195	265	157	118	5	36
		Mean	150.4		8.3	20.9	5.7	8.2	2.1	.6
		Med	1.0		5.0	10.0	3.0	3.0	2.0	.0
		StD	1318		20.0	31.1	14.	19.2	1.2	3.0
Sys5	1055	Min	0		0	0	0	0	1	0
		Max	2744		65	210	71	109	4	64
		Mean	44.9		6.25	26.5	1.7	8.0	1.4	.4
		Med	6.0		3.0	16.0	.0	5.0	1.0	.0
		StD	180.5		7.6	30.9	5.8	9.4	.6	3.0
Sys6	1460	Min	0		0	0	0	0	1	0
		Max	9870		56	270	50	141	5	35
		Mean	25.3		4.3	15.9	1.9	5.5	1.3	.2
		Med	1.0		3.0	8.0	1.0	3.0	1.0	.0
		StD	283.0		5.4	20.3	3.8	8.2	.6	1.4
Total	3254									

Table 4. System 5. (N= 1055)

Metric	a	B	c	d	E	f	g	H
WMC a	1.000	-.104**	.102**	.420**	.758**	.765**	.212**	.846*
DIT b	-.104 **			.077*			-.136**	
NOC c	.102**			.062*		.077*		.114**
CBO d	.420**	.077*	.062*		.793**	.298**		.270**
RFC e	.758**			.793**		.610**	.084**	.572**
LCOM f	.765**		.077*	.298**	.610**		.090**	.654**
CA g	.212**	-.136**	.518**		.084**	.090**		.188**
NPM h	.846**		.114**	.270**	.572**	.654**	.188**	

Pearson Correlation at 0.01**, 0.05* levels (2 tailed)

Manuscript received December 5, 2020
Manuscript revised December 25, 2020
https://doi.org/10.22937/IJCSNS.2020.20.12.27

Table 5. System 6. (N= 1460)

Metric	A	B	c	d	E	F	g	H
WMC a	1.000		.094**	.309**	.833**	.625**	.212**	.846*
DIT b				.080**			-.136**	
NOC c	.094**				.065*			.114**
CBO d	.309**	.080**			.661**	.091**		.270**
RFC e	.833**		.065*	.661**		.385**	.084**	.572**
LCOM f	.625**			.091**	.385**		.090**	.654**
CA g	.243**	.065*	.497**	.231**	.252**	.089**		.188**
NPM h	.967**		.096**	.238**	.759**	.625**	.188**	

Pearson Correlation at 0.01**, 0.05* levels (2 tailed)

Table 6 System 4 (N=383)

Metric	a	b	c	d	e	f	g	h
WMC a	1.000	-.211**		.889**	.849**	.822**	.268**	.992**
DIT b	-.211 **			.131*	-.268	-.105	-.130*	-.191**
NOC c	.102**						.249**	
CBO d	.889**	.131*			.807**	.782**	.225**	.892**
RFC e	.849**	-.268**		.807**		.547**	.144**	.813**
LCOM f	.822**	-.105*	.249**	.782*	.547**		.332**	.843**
CA g	.268**	-.130*		.225**	.144**	.332**		.274**
NPM h	.992**	-.191**		.892**	.813**	.843**	.274**	

Table 7. System 3. (N= 318)

Metric	a	b	c	d	e	f	g	h
WMC a	1.000			.265**	.792**	.810**		.786**
DIT b								
NOC c								
CBO d	.265**				.716**		.281**	
RFC e	.792**			.716**		.571**	.211**	.523**
LCOM f	.810**				.571**			.343**
CA g				.281**	.211**			
NPM h	.786**		*		.523**	.343**		

Pearson Correlation at 0.01**, 0.05* levels (2 tailed)

5. Conclusion

In conclusion, measuring software quality at code level using cohesion and coupling offers many advantages. It assists in understanding the software and its operations from the code view which gives confidence about its likely operations when deployed in the user environment. Code level quality measurements satisfy the expectation of rigorously building software that is reliable, reusable, maintainable and robust. At design level, software developers with the aid of code level measurements easily identify poorly designed classes (modules) and are able to fix such issues for better working of the software. Therefore, this study underscores the need for software measurement in the production of reliable, reusable, maintainable and robust software systems.

References

.
[1] H. Aman, K. Yamasski,, M.A.Noda, :A proposal of class cohesion metrics using sizes of cohesive parts. Knowledge based software engineering. T. Welzer et al. eds. IOS press102-107 (2002)

[2] L. Badri, and M. Badri, " A proposal of a new class cohesion criterion:an empirical study," Journal of object technology, 3, 4:145-159 (2004).

[3] J.M. Bieman and B. K Kang, "Cohesion and reuse in an object oriented system," Proceedings of the symposium on software reusability (SSR'95), Seattle:WA. 259-262 (1995).

[4] J. M. Bieman, and B. K. Kang, "Measuring Design-level Cohesion. IEEE Transactions on Software Engineering," 24(2), 111–124 (1998).

[5] Candela, I. : Using cohesion and coupling for software modularization: Is it enough? ACM transactions on software engineering and methodology, vol. 25, no.3, article 24:1-24 (2016).

[6] S. R. Chidamber and C. F. Kemerer, " A Metric Suite for Object Oriented Design," IEEE Transactions on Software Engineering 26, 6:476-493 (1994)

[7] P.D. Darcey, C. F. Kemerer, S. A. Slaughter, J. E. Tamayko, M. Farelo, and C Morris, "The structural complexity of software: An experimental test," IEEE transactions on software engineering, 32,1:54-64 (2005).

[8] T.Gruber, Ontology. In. L.L.Ozsu (ed), encyclopedia of databasesystems. Springer-verlag (2009).

[9] B. Henderson-Sellers, Software metrics. U.K:Prentice-Hall, (1996)

[10] M. Hitz and B. Montazeri, "Chidamber and Kemerer's Metric Suite: A Measurement Theory Perspective," IEEE Transactions on Software Engineering 22, 4:267-270 (1996).

[11] M. Z. Jiang, E. A. Hassan, and C. R Holt, "Visualizing clone cohesion and coupling. Proceedings of xiii Asia Pacific software engineering conference (APSEC'06), IEEE computer society 0-7695-2685-3/06^(2006).

[12] Y. Kang, "Ontologies for crisis contagion management in financial institution,". Journal of Information science 35(5) 548-562 (2009).

[13] Y. Lee, B. Liang, and F. Wang, " Measuring coupling and cohesion of an object oriented program based on information flow," Proceedings of the international conference on software quality, Maribor, Slovenia (1995)

[14] T. M. Meyers and D. Binkley, "An empirical study of slice-based cohesion and coupling metrics," ACM Trans. Software. Eng. Methodology, 17, 1, Article 2 (December2007) 27 pages.

[15] P. Mitzias, R. Marina, E. Kontopoulos, T. G. Stavropoulos, S. Andreadis,, G. Meditskos, and I. Kompatsiaris, User-driven ontology population from linked data. http://www (2005). Last accessed 7 November, 2020

[16] S. Oh, H. Y. Yeom, and J. Ahn, "Cohesion and coupling metrics for ontology modules,". Inf Technology Management, **12,** 81, (2011)

[17] L. Ouyang, B. Zou, M. Qu, C. Zhang, " A method of ontology evaluation based on coverage of cohesion and coupling," 11th (FSKD) conference, .IEEE, 2451-2455 (2011)..

[18] E.U. Okike, Measuring class cohesion in object-oriented systems using Chidamber and Kemerer metric and Java as case study. Ph.D Thesis, Department of Computer S, University of Ibadan (2007)

[19] E. U. Okike, T. Motshegwa T and M.N. Kgobathe, "Ontological perspectives in Information systems security and computer attack incidents (CERTS/CIRTS)," Proceedings of the 1st international conference on the internet, Cyber Security, and Information Systems (ICICIS), 46-60 (2016).

[20] L. M. Ott,(2005). Software measurement. Accessed 15 December, 2005. http://www.cs.mtu.edu/~linda/soft.html

[21] M. Page-Jones, The Practical Guide to Structured Systems Design (Yourdon Press Computing Series, Prentice Hall, Englewood Cliffs, New Jersey).

[22] M. Paixao, M, Harman, Y. Zhang, Y. Yu, " An empirical study of cohesion and coupling: balancing optimization and disruption,". IEEE transactions on evolutionary computation, vol. 22, no. 3 (2018).

[23] T. Pierera, H and H. Santos, "An ontology based approach to information security," . F. Sartori., M. A. Siccilia., N. Manouselu (eds): MTSR CCIS, 46,183-192(2009).

[24] H. A. Reijers and I. T.. P Vanderfeesten , Cohesion and Coupling Metrics for Workflow Process Design. In: Desel J., Pernici B., Weske M. (eds) Business Process Management. BPM 2004. Lecture Notes in Computer Science, vol 3080. Springer, Berlin, Heidelberg

[25] P . Silvonen, Ontologies and knowledge base. http://www.ling.helsinki.fi/~stviitan/documents/ontologies_and_kb/ontology.ht ml. Last accessed22 October, 2015.

[26] E.Yourdon, I. I, Constantine, Structured design: fundamentals of a discipline of computer program and systems design. Englewood cliff, New Jersey: Prentice-Hall, (1979)

Ezekiel U. Okike received the BSC (Computer Science), Master of Information Science and PhD (Computer Science) from University of Ibadan, Nigeria in 1992, 1995 and 2007 respectively. He is currently the cluster chair of Information Systems Cluster, Department of Computer Science, University of Botswana. He is a Senior Member of IEEE, and a Member of ACM. His research interests are Software Quality, Models and Architectures; Software Measurements; Information Systems; Software Engineering; Machine Learning; Information Security /Cyber Security, E-Systems.