An Ontology Model for Software Measurement in Software Project Management

Ezekiel U Okike[1], and Ofaletse Mphale[2]
Department of Computer Science, University of Botswana, Gaborone, Botswana[1,2]
euokike@gmail.com[1], ofaletse_offie@hotmail.com[2]

Abstract

Purpose : This paper demonstrates an approach to Software Project Management measurement based on ontology.

Design/Methodology/Approach: The study adopted measurement theory approach in the definition of models of Software Project Management measures of success/failure

Findings: Using this approach, the resulting model was suitable for classifying and categorizing project success measures into ideal success, acceptable tolerance, and unacceptable tolerance

Practical Implication: The paper contributes to project success/failure measurement based on measurement theory

Originality/Value: The study is an application of measurement theory in measuring project success using ontology approach

Key words: Ontology, Software measurement, project management, measurement theory

Introduction

An ontology is a formal approach to specifying a concept and its representation of a domain of interest (Kang Ye etal, 2009). The concept under consideration is explicitly described and represented as appropriate using formalisms or other representation systems. In this description, properties or features of the concept are identified; also relative concepts are discovered as well as and the relationships between the concepts (Silvonen, 2002).

Using ontologies knowledge have been semantically structured in philosophy and the metaphysics disciplines, computational models have been created in Artificial intelligence for automated reasoning, and classes, relations, functions and other objects have been defined (Gruber, 2009). Pereira and Santos (2009) suggested that ontologies support browsing and searching of semantic contents, promote interoperability for facilitating knowledge management and configuration, and assist in the construction of models or theories of domains. Other uses of ontologies include sharing common understanding of objects, enabling knowledge reuse, enabling explicit assumptions, separating and analysing domains, and organising contents in knowledge based Information systems and supportive components such as libraries/digital libraries, data centres, data banks, relational data bases, data ware houses, data marts, dictionaries and thesaurus systems.

Okike, Motshegwa and Molly (2016) suggested the useful applications of ontologies in Information systems from three perspectives namely: Information systems research, Information systems development, and Information systems security.

In research for instance, an ontological approach is utilized when a researchers considers the choice of a research method as shown in figure 1. From figure 1, a research method is either qualitative or quantitative or both, and the strategy can be a case study or an experiment on the one hand; or a deduction or induction on the other. The approach is either empiricist or interprecivist. All of the stages build up to an ontology.

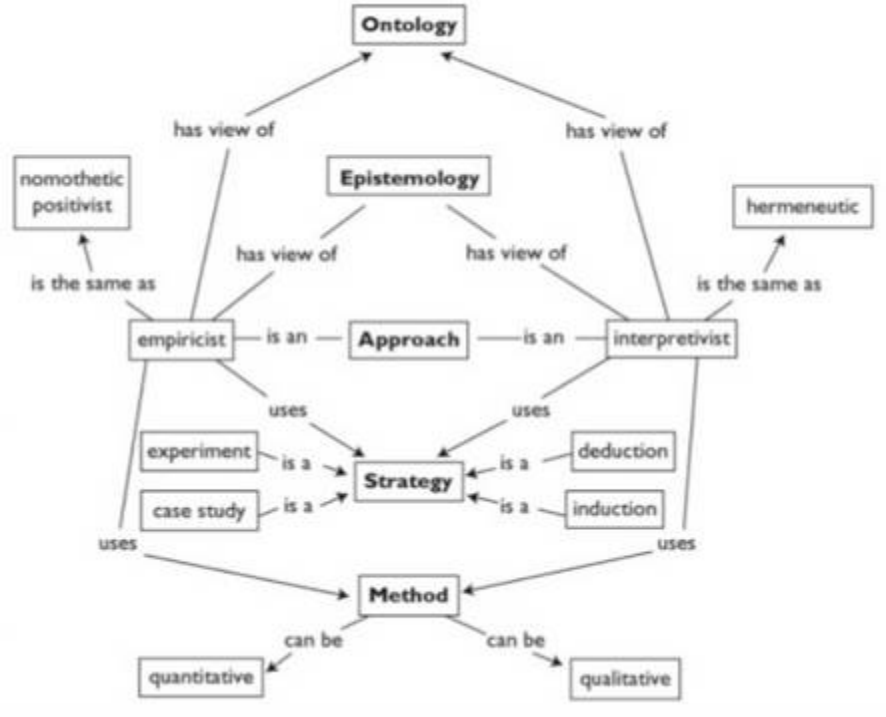


Fig. 1. Impact of ontology in the choice of research methods
(Source:http://research methodology.net/)

In this paper, we discuss the application of ontology in the software measurement domain in order to address crucial issues pertinent to measurements in Software Engineering and Information systems.

Overview of Software Measurement and Measurement Theory

Measurement

Measurement is "a mapping from the empirical world to the formal, relational world**.** Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute" (Fenton & Pfleeger (1997:32). For instance consider two people: James who is 5ft tall, and John, who is 4ft tall. We can establish a representative measure of the heights of James and John by considering two system, namely- the real world and the number systems as illustrated in Table 1below

Table 1. Real world vs number system

Real word system	Number system
James	5ft
John	4ft

Furthermore, define a mapping function *M* which maps the real world system into a number system as:

M: R→N (R is Real world system; N, Number system) 2.17(1)

Let *M* be a mapping with R as its domain, and with its range contained in N. Let r∈*R*, then there is a unique n∈ *N* such that *rMn* holds.

Hence, we can define different mapping functions for James and John, from the real world to a number system which specifies their heights as shown in table 1:

M(James)*: R→N* and *M*(John) *: R→N* (2)

By substituting their heights in (2), we obtain:

M(5), and *M*(4) (3)

We can define a relational system which represents this mapping using the "taller than" relation in the real world, and "greater than", "less than" relations in numerical system. Hence, "James is taller than John" implies that:

M(James) > M(John) (4)

From (1), (2),(3) and (4), measurement involves a representation and also the conditions under

which the measurement is satisfied as obtains in the measurement theory.

Software Measurement and Measurement Theory

Software measurement involves three software activities namely:

i) Processes. These are collections of software related activities such as analysis, design, coding, testing, installation etc.

ii) Products. These are artefacts, deliverables or documents resulting from process activities

iii) Resources. These are entities required by a process activity. Example includes personnel, budgets, computer hardware, other stationaries.

Processes may be measured using appropriate Software Project Management tools and estimation techniques, while Software products may be measured from 2 essential perspectives – internal attributes and external attributes. Internal attributes are measured in terms of the product itself. These attributes include cohesion, coupling, nesting level, data structures, algorithms, and the software itself. External attributes are measured in terms of how the product, process or resource relate to the environment. Examples of external attributes of software include usability, reliability, efficiency, reusability, maintainability, portability, testability etc

Measurement theory specifies the rules for developing and reasoning about all kinds of measurement (including software measurement/measurements in Information systems). Rule based approaches are common in the sciences . For instance, Mathematicians learned about the world by defining axioms for a geometry; by combining axioms and using their result to support or refute their observations, they expanded their understanding and the set of rules that govern the behaviour of objects.

As explained in Okike (2007), the obligation of any software measurement activity is to identify the entities and attributes to be measured. Baker et al (1990), suggested the three approaches of measurement theory as follows:

i) Determine the axioms that capture intuitive understanding and empirical observations about the attribute(s) of an entity of interest

ii) Apply the representation theorem of measurement to show that the attribute can be appropriately represented in a number by a mapping which preserves the axioms

iii) Apply the uniqueness theorem to show that any two functions defined from the set of entities to the set of numbers faithfully represent the attribute, and they are related

According Ralph (2005), one reason for "moving software development into the engineering arena" is to be able to control the process, and to control a process, it must be measurable''. However, a measure is valid only when we are able to verify that the number is representative of the attribute that is being measured. By applying rule based ontology, we can validate a measurement approach as demonstrated in this paper.

Ontology Based Software Measurement Models

An Ontology Based Cohesion Model

The term cohesion is defined as the degree to which the elements in a module belong together. In the Object Oriented paradigm it refers to the degree of relatedness or consistency in functionality of members in a class (Yao, Orme, and Etzkorn, 2005). Hence, it is a measure of how tightly bound or related the internal elements of a software systems are. This measure captures the degree of association of elements within a module, and the programming paradigm used determines what is an element and what is a module. Many researchers proposed cohesion measures for software in the procedural, Object Oriented and lately in the Aspect Oriented paradigm. (Bieman and Ott (1994 ; Bieman and Kang (1998) Chidamber and Kemerer , 1994), Hitz and Montazery (1996), Badri and Badri (2004), Gupta 1997, Okike (2007, 2008, 2010)).

Using data from Okike (2007), cohesion is measured using the Lack of Cohesion in Methods (LCOM) proposed by Chidamber and Kemerer (1994) and its adjusted interpretation as proposed in Okike (2007). Hence LCOM is defined

Lack of Cohesion in Methods (LCOM)

Consider a class C1 with n methods M1, M2,…,Mn. Let{Ii}= set of instance variables used by method Mi. There are n such sets {Ii},…,{In}.

Let P = { (Ii, Ij) | Ii ∩ Ij = ϕ}, and Q = { (Ii, Ij) | Ii ∩ Ij ≠ ϕ }. If all n sets { I1}, …,{In}

are ϕ then let P = ϕ

LCOM = { |P|- |Q|, if |P| > |Q|

= 0, otherwise

Example: Consider a class C with three methods M1, M2

and M3. Let {I1} = {a,b,c,d,e} and {I2} = {a,b,e} and {I3}= {x,y,z}.

{I1} ∩ {I2} is nonempty, but {I1} ∩ {I3} and

{I2} ∩ {I3} are null sets.

LCOM is (the number of null intersections – number of non empty intersections), which in this case is 1.

The theoretical basis of LCOM uses the notion of degree of similarity of methods. The degree of similarity of two methods M1 and M2 in class C1 is given by: σ( ) = {I1} ∩ {I2}

where {I1} and {I2} are sets of instance variables used by M1 and M2 . The LCOM is a count of the number of method pairs whose similarity is 0 (i.e, σ( ) is a null set) minus the count of method pairs whose similarity is not zero.

- The larger the number of similar methods, the more cohesive the class, which is consistent with the traditional notions of cohesion that measure the inter relatedness between portions of a program.
- If none of the methods of a class display any instance behaviour, i.e. do not use any instance variables, they have no similarity and the LCOM value for the class will be zero.
- The LCOM value provides a measure of the relative disparate nature of methods in the class. A smaller number of disjoint pairs (elements of set P) implies greater similarity of methods. LCOM is intimately tied to the instance variables and methods of a class, therefore is a measure of the attributes of an object class

Definition 2

A refinement of previous definition of LCOM to include inherited methods and attributes was proposed by Hitz and Montazeri(1996)

Let P = ϕ, if AR (m) = ϕ ∀ m ∈ M**I** (c)

= {{m1,m2} | m1,m2 ∈ M**I**(c) ∧ m1 ≠ m2 ∧ AR(m1) ∩ AR(m2) ∩ A**I** (c) = ϕ }, else

Let Q = {{ m1,m2} | m1,m2 ∈ M**I**(c) ∧ m1 ≠ m2 ∧ AR (m1) ∩ AR(m2) ∩ A**I**(c) ≠ ϕ }

Then LCOM2(c) = {|P| - |Q|, if |P| > |Q|

= 0, otherwise

LCOM2 of many classes are set to zero although different cohesions are expected

Let P = (n- 1) (1)

Q = n-1 (2)

So that LCOM

$$P - Q = \frac{n!}{(n-2)!2!} - 2(n-1)$$

(3)

From (3), for n<5, LCOM =0;

for n≥ 5, 1<LCOM < n

Proposal of Ontology based Project Management Model

Measuring Project Success

Let *S* be the set of project success measures, and *R* be the set of real numbers. A project success measure *M* is a function from *S* to R, whose mapping is denoted:

M: S→R

Assuming for a set of objects *O* which represents a set of relations *r*, an attribute *A* (potentially possessed by each member of object O) is identified. Suppose A induces a set of empirical relations $R_1,R_2.R_3…R_n$ on object O, then denote this as $r(R_1,R_2.R_3…R_n)$

Let <r,R> be the ordered pair of interest. To measure the attribute *A*, a mapping from the set of objects O possessing the attribute A into a number system is required.

Let *R* the set of real numbers be the number system.

Let N ⊆ R be a set

Let P = $< P_1,P_2,P_3…P_n>$ be a set of relations on N, then

K = <N,P>

Suppose N=R, then several empirical relations, and numerical relations could result upon which the representation condition for a measure *M* for attribute *A* resulting into a map where M maps objects in O (the set of relations r) to elements in N (the real number system R) may apply

The following relations may result from the rule:

$P_1 \in R \times R$: "$(\chi,y) \in P_1$ if $\chi > y$"

$P_2 \in R \times R$: "$\chi \in P_2$ if $\chi > 70$" (i.e $y = 70$ represents some threshold number)

$P_3 \in R \times R$: "$\chi \in P_3$ if $\chi > y \pm 15$" (i.e 15 represents some number, $\pm$ the

Threshold

Empirical Application

Using data from ICT projects collected in Botswana to measure Project success /Failure Mpale (2016), established success thresholds or categorization to include the ideal success at 46/46 (100% fulfilment of success measurement criteria), acceptable success at 36-45 of all success elements fulfilled out of 46 (78-99%) satisfaction, and unacceptable tolerance at less than 36 of success elements fulfilled out of 46 elements. The 46 are the identified success/ failure project measurable criteria as shown in Table 2. Table 3 illustrates this success into their categories. The scenarios explained in the relational model above agrees with the success categorization principle . The result may represented on a success/failure number line as shown in figure 2

Table 2. Success or failure project measurable criteria

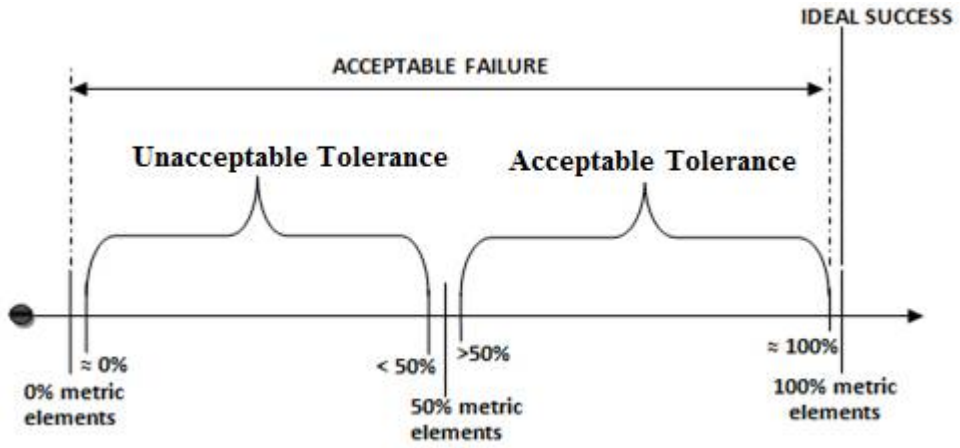
(A) Technological	**(B) Human resource**
A1 - IT functionality/ Capabilities	B1 – Use of Consultants
A2 - Ease of use/ quantity of use	B2 - Project Management
A3 - Happiness/willingness of end users	B3 - Project Manager experience
A4 - Technology and Technological issues	B4 - User training, education and support
A5 - Software development Methodology	B5 – Project Champion
A6 - Software prototyping and testing	B6 - Commitment
A7 - Vendor capabilities	B7 - Cooperation
A8 - Outsourcing strategy	B8 - Productivity
A9 - Implementation strategy	B9 - Empowerment
A10 - IT solved problem(s) that was intended to solve	B10 – Core competency
A11 - Software quality	B11 – Flexibility
A12 – Safety	
(C) Organisational	
C1 - Top management support	C13 – Leadership style
C2 - Project schedule	C14- Stakeholder management
C3 - Project Time	C15 - Security strategy
C4 - Project cost	C16 - Business process re-engineering
C5 - Project accuracy(specifications met)	C17– Organisational Benefits
C6 – Requirement Management	C18- Process improvements
C7 - Change management	C19 - Manual process intervention
C8 - Cultural management	C20 - Operating efficiencies
C9 - Quality management	C21 - Resource Management
C10 – Financial resources	C22 – Tracking of issues since implementation
C11 – Expectations Management	
C12 - Business plan and vision	C23 - Business growth support
Total metrics elements: 46	

Source. Mphale (2016)

TABLE 3: Project Success And Categorization

Organisations rankings	**Summary of project success factors ratings measured up against the Metric model**	**Metric tool success categorisation**
Company A	40/46 (86.9%)	Acceptable success
Company B	36/46 (78'2%)	Unacceptable success
Company F	34/46 (73.9%)	Unacceptable
Company C	33/46 (71.7%)	Unacceptable
Company E	31/46 (67.3%)	Unacceptable
Company D	30/46 (65.2%)	Unacceptable

Note: Ideal Success (46 All elements fulfilled) Acceptable tolerance (45 –36 elements fulfilled) Unacceptable tolerance (less than 36 elements fulfilled)



Definition of IS Project Success /Failure

The Success/failure of an IS project may be estimated as the rate of IS components to the Total IS components elements .

$$\left[\left|SUCCESS/\ FAILURE\right| = \frac{\left(Technology + Organisation + Human\_Resources\right)}{Total\_origional\_metric_0} * 100\%\right]$$

- ✓ $Technology_1$: Total number of metrics elements available in the Technology component of IS during IS project success evaluation
- ✓ $Technology_0$: Total number of metrics elements available in the Technology component of IS in the original metrics.
- ✓ $Organisational_1$: Total number of metrics elements available in the Organisational component of IS during IS project evaluation
- ✓ $Organisational_0$: Total number of metrics elements available in the Organisational component of IS in the original metrics
- ✓ $Human\_Resource_1$: Total number of metrics elements available in the Human resource component of IS during IS project evaluation
- ✓ $Human\_Resource_0$: Total number of metrics elements available in the Human resource component of IS in the original metrics
- ✓ $Total\_origional\_metric_0$: Total number of the metric elements in the original metrics
- ✓ $Total\_origional\_metric_0 = Technology_0 + Organisational_0 + Human\_Resource_0$

Assumptions:

- For a successful IS/IT Project all the IS components and their metric measurement must be available
- If some metric elements or some components are missing then the concept of acceptable failure is used.
- Acceptable failure is when some components are missing and their absence considered insignificant by the project manager.

Metrics components weighting

To evaluate and measure the IS/IT project success, some weights were assigned to the developed IS project success metrics. According to Chittoor (2012) metrics should be measured both during and after the project execution. The metrics weighting of IS components was the number of the metrics measures in each components. Thus in the Technology category of IS metrics, there are 12 metric measures, hence it was given a weight of the value 12. For Organisational component of IS metric there are 23 metric measures and as such it was given the value 23. Finally the Human resource component of IS was given the weight of 11 for 11 metric measures it constitutes. The metric comprised of 46 measures was given the total weight of 46.

Critical scores evaluation

Each metric measure is a critical score. If during project evaluation some metric measures are available there would be assigned a value 1, otherwise 0 to symbolise unavailability of the metric measure. Assuming the Technology component of IS has 11 metric measures, then critical score is 11.

The following assumptions defined the success measure of IS into two major categories;

- Success = 100 % critical score (all metric measures available) – this is the Ideal case category of success
- Failure = less than 100 %, but greater than 0 % critical score (Partial metric measures available)

Acceptable Failure definition and categories

The typical IT project may be subject to review by a host of stakeholder groups, including the project sponsor, system users, project team, maintenance and support personnel, internal and external auditors, and top management. At any point in time, a project may receive an entirely different opinion on success definition and the rate of failure acceptability.

Acceptable failure is when the user is aware and understands that the IS/IT project success is in a failure category but they are still satisfied with the level of success to carry on with the project.

Acceptable failure = Success – n

When n equals partial metrics measures available/ not 100% metric elements

Acceptable failure categories

Acceptable failure is categorised in to two broad categories of success which are;

- Acceptable tolerance = less than 100% metric measures, but greater than 50% of the metrics measures.
- Unacceptable tolerance = greater than 0% metric measures, but less than 50% of the metric measures.

Assumption:

- Acceptable failure cannot be equal to 0% otherwise you have not implemented IS system in your organisation.

Acceptable failure categories are shown in Figure 3

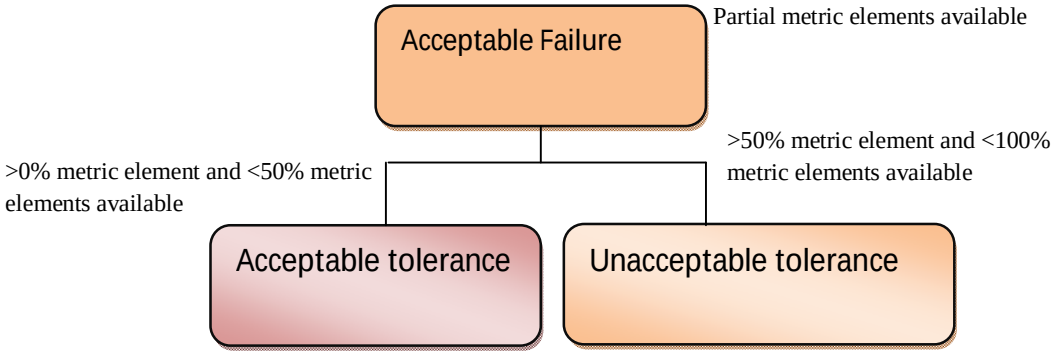


Fig. 3 Acceptable Failure main categories

Relationship between Success and Acceptable failure

The relationship between success measurement and acceptable failure is illustrated in Figure 4 following.

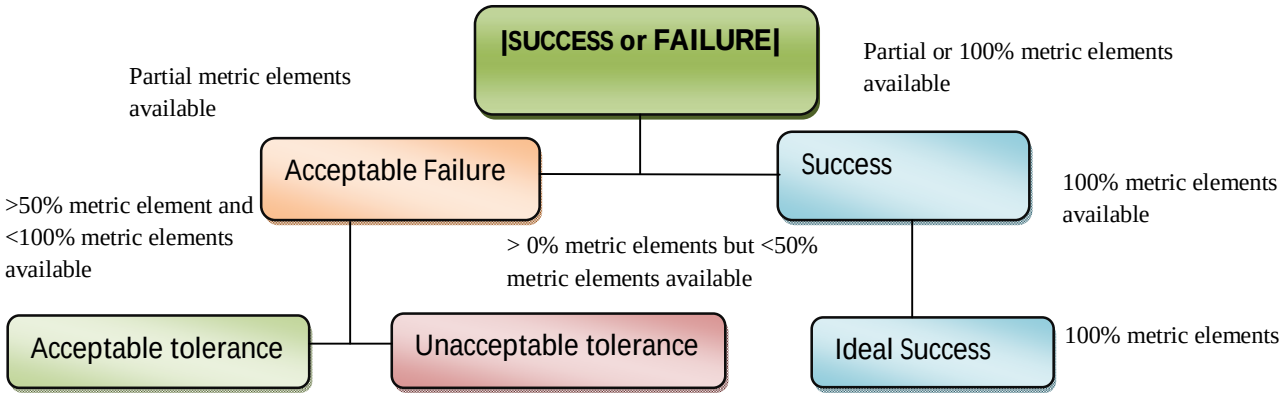


Fig. 4. Success vs. Acceptable Failure relationship

NB: Failure = Acceptable Failure, Success = Ideal success

Conclusion

From the forgoing discussion, we suggest that our proposed Project Management Ontology follows appropriate software measurement theoretic approach as validated in our empirical study with the various identification and categorization of success into the ideal, acceptable and unacceptable measures.

Bibliography

Baker, A. L, Bieman, J. M., Fenton, N; Gustafson, D. A; Melton, A; and Whitty, R. (1990). A Philosophy for Software Measurement. The Journal of Systems and Software 12:127-282

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

Chittoor, R (2012) Metrics for Project Success. Retrieved 27 August, 2015 from http://project-management.com/metrics-for-project-success

Fenton, N., and Pfeedger, S. L (1997). SoftwareMetrics: A Rigorous and Practical Approach, 2nd ed, Boston, MA:PSW Publishing

Gruber, T. (2009). Ontology. In L. L. Özsu (Ed.), Encyclopedia of Database Systems. Springer-Verlag.

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

Kang Ye etal. (2009). Ontologies for crisis contagion management in financial institutions. Journal of Information Science, 35 (5), 548–562.

Mphale, F (2016). Assessment of ICT Project Success/Failure in Botswana Using Project

Metric Models. MSC Dissertation, Department of Computer Science, University of Botswana. Unpublished

Okike, E. U (2007). Measuring Class Cohesion in Object-Oriented Systems Using Chidamber and

Kemerer Metric Suite and Java as Case study. PhD Thesis Department of Computer Science, University of Ibadan. Unpublished

Okike, E. U, Motshegwa, T, and Kgobathe, M. N (2016). Ontological Perspectives in Information Systems, Information Security and Computer Attack Incidents (CERTS/CIRTS). Proceedings of the 1st International Conference on the Internet, Cyber Security, and Information Sytems (ICICIS), PP. 46-60

Pereira, T and Santos, H. (2009). An Ontology Based Approach to Information Security. F. Sartori, M. A. Siccilia, and N. Manouseli (eds): MTSR 2009, CCIS 46, pp. 183-192

Ralph, N. D (2005). A validation by Measurement Theory of Proposed Object Oriented Software metrics. NASA Technical Report Server (NTRS). Reteieved 3/01/2017 from http://ntrs.nasa,gov/search.jsp?R=199611441

Silvonen, P. (2002, October 21). Ontologies and Knowledge Base. Retrieved October 22, 2015, from http://www.ling.helsinki.fi: http://www.ling.helsinki.fi/~stviitan/documents/ Ontologies_and_KB/ontology.html

Yao, H., Orme, A. M, and Etzkorn, L. (2005) Cohesion Metric for Ontology Design and Application. J. Computer Sc., 1(1):101-113