WEAK MEASUREMENT THEORY AND MODIFIED COGNITIVE COMPLEXITY MEASURE

Sanjay Misra, Hürevren Kılıç
Department of Computer Engineering, Atılım University,
Kızılcaşar Köyü, İncek, Gölbaşı, 06836, Ankara, Turkey
{smisra, hurevren}@atilim.edu.tr

Keywords: Validation criteria, weak measurement theory, software complexity measure, scale of a measure

Abstract: Measurement is one of the problems in the area of software engineering. Since traditional measurement theory has a major problem in defining empirical observations on software entities in terms of their measured quantities, Morasca has tried to solve this problem by proposing Weak Measurement theory. In this paper, we tried to evaluate the applicability of weak measurement theory by applying it on a newly proposed Modified Cognitive Complexity Measure (MCCM). We also investigated the applicability of Weak Extensive Structure for deciding on the type of scale for MCCM. It is observed that the MCCM is on weak ratio scale.

1 INTRODUCTION

The key element of any engineering process is measurement. Engineers use measures to better understand and assess the quality of engineered products or systems that they built. However, absolute measures are uncommon in software engineering. Instead, software engineers attempt to derive a set of indirect measures that provide an indication of quality of some representation of software. Software engineers plan 'how' an information system should be developed in order to achieve its quality objectives. The quality objectives may be listed as performance, reliability, availability and maintainability and are closely related to software complexity. Numbers of researchers have proposed variety of software complexity measures (Halstead, 1997), (Kushwaha and Misra, 2006), (Woodward and Hennel, 1979) and (Wang, 2003). Out of the numerous proposed measures, selecting a particular complexity measure is again a problem, as every measure has its own advantages and disadvantages. There is an ongoing effort to find such a comprehensive complexity measure, which addresses most of the parameters of software.

Elements of measurement theory has been proposed and extensively discussed in literature (Briand et.al., 1996), (Basili, 2007), (Fenton, 1994), (___, 1998), (Weyuker, 1998), (Zuse, 1991) and (Zuse, 1992) as a means to evaluate the software complexity measures. However, for formal approach of measurement theory , there is a problem: How can we recognize and describe the attribute in the empirical observation domain in order to relate its values to the proposed metric? (Misra and Kilic, 2006). Naturally, the representational measurement theory does not care about the practical difficulty of making empirical observations on the attribute and their identification. These are the underlying reasons for proposal of weak measurement theory by (Morasca, 2003). He has argued that the representation condition is very demanding for state of art of software engineering measurement. Therefore, he proposed for weakening the representation condition and developed the concept of weak measurement theory. We will discuss on it in detail in section 3.

In this paper, an effort has been made to validate MCCM (Misra, 2006) against the weak measurement theory. This theory for metric evaluation is more practical and useful and

encompasses all the factors which are important for the evaluation of any proposed measure. We know that a newly proposed complexity measure is acceptable, only when its usefulness has been proved by a validation process. It must be validated and evaluated both formally and practically. The purpose of the validation is to prove the usefulness of software attribute, which is measured by proposed metric. Actually, validation in narrow sense is the process through which one can test whether the measure's design purpose is achieved and the intended dimension of software is represented by the measure or not. According to (Fenton, 1991), by narrow sense validation the theoretical soundness of the measure is verified. Using a set of narrow sense validated measures one can show that the authentication of whole prediction system which is called validation in wide sense. From this perspective, the effort in this paper is to validate MCCM in narrow sense while our aim is to verify its theoretical soundness. A detailed discussion about the importance of validating software measures can be found in (Neal, 1997).

A brief introduction of MCCM is given in section 2. We validated MCCM from the perspective of weak measurement theory in section 3. After that in section 4, we examined the scale of the MCCM through weak extensive structure concept. The conclusion drawn is in section 5.

2 MODIFIED COGNITIVE COMPLEXITY MEASURE

The complexity measures based on cognitive informatics are in developing phase. Wang's cognitive functional size measure (Wang, 2003) depends upon internal architecture of the software, input and output. In MCCM (Misra, 2006), occurrences of operators and operands are taken into account in place of inputs and outputs. (Wang and Shao, 2003) claim that basic control structures are used for building logical software architecture, but operators and operands are equally important and part of design information. Once operators and operands have been considered, the number of input and output are automatically included. Further, the occurrence of operators and operands directly affect the architecture and as well as cognitive complexity of software, which was not taken into consideration in the cognitive functional size approach. Based on this, the cognitive complexity should depend on total occurrences of operators, operands and cognitive weights of basic control structures. Accordingly, MCCM is defined as:

$$MCCM = S_{oo} * W_c \quad (1)$$

where, S_{oo} is the total occurrences of operators and operands and given by,

$$S_{OO} = N_{i1} + N_{i2} \quad (2)$$

where, N_{i1}: The total occurrences of operators.
N_{i2}: The total occurrences of operands.
S_{OO}: Total occurrences of operators and operands.

W_c is the cognitive weights of basic control structures. Basic Control Structures (BCS), sequence, branch and iteration (Wang and Shao, 2002), (Wang and Shao, 2003), (Wang, 2004) are the basic logic building blocks of any software. The cognitive weight of BCS is the extent of difficulty or relative time and effort for comprehending given software modelled by a number of BCS's. There are two different architectures for calculating W_{bcs}: either all the BCS's are in a linear layout or some BCS's are embedded in others. In the former case, sum of the weights of all n BCS's; are added and in the latter, cognitive weights of inner BCS's are multiplied with the weights of external BCS's. The total cognitive weight of a software component W_c is defined as the sum of cognitive weight of its q linear blocks composed in individuals BCS's. Since each block may consists of m layers of nesting BCS's, and each layer with n linear BCS's, the total cognitive weight, W_c can be calculated by:

$$W_c = \sum_{j=1}^{q} \left[\prod_{k=1}^{m} \sum_{i=1}^{n} W_c(j,k,i) \right] \quad (3)$$

In fact, cognitive weights correspond to the number of executed instructions. For example, if in a simple program without any loop, the weights assigned to such code is one. Cognitive weights of basic control structures are basic building blocks of software and the standard weights for different control structures are given in (Wang and Shao, 2003).

In Equation-1, the S_{oo} values are multiplied by W_c values because of the possible higher structure value. For a simple program having only basic control structure the "sequence," W_c will not have any additional contribution to complexity. Therefore, for those programs the complexities are only due to S_{oo}. The above measure has been illustrated with the help of an example as described below:

Example 1. An algorithm to calculate the factorial of a number, to illustrate the application of MCCM

```
#include< stdio.h >
#include< stdlib.h >
#include< conio.h >
 int main ()
 {
   long int fact=1;
   int n;
   clrscr();
   printf("\ input the number");
   scanf ("%d", &n);
   if (n==0)
   else
     for (int i=n;i>1;i--)fact=fact*i;
   printf("\nfactorial(n)=%1d",fact);
   getch();
 }
```

We illustrate the MCCM to calculate the complexity of this program as under:

Total number of operands =15.

Total number of operators = 24.

S_{oo}= 24+15= 39.

BCS (sequence) $W_1 = 1$.

BCS (branch) $W_2 = 2$.

BCS (iteration) $W_3 = 3$

$W_c = W_1+W_2+W_3=1+2+3 = 6$.

MCCM = S_{oo}* W_c = 39 * 6 = 234 CCU.

Thus, the cognitive complexity measure value of the algorithm is 234 CCU.

3 VALIDATING MCCM BY WEAK MEASUREMENT

Measurement is simply the process of converting qualities to quantities. Such conversion process requires a formal description of the systems worked on. The components of the qualified system are (1) Entities whose attributes are wanted to be quantified; (2) Empirical binary relations showing the intuitive knowledge about the attributes and (3) Binary operations describing the production of new entities from the existing ones. Entities can either be physical objects or abstract artifacts that can be characterized or defined by a set of basic characteristics known as attribute (Wang, 2003). In the following paragraphs we describe the basic definition of measurement theory and check the validity of MCCM against it. We have also shown the problem related with the empirical observations in empirical relation system.

Definition 1: (Empirical Relational System-ERS) (Zuse, 1991). For a given attribute, an Empirical Relational System is an ordered tuple

$ERS=<E, R_1,...,R_n, o_1,..., o_m>$ where

E : the set of entities,

$R_1, ..., R_n$ denote n empirical relations such that each R_i has an *arity* n_i, and $R_i \subseteq E^{n_i}$

$o_1, ..., o_m$ denote m empirical binary operations on the entities that produces new entities from the existing ones, so o_j: $E \times E \rightarrow E$ and the operations are represented with an infix notation, for example, $e_k = e_i \, o_j \, e_l$.

The components of the quantification system are the values representing the decided quantities; the binary relations showing the dependencies among them and the binary operations describing the production of new values from the existing ones. In MCCM, the entities are the program bodies. The only empirical relation is assumed to be *more_or_equal_complex* and the only empirical binary operation is the *concatenation* of program bodies. However, from practical point of view there is a major problem for the identification and possibly the existence of such empirical observations. We can explain it by a solid example. Assume that we are given a program body P and we obtain a new program body Q by simply duplicating P. Also, assume that we are given another program body R for which there is no direct clear relation between P and R. One may easily establish the relation *more_or_equal_complex* between P and Q however it may not easy to make such an empirical observation between P and R. This is due to that we may not reach a consensus on how to order P and R based on their complexity.

Definition 2: (Numerical Relational System-NRS). A Numerical Relational System is an ordered tuple

$NRS=<V, S_1,...,S_n, p_1,..., p_m>$ where

V : the set of values,

$S_1, ..., S_n$ denote n relations such that the arity of S_i is equal to the arity of R_i, and $S_i \subseteq V^{n_i}$

$p_1, ..., p_m$ denote m numerical binary operations on the values that produces new values from the existing ones, so p_j: $V \times V \rightarrow V$ and the operations are represented with an infix notation, for example, $v_k = v_i \, p_j \, v_l$.

For MCCM, V is the set of positive integers, the binary relation is assumed to be $\geq$ and the numerical binary operation is the addition (i.e. +) of two positive integers.

Definition 3: Measure m is a mapping of entities to the values i.e. m: $E \rightarrow V$.

The measure for MCCM is defined by Equation (1). Note that the measure by itself does not provide

any mapping between empirical and numerical knowledge.

Definition 4: A measure must satisfy the following two conditions known as *Representation Condition.*

$$\forall i \in 1..n \; \forall <e_1, \ldots, e_{n_i}> \in E^{n_i}$$
$$(<e_1, \ldots, e_{n_i}> \in R_i \Leftrightarrow <m(e_1), \ldots, m(e_{n_i})> \in S_i)$$
(Part 1)

$$\forall j \in 1..m \; \forall <e_1, e_2> \in E \times E$$
$$(m(e_1 \; o_j \; e_2) = m(e_1) \; p_j \; m(e_2))$$
(Part 2)

The first part of the Representation Condition says that for a given empirically observed relation between entities, there must exist a numerical relation between corresponding measured values and vice versa. In other words, any empirical observation should be measurable and any measurement result should be empirically observable. The second part says a measured value of an entity which is obtained by the application of an empirical binary operation on two entities should be equal to the value obtained by corresponding numerical binary operation executed over individually measured values of entities. In other words, complexity of the whole should be definable in terms of complexities of its parts and their higher order relations.

For MCCM, the representation condition requires that (1) if for any two program body e_1 and e_2 are in *more_or_equal_complex* relation (i.e.$<e_1, e_2> \in$ *more_or_equal_complex*) then the measured complexity value of entity e_1 should be greater than the measured complexity value of entity e_2 (i.e. $m(e_1) > m(e_2)$) and vice versa. When we reconsider the program bodies P and Q where Q is the double of P, we can say that since MCCM is based on the counting of operators, operands and cognitive weights of basic control structures, they also become double or vice versa. Consequently, for part (1) of the condition we can say that the empirically observed *more_or_equal_complex* relation between two program bodies leads to a numerical binary relation > among those entities or vice versa. However, part (1) is only satisfied if there is such clear empirically observable relations between program bodies for example P and Q. On the other hand, in case of P and R since we do not have any clear empirical relation between them, the requirement

$$\forall i \in 1..n \; \forall <e_1, \ldots, e_{n_i}> \in E^{n_i}$$
$$(<m(e_1), \ldots, m(e_{n_i})> \in S_i \Rightarrow <e_1, \ldots, e_{n_i}> \in R_i)$$

implied by part (1) may not be required anymore. The formal approach describing such relaxation is proposed by (Morasca, 2003). He has argued that the original definition of Representation Condition is very demanding for state of art of software engineering measurement. Therefore, he suggested weakening (only) the first part of the condition two way link $\Leftrightarrow$, to a one way link, $\Rightarrow$ as follows:

Definition 5: Weak Representation Condition is defined by[8].

$$\forall i \in 1..n \; \forall <e_1, \ldots, e_{n_i}> \in E^{n_i}$$
$$(<e_1, \ldots, e_{n_i}> \in R_i \Rightarrow <m(e_1), \ldots, m(e_{n_i})> \in S_i)$$
(Part 1)

$$\forall j \in 1..m \; \forall <e_1, e_2> \in E \times E$$
$$(m(e_1 \; o_j \; e_2) = m(e_1) \; p_j \; m(e_2))$$
(Part 2)

When we consider the above example again, although we can calculate the MCCM values for P and R, this does not imply the existence of corresponding empirical relations between P and R. On the other hand, for a given *more_or_equal_complex* relation between P and Q that can be empirically observable one can always find corresponding metric values satisfying the *Weak Representation Condition*.

For part two of the Representation Condition, we can say that the complexity value of a program body which is obtained by concatenation (i.e. the empirical binary operation) of e_1 and e_2 is equal to the sum (i.e. the numerical binary operation) of their calculated complexity values. Therefore, MCCM satisfies the second part of the Representation Condition. Finally, we can say that MCCM satisfies the Weak Representation condition.

Showing the MCCM satisfies the Weak Representation Condition, we can investigate the type of the scale for our proposal. In order to be able to decide on the scale type we need to define the *Weak Scale* and *Weak Meaningful Statement* concepts (Morasca, 2003).

Definition 6: A *weak scale* is a triple <*ERS, NRS, m*>, where *ERS* is an Empirical Relational System, *NRS* is a Numerical Relational System, and *m* is a measure that satisfies the *Weak Representation Condition*.

Definition 7: A statement is called *Weak Meaningful Statement* if its truth value does not change if a weak scale is replaced by another weak scale. Formally, if *S(m)* is based on measure *m* and *S(m')* is the same statement obtained by replacing *m* with *m'*, we have *S(m)* $\Leftrightarrow$ *S(m')*.

Based on the notion of weak meaningful statement we can talk about four different types of weak scales:

Weak nominal scale: The meaningful statements of this class of scales are of the form $m(e_1) = m(e_2)$ for at least one pair of entities e_1 and e_2. If for one scale, $m(e_1) = m(e_2)$ is satisfied for a pair of entities e_1 and e_2 then we must have $m'(e_1) = m'(e_2)$ for all other scales m'.

Weak ordinal scale: <*ERS*, *NRS*, *m*> is a weak ordinal scale if $m(e_1) > m(e_2)$ is a weak meaningful statement for at least one pair of entities e_1, e_2. It is not required that $m(e_1) > m(e_2)$ or $m(e_1) = m(e_2)$ be weak meaningful statements for all pairs of entities e_1, e_2.

Weak interval scale: <*ERS*, *NRS*, *m*> is a weak interval scale if $(m(e_1) - m(e_2)) / (m(e_3) - m(e_4)) = k$ is a weak meaningful statement for at least one four-tuple of entities e_1, e_2, e_3, e_4 i.e., k is a constant value of all scales. It is not required that this statement is meaningful for all four-tuples of entities.

Weak ratio scale: <*ERS*, *NRS*, *m*> is a weak ratio scale if $m(e_1) / m(e_2) = k$ is a weak meaningful statement for at least one pair of entities e_1, e_2 i.e., k is a constant value of for all scales defined by the corresponding meaningful statement. Reconsider the two program bodies P and Q above as entities e_1 and e_2 where we calculate k as 2. Then, the statement $m(Q) / m(P) = 2$ is also a Weak Meaningful Statement for LOC or Control Complexity metrics. Therefore, we can informally say that MCCM is defined on *weak ratio scale*.

A formal way of proving a given scale is a weak ratio scale or not, is done by investigating whether the scale's Empirical Relation System is a Weak Extensive Structure or not (Briand, et.al., 1996).

Definition 8: A *hierarchy* is a pair <E, R> where $R \subseteq E \times E$ is a binary relation on E such that it does not contain any cycle, i.e. any sequence of pairs $\{<e_1, e_2>, <e_2, e_3>, \ldots, <e_i, e_{i+1}>, \ldots, <e_n, e_{n+1}>\}$ of any length n with $\forall i \in 1..n$ $R(e_i, e_{i+1})$ such that $e_1 = e_{n+1}$.

4 WEAK EXTENSIVE STRUCTURE

Definition 9: Let E be a set, R be a binary relation on E, and o is a total function o: $E \times E \rightarrow E$. The relational system (E, R, o) is a Weak *Extensive Structure* if and only if the following axioms holds (Morasca, 2003).

A1:
$\forall e_1, e_2, e_3 \in E$ ($Eq(e_1 \text{ o } (e_2 \text{ o } e_3), (e_1 \text{ o } e_2) \text{ o } e_3)$ where Eq is an equivalence relation defined as $Eq(e_1, e_2) \Leftrightarrow \neg R(e_1, e_2) \wedge \neg R(e_2, e_1)$ (axiom of *weak associativity*).

A2:
< E, R > is a hierarchy (axiom of *hierarchy*).

A3:
$\forall e_1, e_2, e_3 \in E$ ($R(e_1, e_2) \Rightarrow \neg R(e_2 \text{ o } e_3, e_1 \text{ o } e_3)$) (axiom of *monotonicity*).

A4:
$\forall e_1, e_2, e_3, e_4 \in E$ ($R(e_1, e_2) \Rightarrow \exists n \in N \neg R(ne_2 \text{ o } e_4, ne_1 \text{ o } e_3)$ where *ne* is recursively defined for any $e \in E$ as $1e = e$ and $\forall n > 1$ $ne = (n-1)e \text{ o } e$ (*Archimedean Axiom*).

For our proposal MCCM, the empirical relation R has the meaning "more or equal complex than" and the binary operation o between two objects is the "concatenation" of two program bodies. Now, we will investigate the validity of the above axioms for our empirical relation system (*ERS*=<*E, more_or_equal_complex, concatenation*>) defined for MCCM:

A1:
When we consider the example program bodies P, Q and R, since we do not have any knowledge of relation between R and the other two, we cannot say that P concatenated with (Q concatenated with R) is *more_or_equal_complex* than (P concatenated with Q) concatenated with R. Therefore, the concatenation operator of MCCM satisfies the *weak associativity* property.

A2:
For any program bodies X being *more_or_equal_complex* Y and Y being *more_or_equal_complex* Z, the Z can never be *more_or_equal_complex than X.* Therefore, we can say that <*E, more_or_equal_complex* > is a *hierarchy*.

A3:
When we consider the example program a body P, Q and R, having Q is more_or_equal_complex than P we cannot say that the same relation between Q concatenated with R and P concatenated with R because we have no knowledge of empirical relation of R and the others. Then, *monotonicity* property is also satisfied.

A4:
If entity e_1 is *more_or_equal_complex* than e_2 then for any e_3 e_4, we cannot establish a new *more_or_equal_complex* relation by any number of concatenations; say n times, of e_1 and e_2 to themselves followed by concatenation of e_3 and e_4 with them, respectively. This is because we may not have any knowledge of relation between the results

of ne_2 concatenated with e_4 and ne_1 concatenated with e_3 due to unknown relation between each of e_3 and e_4 with other two. Consequently, Archimedean axiom is also satisfied.

As a result, the ERS description of the proposed MCCM is a Weak Extensive Structure. Based on the theorems "Existence of an Additive Scale for a Weak Extensive Structures" and "Weak Additive Scale and Weak Ratio Scales" given by in (Morasca, 2003) we can say that MCCM is defined on W*eak Ratio Scale*. Note that among the scales defined above, the ratio scale is the highest in level. Therefore, it may be more powerful than the other scales reflect.

5 CONCLUSIONS

MCCM is a new proposed complexity measure based on cognitive aspects software development. Any proposed complexity measure should be validated and evaluated against mathematical tool of measurement theory which is extensively used in the literature as a means to evaluate software engineering metrics. However it is known that in classical measurement theory there is problem in defining empirical observations on software entities in terms of their measured quantities. Consequently, the proposal of weak measurement theory is thought to be a useful alternative for validating and evaluating the MCCM. We showed that MCCM satisfies most of the parameters required by the weak measurement theory and it is also found that the proposed measure is on weak ratio scale.

In the light of the experiences we propose the future work to include the following:

1. Further researches on weak measurement theory are required. Weak measurement theory is only a partial solution to problem related to definition of a measure based on measurement theory.

2. To the best of our knowledge, complexity measures based on cognitive aspects are not tested by the practitioners. This is also a task for future work.

REFERENCES

Briand, L.C., Morasca, S., and Basili, V.R., 1996. Property based Software Engineering Measurement. IEEE Transactions on SE, vol. 22, 1, pp.68-86.

Basili, V., 2007. The Role of Controlled Experiments in Software Engineering Research," in Empirical Software Engineering Issues, LNCS 4336, V.Basili et al., (Eds.), pp.33-37.

Halstead M.H., 1997. Elements of Software Science. Elsevier North-Holland. New York.

Fenton, N., 1994. Software Measurement: A Necessary Scientific Basis," IEEE Trans. Software Engineering, Vol. SE-20, no. 3, pp.199-206.

Fenton, N. Software Metrics: A Rigorious Approach, Chapman & Hill, London, UK, 1991.

____, 1998, IEEE Computer Society Standard for Software Quality Metrics Methodology. Revision IEEE Standard 1061-1998.

Kushwaha D.S. and Misra A.K., 2006. Robustness Analysis of Cognitive Information Complexity Measure using Weyuker's Properties. ACM SIGSOFT Software Engineering Notes. 31: 1-6.

Morasca S., 2003. Foundations of a Weak Measurement-Theoretic Approach to Software Measurement. LNCS. 2621, pp.200-215.

Misra S. and Hurevren Kilic, 2006. Measurement theory and Validation Criteria for Software Complexity measure. ACM SIGSOFT Software Engineering Notes. 31, no.6. pp.1-3

Misra S., 2006. Modified Cognitive Complexity measure. Proc. of 21st ISCIS'06, Lecture Notes in Computer Science. 4263.pp. 1050-1059.

Neal R.D., 1997, Modeling the Object-Oriented Space Through Validated Measures, Aerospace Conference, 1997. Proceedings., IEEE Volume 4, 1-8 Feb. 1997 Page(s):315 - 327 vol.4

Weyuker E.J., 1988. Evaluating Software Complexity Measure. IEEE Transaction on Software Engineering. 14: pp. 1357-1365.

Woodward M.R. and Hennel M. A., 1979. David. A Measure of Control Flow Complexity in Program Text. IEEE Transaction on Software Engineering. 1. pp.45-50.

Wang Y.; Shao J., 2002. On Cognitive Informatics. Keynote Lecture. Proceeding of the 1st IEEE International Conference on Cognitive Informatics. pp. 34–42.

Wang Y. and Shao J. , 2003. A New Measure of Software Complexity Based on Cognitive Weight. Can. J. Elect. Comput. Engg. Pp.69-74.

Wang Y., 2004. On Cognitive Informatics: Foundation of Software Engineering. Proceeding of the 3rd IEEE International Conference on Cognitive Informatics (ICCI'04) .IEEE CS Press .pp.22-31.

Wang, Y., 2003. The Measurement Theory for Software Engineering. Proceedings of Canadian Conference on Electrical and Computer Engineering CCECE 2003, pp. 1321-1324.

Zuse, H., 1991. Software Complexity Measures and Methods. de Gruyter.

Zuse, H., 1992. Properties of Software Measures. Software Quality Journal, vol. 1, pp. 225- 260.