http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 9 (7): 1317-1326, 2010
ISSN 1812-5638

Complexity Metrics for Executable Business Processes

[1,2]G.M. Muketha, [1]A.A.A. Ghani, [1]M.H. Selamat and [1]R. Atan
[1]Department of Information Systems, Faculty of Computer Science and Information Technology,
University Putra Malaysia, 43400 Serdang, Selangor, Malaysia
[2]Department of Computer Science, Faculty of Science,
Masinde Muliro University of Science and Technology, P.O. Box 190-50100 Kakamega, Kenya

Abstract: In this study, seven metrics are proposed for measuring the complexity of Executable Business Processes (EBP). The metrics are either derived from existing business process metrics or adapted from software metrics. Evaluation was carried out in three case studies with the goal of finding out if the metrics are theoretically sound and at the same time intuitional. In case 1, metrics values were computed from three processes and then analyzed to check whether they agree with reality. In case 2, the metrics were grouped into two categories of length and complexity and then separately checked for their conformance to Briand's framework. In case 3, all the metrics were treated under one complexity category and then checked for their conformance to Weyuker's properties. Results indicate that the new metrics are intuitional and are good if used in their respective categories, or when used together to complement each other in order to give a fuller view of process complexity.

Key words: BPEL, cognitive complexity, information flow, structural complexity, complexity metrics

INTRODUCTION

Many people in industry and academia have in recent years shown a lot of interest in business processes created using the Business Process Execution Language (BPEL). Several researchers have attributed BPEL's increased demand to the fact that it has richer semantics than most other business process modeling languages (Charfi and Mezini, 2007) and that it is quickly becoming the de facto industry standard for Web services composition and execution (Modafferi and Conforti, 2006; Zheng *et al*., 2007).

In recent years, there has been a shift from the graphical process modeling to executable service-oriented process programming, which should affect how business processes are defined (Lindsay *et al*., 2003). According to Michelson (2005), a BPEL process is a long-running service, coordinating the actions of multiple parties through a series of work steps. Parties are partner Web services while steps are the activities needed to coordinate services. Web services depend on the ability of often heterogeneous parties to communicate with each other (Misra *et al*., 2006). Formally, a BPEL process P may be defined as a 2-tuple <E, R>, where, E is the set of basic activities in the process and R is the set of control-flow relations that connect one activity to another. This definition is based on Briand's work on modular software systems, where, a program may consist of several procedures, functions or classes (Briand *et al*., 1996). Unlike in the case of software, BPEL lacks sufficient modularity features (Charfi and Mezini, 2007). It is possible however, to partition a process alongside control-flow boundaries such that each control-flow block is treated as a module. Furthermore, some simple processes contain only one control-flow structure and can also be treated as modules.

The problem with BPEL processes, also called Executable Business Processes (EBP) in this study, is that they can get highly complex with age (Cardoso, 2006). Some known causes of complexity include the introduction of new activities to accommodate new partners and dynamic partner links which can only be determined at run time. High complexity in EBP makes them difficult to understand, analyze and modify (Cardoso, 2006) and consequently, difficult to maintain (Canfora *et al*., 2005). High complexity is therefore undesirable and should be measured for purposes of controlling it. Parthasarathy and Anbazhagan (2006) argued that if used properly, metrics allow us to quantify success or failure, improvement and make useful managerial decisions concerning software or processes. However, very few business process metrics have

Corresponding Author: Geoffrey M. Muketha, Department of Information Systems,
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia
Tel: +6 017 369 8350 Fax: +6 03 8946 6577

appeared in literature to date probably because this is still a fairly new research area. Some frequently cited business process metrics may be found in the works of Cardoso *et al.* (2006), Cardoso (2006) and Gruhn and Laue (2006). A few of these metrics were designed for graphically modelled processes and are therefore not directly applicable to EBP. Clearly, new metrics are needed not only to measure business processes, but more specifically, to measure EBP complexity.

In this study, seven metrics are proposed with the goal of measuring the complexity of EBP. In order to achieve this goal and also in line with recent business process research trends, the methodology used in this study is to derive new metrics from either existing business process metrics, or from related software metrics through adaptation. Other researchers who have used adaptation method include Cardoso *et al.* (2006), Cardoso (2006) and Gruhn and Laue (2006). Adaptation of software metrics is made possible by the fact that there are many similarities between software programs and business processes (Vanderfeesten *et al.*, 2007; Cardoso *et al.*, 2006). This method is advantageous in that technology can be reused without reinventing the wheel. For instance in BPEL, basic activities correspond to simple program statements such as procedure calls, assignment statements etc. while structured activities correspond to program control-flow statements such as sequence, branch, iteration etc. Some of the metrics presented here fall under structural and cognitive complexity categories while others fall under length and information flow categories. A description of categories of complexity metrics may be found in the work of Koh *et al.* (2008). Three BPEL processes are given to illustrate how the new metrics can be computed. The metrics were also validated theoretically using two complimentary case studies where, they were checked for conformance to Briand's framework (Briand *et al.*, 1996) and to Weyuker's properties (Weyuker, 1988).

In Cardoso *et al.* (2006), are proposed several length metrics such as the Number of Activities (NOA) and the Number of Activities and Control-flows (NOAC) to measure process length. These metrics are business process adaptations of the Lines of Code (LOC) metric in software engineering. These metrics are however, limited in that they fail to show the length of individual control-flow blocks. Hence, they lack sufficient information that process designers need to know, for example, when to redesign a large control-flow block.

Other metrics that influenced this work include the Shepperd metric (Ince and Shepperd, 1989) and its predecessor, the information flow metric (Henry and Kafura, 1981). The Shepperd metric is computed as a square of the product of fan-in and fan-out information flows. These authors define fan-in as the information flows terminating at a module and fan-out as the information flows leaving a module. The formula for the original information flow metric of a module (M) = length (M)* ((fan-in (M)* fan-out (M))^2. A metric called interface complexity (IC) has been proposed as the business process adaptation of the original information flow metric (Cardoso *et al.*, 2006). Shepperd refined the original metrics so that the complexity of a module (M) = ((fan-in (M)* fan-out (M))^2. The refined metric prevented blurring with control-flow and concentrated purely on information flow (Ince and Shepperd, 1989). The Shepperd metric's potential usefulness to business process measurement has not been investigated yet.

Another important metric is the Cognitive Functional Size (CFS) for software (Shao and Wang, 2003). Earlier studies by Wang that influenced CFS include Wang (2002, 2003). CFS is computed by multiplying cognitive weights assigned to control-flows with total input and output information flows. Cognitive weights of 1, 2, 3 and 4 are assigned to sequence, branch, loop and parallel structures respectively (Shao and Wang, 2003). According to Shao and Wang (2003), cognitive complexity is the ease of understandability of a model. Understanding how the human memory works can help us to determine the mental effort that a designer needs in order to understand a particular unit of information. Gruhn and Laue (2006) have already made an effort to adapt CFS for business processes. However, Gruhn and Laue (2006) work focuses on graphically modeled processes and therefore does not utilize input and output information flows found in the original CFS metric. As a consequence, this metric cannot be effectively used for measuring the cognitive complexity of EBP.

Törn *et al.* (1999) have proposed the Structural Complexity (SC) metric for software, which is computed by multiplying the length of a program block by the complexity weight of its control-flow structures. Complexity weights are used to obtain structural complexity. Törn *et al.* (1999) assigned complexity weights of 1.1, 1.3 and 1.5 to sequence, branch and loop structures respectively. Further empirical validation studies on this metric have been undertaken by Costea (2007). The difference between the SC and the CFS metrics is that SC is a function of control-flow weights and length of a software module while CFS is a function of control-flow weights and total input and output information flows. There is also some difference in the weights assigned to control-flow blocks, but generally, both SC and CFS recognize iteration as being more complex than branching, which is more complex than sequence. Although, the SC

metric for software looks promising, it requires some adaptation before it can be used for EBP measurement.

IDENTIFICATION OF MEASUREMENT ATTRIBUTES

This section describes a list of attributes to be measured from a BPEL process. The attributes are divided into two levels: process level and structured activity level. The attributes to be measured include:

- Process-level attributes
  - Length of process (number of basic activities)
  - Length of process (number of structured activities)
  - Information flow complexity of process
  - Cognitive complexity of process
  - Structural complexity of process
- Structured activity-level attributes
  - Average length of structured activity
  - Average cognitive complexity of structured activity

METRICS DEFINITION

Process-level metrics

Number of Basic Activities in a process (NOBA): NOBA metric is a length metric that counts the number of basic activities in a process. Length is a simple one-dimensional metric, unlike other complexity metrics which manipulate two or more dimensions of a process.

Number of structured activities in a process (NOSA): NOSA metric is another simple one-dimensional length metric similar to NOBA. However, instead of counting basic activities, it counts the number of structured activities in a process. It should be noted that NOSA simply counts the number of structured activities but does not attach any weights to them.

Information Flow complexity for Business Process (IF4BP): IF4BP metric is a business process adaptation of Shepperd metric (Ince and Shepperd, 1989). In EBP, fan-in is represented by input activities while fan-out is represented by output activities. The IF4BP of a BPEL module is defined as the square of the product of the Number of Input Activities (NOIA) and the Number of Output Activities (NOOA) contained in it. This is shown in Eq. 1:

$$IF4BP_m = (NOIA * NOOA)^2 \quad (1)$$

where, m is a BPEL module.

For a large process with several BPEL modules, a summation of the complexities of all modules contained in the process is obtained as shown in Eq. 2:

$$IF4BP = \sum_{m=1}^{n} IF4BP_m \quad (2)$$

where, n is number of modules in the process and m is a BPEL module.

Cognitive Complexity for Business Process (CCBP): CCBP metric is a business process adaptation of the cognitive functional size measure (Shao and Wang, 2003). It is a function of the total NOIA and NOOA and the total cognitive weight (W_c) of the structured activities in a process. Table 1 shows the cognitive weights assigned to each category of control-flow structures. Assignment of these weights represents the psychological effort needed by a designer to comprehend a control-flow block of the process.

To calculate CCBP, a summation of cognitive weights (W_c) for all structured activities in the process is obtained and then multiplied by the total input and output activities as shown in Eq. 3:

$$CCBP = (NOIA + NOOA) * W_c \quad (3)$$

Structural Complexity of Business Process (SCBP): SCBP is a business process adaptation of the SC metric for software (Törn *et al*., 1999). SCBP is a function of two attributes called average structural complexity and process length. The SCBP of a process P is the product of its length and its average structural complexity. As shown in Eq. 4, l(P) is the total length of the process and asc(P) is the total average structural complexity of all control-flow structures in the process:

$$SCBP=l(P)*asc(P) \quad (4)$$

Understanding l(P): Measuring the length of a software program is different from measuring the length of a process. The length of a software program can be

Table 1: Assigning weights to categories of control-flow structures

Category	Activity	W_c
Sequence	Sequence	1
Branch	if, pick,	2
Loop	while, for each, repeat until	3
Parallel	Flow	4

Table 2: Formulas for categories of control-flow structures

Category	Formula	Description
Sequence	asc (P_1, P_2,... P_n) = asc_sasc (P_1, P_2,... P_n)	do p_1, p_2...p_n
Branch	asc (if) = asc_iasc (b, p, q), P_2,....P_n	if b then p else q
Loop	asc (while) = asc_wasc (b, p)	while b do p
Parallel	asc (flow) = asc_fasc (b, p, q)	if b then p and q

Table 3: Assigning weights to categories of control-flow structures

Category	Activity	asc
Sequence	Sequence	1.1
Branch	if, pick	1.3
Loop	while, forEach, repeatUntil	1.5
Parallel	Flow	1.7

conveniently measured by counting its lines of code. However, since process statements are called activities, the length of a process can be measured by counting the number of activities contained in it. The simplest process has at least one structured activity a few basic activities. A length of 1 is assigned to each basic activity. This means that a process with one structured activity that encloses two basic activities will have a length of 2 i.e., l(P) = 2, where, P is a process. Length is additive, hence, a process with three basic activities will have a length l(P) of 3.

Understanding asc(P): Structural complexity is additive (Törn *et al*., 1999), therefore, for a process with many units P = {p_1, p_2, …, p_n}, the average structural complexity asc(P) is calculated as the average of the individual units of complexity as shown in Eq. 5:

$$asc(P) = asc(P_1, P_2, ..., P_n) = \frac{\sum_{i=1}^{n} l(P_i) * asc(P_i)}{\sum_{i=1}^{n} l(P_i)} \quad (5)$$

Törn *et al.* (1999) also define the formulas for calculating complexities for sequence, branch and loop control structures. These formulas were extended in order to cater for BPEL's parallel structure as shown in Table 2.

Törn *et al.* (1999) proposed complexity weights of 1.1, 1.3 and 1.5 for sequence, branch and iteration structures respectively, based on the assumption that each control structure has an inherent complexity. Additionally, Törn *et al.* (1999) follows the intuition that a sequence is less complex than a branch which is less complex than a loop. Unlike in the case of CCBP metric where, a weight of 1 is assigned to a sequence, Törn *et al.* (1999) proposes that a weight of 1.1 be assigned to a sequence instead. The justification for this type of weight is that a sequence contains ordering information for its elements (i.e., a sequence is a set of ordered elements) and is therefore more complex than a collection (which is a set of unordered elements). This is in recognition of the fact that ordering does in fact introduce extra cost to the existing set. Törn's initial values for sequence, branch and iteration were extended with a new weight of 1.7 to cater for BPEL's parallel structure. The reason for assigning a heavier weight for parallel processing is that in addition to executing activities in parallel, results must also be synchronized. The extended weights are shown in Table 3.

Applying the formulas to compute the SCBP of structured activities: When applying the formulas to compute BPEL structures, atomic nodes are first represented in node notation as described in Table 3. Next, the node notation of each atom is substituted with values for its l and asc. A BPEL process has two types of atomic nodes:

- Decision nodes-includes branch, loop and parallel structures
- Simple statement nodes-includes basic activities appearing in a sequence structure

Decision nodes are represented as (b l_a asc_a), where, l_a and asc_a are the length and complexity of the activities. Similarly, simple statement nodes are represented as (n l_a asc_a), where, l_a and asc_a are the length and complexity of the activities in sequence.

Examples: Computing SCBP of structured activities in the process.

Here, several examples on how to calculate the SCBP of BPEL processes are given.

SCBP for sequence processing: The SCBP of a sequence S with two basic activities is calculated by first representing it using the node notation sequence formula and then substituting the formula as shown below:

S (n l_a asc_a) (n l_a asc_a), where, a is an atomic activity
= (S(n 1 1)(n 1 1)), substituting formula
= (n 2 1.1), add lengths of the two atoms, substitute complexity weight of atom with that of sequence
= 2*1.1 = 2.2, apply the formula for structural complexity
Therefore, l = 2, asc = 1.1 and SCBP = 2.2.

SCBP for branch processing: The SCBP of a branch with one decision node and two basic activities invoke and the following structure if condition then invoke B, is calculated as follows. First represent it using the decisions formula and then substitute the formula as shown below:

if(b l_a asc_a) (n l_a asc_a) (n l_a asc_a), where, a is an atomic activity
= (if(b 1 1)(n 1 1) (n 1 1))
= (n 3 1.3*(1+1+1)/3)
= (n 3 1.3)
= 3*1.3 = 3.9
Therefore, l = 3, asc = 1.3 and SCBP = 3.9.

SCBP for loop processing: The SCBP of a loop with one decision node and one basic activity and the following structure while condition then invoke B, is calculated as follows. First represent it using the decisions formula and then substitute the formula as shown below:

do(b l_a asc_a) (n l_a asc_a), where, a is an atomic activity
= (do(b 1 1)(n 1 1))
= (n 2 1.5*(1+1)/2)
= (n 2 1.5)
= 2*1.5 = 3
Therefore, l = 2, asc = 1.5 and SCBP = 3.

SCBP for parallel processing: The SCBP of a parallel structure with one decision node two basic activities to be executed in parallel and the following structure if condition then invoke A and B, is calculated as follows: First represent it using the decisions formula and then substitute the formula as shown below:

flow(b l_a asc_a) (n l_a asc_a) (n l_a asc_a), where, a is an atomic activity
= (flow(b 1 1)(n 1 1) (n 1 1))
= (n 3 1.7*(1+1+1)/3)
= (n 3 1.7)
= 3*1.7 = 5.1
Therefore, l = 3, asc = 1.7 and SCBP = 5.1.

Structured activity-level metrics

Average Length of Structured Activity (ALSA): The metric ALSA is a ratio between basic activities and structured activities in the process. It calculates the average length of structured activities in a process. Since, structured activities in BPEL are treated as equivalent to modules, then ALSA metric is equivalent to calculating average module size in software engineering. Information from ALSA would help designers to for example, decide when to redesign the process. ALSA is computed by dividing the number of basic activities with the number of structured activities in the process as shown in Eq. 6:

$$ALSA = NOBA/NOSA \quad (6)$$

Average Cognitive Complexity of Structured Activity (ACCSA): The metric ACCSA is derived from CCBP. It is the ratio between CCBP and the total number of structured activities in the process. As in the case of ALSA, ACCSA provides useful information to designers that can help to determine when to redesign the process. ACCSA is computed by dividing the cognitive complexity with the number of structured activities in the process as shown in Eq. 7:

$$ACCSA = CCBP/NOSA \quad (7)$$

RESULTS

This section presents evaluation results obtained from three case studies. Case 1 involved computing of the metrics values for three BPEL processes and its aim was to find out whether the metrics values were intuitional. Cases 2 and 3 were complementary theoretical validation studies whose goal was to establish the theoretical soundness of the metrics. In case 2, the metrics were checked for conformance to Briand's framework (Briand *et al*., 1996) and in case 3, they were checked for conformance to Weyuker's properties (Weyuker, 1988).

Case 1: Computing metrics values for three BPEL processes: Here, the new metrics are used to calculate the lengths and complexities of three BPEL processes. In Fig. 1, the Loan Eligibility Process receives a loan request from a customer. The process then checks if the customer is eligible for the loan. The results are used to accept or reject the application. The appropriate message is finally sent back to the customer.

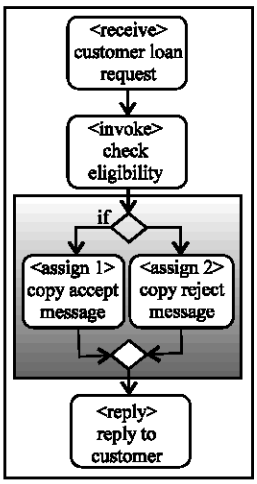

Fig. 1: Loan eligibility process

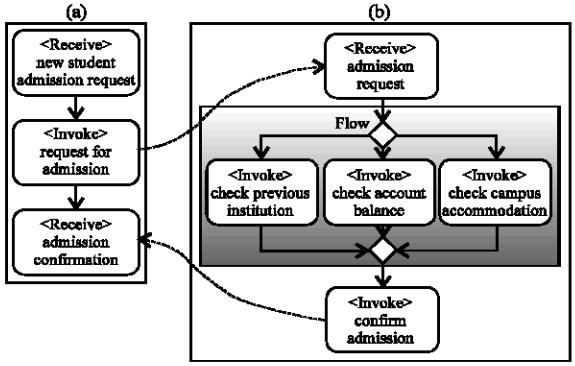

Fig. 2: Two interacting processes to facilitate new student admission. (a) New Student Process and (b) Admission Process

Figure 2 shows two interacting but otherwise different processes. Figure 2a is a new student process that can be used as an entry point by a prospective student to apply for university admission. This simple process receives a new student request for admission and then invokes the university admissions process in Fig. 2b. The new student process waits for confirmation of admission from admission process, after which it terminates. Thus, the new student process is treated as a client of admission process. After receiving a request from the new student process, the admission process invokes three Webs services in parallel. The first Web service provides information on the student's previous institution for confirmation of student's academic record, the second Web service provides information on the student's bank account to confirm that the student has the financial capability to undertake the course and the third Web service provides information on whether there is on-campus accommodation for the new student. All these three issues must be satisfied before confirmation of admission can be sent back to the prospective student.

Metrics values for the three processes are presented in Table 4. The calculations are done with the understanding that LoanEligibilityProcess has two structured activities (an if and a sequence), NewStudentProcess has one structured activity (a sequence) and AdmissionProcess has two structured activities (a flow and a sequence).

Case 2: Validating metrics with Briand's generic measurement framework: Briand's framework proposed five metrics categorizes, namely, size, length, complexity and coupling and cohesion (Briand *et al*., 1996). The new metrics were grouped into two categories: length and complexity. NOBA, NOSA and ALSA metrics were validated under length category while IF4BP, CCBP, ACCSA and SCBP metrics were validated under complexity category. Before validation commenced, an effort was made to map the length and complexity categories of Briand's framework into business process perspective. Other categories were left out because they are not related to the metrics proposed in this study. Mapping was necessary because the framework was initially intended to be used in the software engineering field.

Table 4: Values of metrics for the three processes

Metric	Loan eligibility process	New student process	Admission process
NOBA	5.0	3.0	5.0
NOSA	2.0	1.0	2.0
IF4BP	4.0	4.0	16.0
SCBP	7.2	3.3	9.0
CCBP	9.0	3.0	25.0
ALSA	2.5	3.0	2.5
ACCSA	4.5	3.0	12.5

Length of the process: The length of a process P is a function Length(P) that is characterized by the following five properties.

Length 1: Nonnnegativity. The length of a process P = <E, R> cannot be negative, but can be null if a system has got no elements, i.e., Length (P)≥0.

Length 2: Null value. The length of a process P = <E, R> is null if P is empty i.e., if P has got no activity nodes in it.

Length 3: Nonincreasing monotonicity for connected components. Let P be a process and m be a module of P such that m is represented by a connected component of the graph representing P. Adding relationships between elements of m does not increase the length of P, i.e., (P = <E, R> and m = <E_m, R_m> and m⊆P) and m'' is a connected component of P'' and P''=<E, R'> and R' = R∪{<e_1, e_2>}} and <e_1, e_2>∉R and e_1∈E_{m1} and e_2∈E_{m1}⇒Length (P).≥Length (P').

Length 4: Nondecreasing monotonicity for non connected components. Let P be a process and m_1 and m_2 be two modules of P such that m_1 and m_2 are represented by two separate connected components of the graph representing P. Adding relationships from elements of m_1 to elements of m_2 does not decrease the length of P i.e., (P = <E, R> and m_1 = <E_{m1}, R_{m1} and m_2 = <Em_2, Rm_2> and m_1⊆P and m_1⊆P'' are separate connected components of P'' and (P' = <E, R'> and R' = R∪{<e_1, e_2>} and <e_1, e_2>∉R and e_1∈E_{m1} and e_2∈E_{m1}⇒Length (P).≥Lengrh (P').

Table 5: Summary of results of length metric validated with Briand's framework

Property	NOBA	NOSA	ALSA
Length 1	✓	✓	✓
Length 2	✓	✓	✓
Length 3	✓	✓	✓
Length 4	✓	✓	✓
Length 5	✓	✓	✓

✓: Satisfied property

Table 6: Summary of results of complexity metrics validated with Briand's framework

Property	IF4BP	CCBP	SCBP	ACCSA
Complexity 1	✓	✓	✓	✓
Complexity 2	✓	✓	✓	✓
Complexity 3	✓	✓	✓	✓
Complexity 4	✓	✓	✓	✓
Complexity 5	✓	✓	✓	✓

✓: Satisfied property

Length 5: Disjoint modules. The length of a process P = <E, R> composed of two disjoint modules m_1, m_2 is equal to the maximum of the lengths of m_1 and m_2 i.e., P = $m_1 \cup m_2$ and $m_1 \cap m_2 = \phi$ and E = $E_{m1} \cup E_{m2} \Rightarrow$ Length (P) = max {Length (m_1), Length (m_2)}.

The metrics NOBA, NOSA and ALSA satisfied the five length requirements as specified in Briand's framework. This means their values cannot be negative, can be null if empty and are neither increased nor decreased whenever relationships between two connected components are added. Also, the length of two disjoint modules in a process is equal to the maximum of the lengths of the two modules. A summary of these results is shown in Table 5.

Complexity of the process: The complexity of a process P is a function complexity (P) that is characterized by the following five properties.

Complexity 1: Nonnnegativity. The complexity of a process P = <E, R> cannot be negative, but can be null if a system has got no elements i.e., Complexity (P)≥0.

Complexity 2: Null value. The complexity of a process P = <E, R> is null if the R is empty i.e., if P has got no structured activities in it, then its complexity is null i.e., R = $\phi \Rightarrow$Complexity (P) = 0.

Complexity 3: Symmetry. The complexity of a process P = <E, R> does not depend on the convention chosen to represent the relationships between its elements i.e., (P = <E, R and P^{-1} = <E, R^{-1}>)⇒Complexity (P) = Complexity (P^{-1}).

Complexity 4: Component monotonicity. The complexity of a process P = <E, R> is no less than the sum of the complexities of any two of its modules with no relationships in common i.e., (P = <E, R> and m_1 = <E_{m1}, R_{m1}> and m_2 = <E_{m2}, R_{m2}> and $m_1 \cup m_2 \subseteq P$ and $R_{m1} \cap R_{m2} = \phi \Rightarrow$ Complexity (P)≥Complexity (m_1) + Complexity (m_2).

Complexity 5: Disjoint module additivity. The complexity of a process P = <E, R> composed of two disjoint modules m_1, m_2 is equal to the sum of the complexities of the two components i.e., (P = <E, R and P = $m_1 \cup m_2$ and $m_1 \cap m_2$ and $\Rightarrow$ complexity (P) = Complexity (m_1) + Complexity (m_2).

The complexity metrics IF4BP, CCBP, SCBP and ACCSA satisfied the five complexity properties as specified in Briand's framework. This means that they cannot be negative, can be null if empty and do not depend on the convention used to represent the relationships. In addition, they always return a complexity value of a process that is greater than or equal to the sum of any two of its components. Finally, a process composed of any two disjoint components has the same complexity to that of the sum of the complexities of its components. A summary of these results is shown in Table 6.

Case 3: Validating metrics with Weyuker's properties: Weyuker proposed nine properties for validating complexity metrics (Weyuker, 1988). There is a large number of metrics that have been validated with Weyuker's properties such as Cardoso (2005) and Misra (2007). In this study, Weyuker's properties were used to validate all the new metrics despite the fact that they were initially intended to validate only complexity metrics (Weyuker, 1988). The motivation here was to complement results in case 1 and also to find out how the metrics NOBA, NOSA and ALSA would fare when compared to multi-dimensional metrics such as CCBP and SCBP, given the fact that process length (count of number of activities) has been categorized by Cardoso as a form of complexity, or more precisely, activity complexity (Cardoso, 2006).

Property 1: ($\exists$P) ($\exists$Q(|P| ≠ |Q|: There exist processes P and Q such that |P| is not equal to |Q|. This property requires that a good metric should be able to discriminate between two different processes such that they do not return same measurement results.

The metrics NOBA, NOSA, ALSA, IF4BP, CCBP, SCBP and ACCSA always return a different complexity for any two non identical processes where, either the number of basic activities or type of decision node is varied. Therefore, they all satisfied Property 1.

Property 2: Let, c be a non-negative number. Then there are only finitely many processes of complexity c. This

property asserts that a changing process must also cause a change to its complexity. NOBA, NOSA, ALSA, CCBP, SCBP and ACCSA can detect changes in complexity when the number of basic activities is varied and type of decision node is kept constant. However, IF4BP can detect changes in complexity when number of input and output variables is varied, but cannot detect change in complexity when decision type is varied. Therefore, all the proposed metrics satisfied Weyuker's Property 2 except IF4BP.

Property 3: There exist distinct processes P and Q for which |P| = |Q|. This property asserts that there exist two different processes whose effect is identical i.e., two different processes with identical values. For example, two processes could differ only in the naming of variables but may otherwise define the variables with the same data type and value. According to Weyuker, a good metric should return same complexity for such processes.

Two processes P and Q will have the same complexity provided that they are identical in terms of structure, length and interface. Therefore, this property held true for all proposed metrics.

Property 4: ($\exists$P) ($\exists$Q (P≡Q and $|P| \neq |Q|$): There exist processes P and Q such that the external effect of P and Q are identical, but |P| is not equal to |Q|. This property asserts that two processes could look identical externally but indeed be different in their internal structure. A good metric should be able to look beyond the external features and discriminate two metrics based on their internal structure. According to Weyuker, processes could execute the same function but differ in their implementation and therefore metrics that are implementation independent satisfy this property (Weyuker, 1988).

Two processes with same number of basic and structured activities can return different values for the proposed metrics if decision node types are changed. In addition, IF4BP can detect different values if inputs and output variables are altered. Therefore, all the proposed metrics satisfied this property.

Property 5: For all processes P and Q, considering also the process P;Q obtained by combining P and Q, |P| + |Q| is less than or equal to |P;Q|. This property asserts that two interacting processes may have zero or additional (but never negative) complexity to that which is present in the two initial processes themselves. This complexity is introduced whenever processes interact.

Concatenation of two sets of numbers cannot be negative, but can be zero if the sets are null. The metrics proposed in this study return numeric results and the set of numbers conform to Property 5 since, they have the property ($\exists$P) (P≤P+Q) and (Q≤P+Q) Due to this, all proposed metrics satisfied Property 5.

Property 6: There exist processes P, Q and R such that |P| is equal to |Q| but |P;R| is not equal to |Q;R|. This is an assertion that it is possible to have two identical processes, but when concatenated to a third same process, their resulting complexities are not equal. This is an indicator that the act of combining two processes has the potential of introducing complexity additional to that inherent in the in the original processes. Also, this new added complexity is not completely determined by either of the interacting processes.

The metrics NOBA, NOSA, ALSA, CCBP, ACCSA and SCBP assign fixed values to each of their nodes. Due to the presence of these constants, every time two processes are concatenated, there is no possibility of external complexity being introduced. Therefore, they failed to satisfy this property. However, IF4BP satisfied this property because its components are not physical. This means that IF4BP's complexity is not necessarily increased by adding extra nodes, unless if they contain input and output information.

Property 7: There exist processes P and Q which are composed of the same statements in a permuted order for which |P| is not equal to |Q|. This property argues that the order of statements affects complexity i.e., two identical processes can have different complexity when the order of their statements is changed.

The metrics NOBA, NOSA, ALSA, CCBP, ACCSA and SCBP assign fixed values to each of their nodes. If the lengths and decision node types of two processes are held constant such that the only operation is to permute their order, then the values of ALSA, CCBP, ACCSA and SCBP metrics will remain the same i.e., the two permuted processes will be treated as identical. However, the complexity value of IF4BP is affected when statements order is changed, because messages could be passed to different recipients thus, affecting the results of the transaction. Therefore, Property 7 held true for IF4BP but does not hold for all other metrics.

Property 8: If two processes P and Q differ only in the choice of names for different elements, then |P| is equal to |Q|. This property asserts that two processes are equal if their only difference is the choice of names. It is a suggestion that a metric could fail to discriminate two equal processes as a result of their using different names.

Table 7: Summary of results of complexity metrics validated with Weyuker's properties

Property	NOBA	NOSA	ALSA	IF4BP	CCBP	ACCSA	SCBP
1	✓	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	×	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓	✓
6	×	×	×	✓	×	×	×
7	×	×	×	✓	×	×	×
8	✓	✓	✓	✓	✓	✓	✓
9	×	×	×	✓	✓	✓	✓

Key: ✓ = Satisfied property; × = Unsatisfied property

All proposed metrics return numeric values. This means that renaming a process or its parts cannot affect its length, interface, structural or cognitive complexities. Therefore, all proposed metrics satisfied this property.

Property 9: There exist processes P and Q for which |P|+|Q| is less than |P;Q|. This property asserts that interaction between parts of a process cause additional positive complexity i.e., it makes additional complexity a requirement when two processes keep on interacting for some time, or as the process grows with age. Since, growth in process complexity occurs when new nodes are added and none of the nodes has negative values, then it is clear that the complexity of the new process is always equal to or greater than the sum of the two original processes. Consequently, IF4BP, CCBP, ACCSA and SCBP metrics satisfied Property 9. However, NOBA, NOSA and ALSA failed to satisfy this property because they view process components as having fixed lengths.

A summary of the validation results for NOBA, NOSA, ALSA, IF4BP, CCBP, ACCSA and SCBP metrics are shown in Table 7.

DISCUSSION

Findings in case 1 show that the new metrics are intuitional, for example, processes with more activities returned higher lengths than those with fewer activities. In addition, admission process was found by all complexity metrics to be more complex than loan eligibility process. This is reasonable given that admission process uses parallel processing while loan eligibility process uses branch processing. Parallel processing is expected to be more costly than branch processing because its branches must not only be executed in parallel but the results must also be synchronized.

In case 2, the length metrics NOBA, NOSA and ALSA satisfied all five length requirements while complexity metrics IF4BP, CCBP, ACCSA and SCBP satisfied all five complexity requirements in Briand's framework. This indicates that they are good metrics in their respective categories.

In case 3, both length and complexity metrics were validated together using Weyuker's properties. Findings show that NOBA, NOSA and ALSA failed to satisfy Weyuker's properties 6, 7 and 9. This is because they are one-dimensional, measuring only length and thus, totally ignoring any effects that could be brought about by different types of control-flows. The unsatisfied properties are critical and consequently mean that NOBA, NOSA and ALSA, although good as length metrics, score poorly when treated as complexity metrics. They can however, be used to complement other metrics in order to give a fuller picture of process complexity.

The next metric to be validated in case 3 was IF4BP, which failed to satisfy Property 2. The reason for its failure is because it relies only on input and output information flows and cannot sense any change when the physical size of the process is increased (such as incorporation of new assign activities to the process). The last set of metrics to be validated in case 3 included CCBP, ACCSA and SCBP. Each of these three metrics failed to satisfy Weyuker's properties 6 and 7, because they assign fixed complexity weights to control-flow blocks, which prevents them from detecting extra external complexity that could arise from interactions or from permutation of statements. In contrast, IF4BP satisfied Property 6 since, it is not fixed and not physical. This means that, adding extra activities does not necessarily increase IF4BP's complexity unless if the added activities included input and output information. IF4BP also satisfied Property 7 because changing the order of statements could send messages to different recipients thus, changing the flow of transaction and consequently changing complexity that arises from information flow.

CONCLUSIONS

In this study, seven metrics were proposed for measuring the complexity of EBP. The metrics were evaluated using three case studies. In case 1, values computed from three processes showed that the metrics are intuitional. For example, processes that appear to be more complex returned higher complexity values than their counterparts.

Cases 2 and 3 were complementary theoretical validation studies. Findings from case 2 show that the length metrics satisfied all five length requirements while complexity metrics satisfied all five complexity requirements. In addition, findings from case 3 show that ALSA satisfied 6 out of 9 Weyuker's properties, IF4BP satisfied 8 out of 9 Weyuker's properties, while CCBP, ACCSA and SCBP metrics each satisfied 7 out of 9 Weyuker's properties. Although the unsatisfied Weyuker's properties for the four complexity metrics are

indeed limitations to the concerned metrics, the metrics still satisfied a significant number of properties, which means that their theoretical soundness is acceptable. The length metrics that were given same treatment alongside their complexity counterparts in case 3 showed less significance, but otherwise rated very highly when treated under length category in case 2. Therefore, a conclusion was reached that the new metrics are structurally good metrics when used in their respective categories. The metrics can also be used as a suite so that they complement each other to give a fuller view of EBP complexity.

In future, empirical validation studies need to be conducted on these metrics in order to enhance the validity of the results obtained here.

REFERENCES

Briand, L.C., S. Morasca and V.R. Basili, 1996. Property-based software engineering measurement. IEEE. Trans. Software Eng., 22: 68-86.

Canfora, G., F. Garcia, M. Piattini, F. Ruiz and C.A. Visaggio, 2005. A family of experiments to validate metrics for software process models. J. Syst. Software, 77: 113-129.

Cardoso, J., 2005. Control-flow complexity measurement of processes and weyuker's properties. Proceedings of the 6th International Enformatika Conference, (IEC'05) World Academy of Science, Engineering and Technology, pp: 213-218.

Cardoso, J., 2006. Complexity analysis of BPEL web processes. Software Process: Improve. Practice J., 12: 35-49.

Cardoso, J., J. Mendling, G. Neumann and H.A. Reijers, 2006. A discourse on complexity of process models. Proceedings ofthe BPM 2006 Workshops on Business Process, (BWBP'06), Vienna, Austria, pp: 115-126.

Charfi, A. and M. Mezini, 2007. AO4BPEL: An aspect-oriented extension to BPEL. World Wide Web, 10: 309-344.

Costea, A. 2007. On measuring software complexity. J. Applied Quantitative Methods, 2: 98-108.

Gruhn, V. and R. Laue, 2006. Adopting the complexity measure for business process models. Proceedings of the 5th IEEE International Conference on Cognitive Informatics, (IICCI'06), Beijing, China, pp: 236-241.

Henry, S. and D. Kafura, 1981. Software structure metrics based on information-flow. IEEE Trans. Software Eng., 7: 510-518.

Ince, D.C. and M.J. Shepperd, 1989. An empirical and theoretical analysis of an information flow-based system design metric. LNCS., 387: 86-99.

Koh, T.W., M.H. Selamat and A.A.A. Ghani, 2008. Exponential effort estimation model using unadjusted function points. Inform. Technol. J., 7: 830-839.

Lindsay, A., D. Downs and K. Lunn, 2003. Business processes-attempts to find a definition. Inform. Software Technol., 45: 1015-1019.

Michelson, B., 2005. Business Process Execution Language (BPEL) primer: Understanding an important component of SOA and integration strategies. http://www.psgroup.com/detail.aspx?id=630.

Misra, R.B., S. Srinivasan and D.P. Mital, 2006. The use of web services technology in the design of complex software interfaces: An educational perspective. Inform. Technol. J., 5: 1127-1131.

Misra, S., 2007. Cognitive program complexity measure. Proceedings of the 6th IEEE International Conference on Cognitive Informatics, (IICCI'07), Lake Tahoo, CA., pp: 120-125.

Modafferi, S. and E. Conforti, 2006. Methods for enabling recovery actions in WS-BPEL. Lecture Notes Comput. Sci., 4275: 219-236.

Parthasarathy, S. and N. Anbazhagan, 2006. Analyzing the software quality metrics for object oriented technology. Inform. Technol. J., 5: 1053-1057.

Shao, J. and Y. Wang, 2003. A new measure of software complexity based on cognitive weight. Can. J. Elect. Comput. Eng., 28: 69-74.

Törn, A., T. Andersson and K. Enholm, 1999. A complexity metrics model for software. South Afr. Comput. J., 24: 40-48.

Vanderfeesten, I., J. Cardoso, J. Mendling, H.A. Reijers and W. van der Aalst, 2007. Quality Metrics for Business Process Models. In: Workflow Handbook 2007, Fischer, L. (Ed.). Future Strategies Inc., Lighthouse Point, FL., USA., pp: 179-190.

Wang, Y., 2002. On cognitive informatics, Keynote lecture. Proceeding of the IEEE International Conference Cognitive Information, Aug. 19-20, Calgary, Alberta, Canada, pp: 34-42.

Wang, Y., 2003. Using process algebra to describe human and software behaviours. Brain Mind, 4: 199-213.

Weyuker, E.J., 1988. Evaluating software complexity measures. IEEE Trans. Software Eng., 14: 1357-1365.

Zheng. Y., J. Zhou and P. Krause, 2007. Analysis of BPEL data dependencies. Proceedings of 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, Aug. 28-31, Lubeck, pp: 351-358.