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): 1336-1344, 2010
ISSN 1812-5638


# A Survey of Business Process Complexity Metrics

[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:** Business processes have an inherent complexity which if not controlled can keep on increasing with time, thus making the processes error-prone, difficult to understand and maintain. In the last few years, several researchers have proposed a number of metrics which can be used to measure and therefore control the complexity of business processes. In this study, a survey of business process complexity metrics is conducted with the goal of investigating if there are any gaps in literature. Initially, a description of the process of metrics definition and validation is presented, followed by an analysis of business process complexity metrics that have appeared in literature in the last 5 years. The reviewers checked whether the identified metrics have any tool support, whether they have been validated and whether validation results are significant or not. Findings show that few business process complexity metrics have been proposed so far and that even fewer have been validated. In order to address these issues, some future research directions are proposed.

**Key words:** Business processes, complexity metrics, empirical validation, theoretical validation

## INTRODUCTION

Web-based organizations are nowadays turning to business process modeling so as to gain a competitive advantage in their respective markets. In order to meet this need, many business process modeling languages have been proposed such as the popular Business Process Execution Language (BPEL) (BPEL, 2007). By using BPEL, organizations can for instance create business processes that describe the logic of their business transactions. One problem with business processes is that they tend to grow larger and more complex with age whenever new activities are introduced into the existing process (Cardoso, 2008). Highly complex processes are error-prone, difficult to understand and difficult to maintain. Consequently, high complexity in business processes is undesirable and must be controlled.

Over the years, software metrics have proven to be very effective in controlling complexity and therefore ensuring that high quality software is produced. This idea is also supported by Parthasarathy and Anbazhagan (2006), who observe that if used properly, metrics can allow us to quantify success or failure, improvement and make useful managerial decisions concerning software or processes. Many valuable software product and process metrics have been proposed in the last few decades. Most of these metrics fall under size and complexity categories while others fall under cohesion and coupling. For instance, Al-Hajri *et al*. (2005) proposed a modification of the standard Function Points (FP) measure by replacing its subjective weights with improved weights that are derived using an artificial neural network technique; Koh *et al*. (2008a) proposed an exponential effort estimation model that eliminates FP's general system characteristics; and Atan *et al*. (2007) proposed a suite of six metrics to measure software process models. Koh *et al*. (2008b) provided more insights on software complexity metrics for object oriented software in their survey paper. The success enjoyed by software metrics has inspired business researchers to propose some useful complexity metrics with the goal of ensuring that enacted business processes are of high quality. The term complexity has been used by metrics researchers to refer to how difficult an entity is to understand. The problem is that very few business process metrics have been proposed so far (Ghani *et al*., 2008; Cardoso *et al*., 2006) and even fewer

**Corresponding Author:** Geoffrey M. Muketha, Department of Information Systems,
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia,
43400 Serdang, Selangor, Malaysia Tel: +6 017 369 8350 Fax: +6 03 8946 6577

appear to have been validated, which is worrying because metrics cannot be assumed valid until validated. This means that there could be many information gaps that are not fully understood and consequently, not yet addressed.

In this study, a literature review of business process complexity metrics is conducted. The main goal of this survey paper is to shed more light on what business process complexity metrics exist, whether these metrics have any tool support, whether they have been validated and whether there any gaps in literature which need to be investigated further.

## METRICS DEFINITION AND VALIDATION STEPS

There is consensus between metrics researchers (Serrano *et al*., 2002; Soni *et al*., 2009) that only three steps are needed to define and validate metrics. These steps include defining new metrics, validating them theoretically and then validating them empirically. These steps are shown in Fig. 1. A fourth step, which is not included in Fig. 1 due to the fact that it is optional, is metrics tooling. Tooling helps in automating the process of metrics capture and computation and is therefore desirable.

**Metrics definition:** New metrics are defined based on the rigorous metrics definition model described by Fenton and Pfleeger (1997). According to Fenton and Pfleeger (1997), the process of defining new metrics involves three steps: identify measurement entity, identify attributes of the entity that are to be measured and then define new metrics that can be used to measure each attribute. These steps should be executed in the order in which they appear.

An entity is an object such as a software module, while an attribute is a measurable property of the object. Entities fall into three categories, namely, products, processes and projects (Fenton and Pfleeger, 1997). A process is any activity related to software development, a product is any artifact produced during software development and a resource is people, hardware, or software needed for the processes (Fenton and Pfleeger, 1996).

Attributes are properties of entities such human height. Attributes may be grouped into two main categories: internal and external. Internal attributes are measured directly from the entity. They are the easiest to measure. Such measures are also called direct measures due to the direct way in which they are measured. A good example of an internal product attribute is length and a frequently used direct measure is to count the number of lines of source code of a software program. On the other hand, external attributes are indirect and are measured as derivatives of internal attributes. An example of an external product attribute is quality, which may be measured by for example counting the number of bugs found per class. It can then be interpreted to mean that the lesser the bugs the higher the quality of the class. Figure 2 shows the metrics definition process, with the arrow indicating direction of process execution.

While defining new metrics, the representational theory of measurement should be followed in order to avoid defining flawed metrics. A good description of the representational theory of measurement can be found in Fenton and Pfleeger (1997). According to Fenton and Pfleeger (1997), measurement may be defined as the process of assigning numbers or symbols to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.

The actual definition of metrics may be done by studying the behavior of attributes and entities. Metrics can also be adapted from related fields. Adaptation, which involves technology reuse, is very attractive because it has the potential to significantly cut on cost and effort of

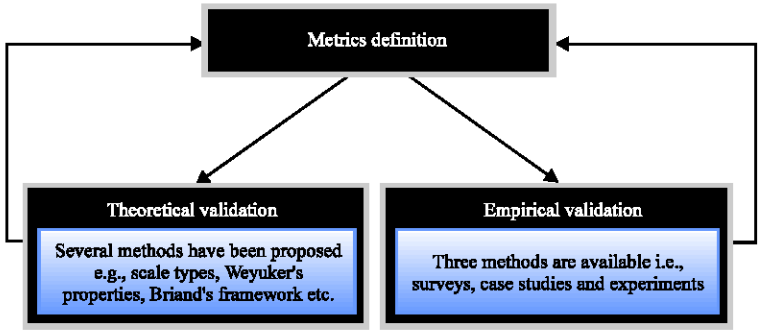


Fig. 1: Metrics definition and validation steps

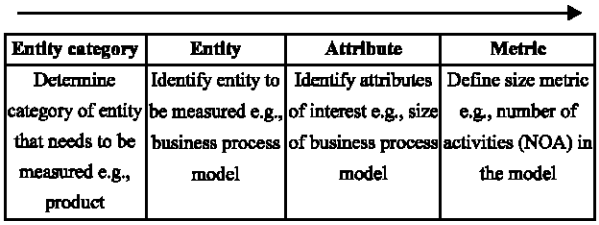


Fig. 2: Metrics definition process

metrics definition. However, reuse is only possible where the primary domain of the technology and the new domain are closely related such that the cost of extending the technology is minimal. Several authors have observed that there are many similarities between software programs and business processes (Ghani *et al*., 2008; Vanderfeesten *et al*., 2007; Cardoso *et al*., 2006). The notion that processes can be treated like software was first coined by Osterweil, who advocated process programming arguing that technologies that are used to build application software can also be used to build processes since the two bear structural similarities (Osterweil, 1997). However, the two fields have some differences too and software metrics cannot be consumed directly by business processes. Sometimes the differences are only in the naming of identifiers, in which case mapping is easy and straight forward. Other differences are more fundamental, for example there are semantic differences between certain control-flow structures of business processes and those of software programs, as is the case with BPEL's pick structure, which instead of simply testing a Boolean condition like ordinary if-else structures, waits for an on Message or on Alarm message before it can execute. Another case is the handling of parallel processing, which is implemented in BPEL using the flow structure. The flow structure executes several activities in parallel and then synchronizes their results. This makes parallelism more costly than either iteration or branch structures. While parallel programming is not a new concept in software engineering (Qureshi and Manuel, 2006), many imperative and object oriented languages are yet to implement it. These differences need to be taken into consideration if adaptation is to be a viable option for obtaining new business process metrics.

**Theoretical validation:** Here, three theoretical validation methods that are frequently cited in business process literature are presented. These methods include: (1) identifying scale types of new metrics, (2) checking whether the metrics satisfied Briand's generic measurement framework and (3) checking whether the metrics satisfied Weyuker's properties. The main aim of theoretical validation is to establish whether the new metrics are structurally sound and if they do not violate measurement theory.

**Scale types:** Identifying the scale types of newly defined metrics is perhaps the first form of theoretical validation that the metrics should be subjected to. A knowledge of the scale type of a metric helps to determine what transformations are admissible on the metric. It also sheds light on the meaningfulness of the measures that the metric will generate (Fenton and Pfleeger, 1997). This is because a set of admissible transformations exist for each scale type. As an example, if a person's weight is represented with two different units such as kilograms and pounds, then the weight of the person should not be distorted since the two units are both related by an admissible relation. Therefore, a measure is meaningful if its truth or false values remain unchanged when different units are used. Scale types also allow us to determine what set of computations can be performed on them and which ones cannot be performed on them. Table 1 shows these scale types together with their admissible transformations.

**Briand's generic measurement framework:** Briand *et al*. (1996) have proposed a framework that categorizes metrics into size, length, complexity and coupling and cohesion. Each of the five categories proposes a set of properties which need to be satisfied by any metric falling under that category.

This section presents a summary of size, length and complexity categories from Briand's framework. Size and length are of interest to business process researchers since the two categories constitute a form of complexity called activity complexity (Cardoso, 2006). Coupling and cohesion metrics are excluded in this study since they belong to a different category other than complexity. Briand's framework was meant to validate metrics for modular software programs, but an effort was made to map it into the business process perspective in order to be able to utilize it effectively. While some business process modeling languages have sub-programs which can be treated as modules, others such as BPEL lack sufficient modularity and flexibility features (Charfi and Mezini, 2007). Charfi and Mezini (2007) have attempted to address these issues using an aspect-oriented approach for BPEL while Zhang and Du (2009) have proposed a process generation approach that can make workflow processes more flexible. In the case of BPEL, an assumption can be made that the equivalent of a module is a structured activity structure. Some simple processes that contain only one structured activity in them may also be treated as modules.

Table 1: Scales of measurement (Adapted from Fenton and Pfleeger, 1997)

| Scale type | Admissible transformation (how measures M and M' must be related) | Examples |
|---|---|---|
| Nominal | 1-1 mappings from M to M' | Labeling, classifying entities |
| Ordinal | Monotonic increasing function from M to M' i.e., M' i.e., M (x)≥M (y) implies M' (x)≥M (y) | Preference, hardness, air quality, intelligence tests (raw scores) |
| Interval | M' = aM+b (a>0)) | Relative time, temperature (Fahrenheit, Celsius intelligence tests (standardized scores) |
| Ratio | M' = aM (a>0) | Length, time intervals, temperature (Kelvin) |
| Absolute | M' = M | Counting entities |

**Size of the process:** The size of a process P is a function Size (P) that is characterized by the following three properties which need to be satisfied by size metrics.

- **Size 1:** The size of a process is nonnegative
- **Size 2:** The size of a process is null if the process is empty
- **Size 3:** The size of a process is equal to the sum of the sizes of two of its modules (e.g., sub-process, scope etc.) such that any element of the process is an element of either the first or the second module

**Length of the process:** The length of a process P is a function Length (P) that is characterized by the following five properties which need to be satisfied by length metrics.

- **Length 1:** The length of a process cannot be negative, but can be null if the process has got no elements
- **Length 2:** The length of a process is null if the process is empty
- **Length 3:** Adding relationships between elements of a connected component of a process does not increase the length of the process
- **Length 4:** Adding relationships from elements of two separate connected components of a process does not decrease the length of the process
- **Length 5:** The length of a process that is composed of two disjoint modules is equal to the maximum of the lengths of the two modules

**Complexity of the process:** The complexity of a process P is a function complexity (P) that is characterized by the following five properties which need to be satisfied by complexity metrics.

- **Complexity 1:** The complexity of a process cannot be negative, but can be null if a system has got no elements.
- **Complexity 2:** The complexity of a process is null if the process has got no structured activities in it
- **Complexity 3:** The complexity of a process does not depend on the convention chosen to represent the relationships between its elements
- **Complexity 4:** The complexity of a process is no less than the sum of the complexities of any two of its modules with no relationships in common
- **Complexity 5:** The complexity of a process composed of two disjoint modules is equal to the sum of the complexities of the two modules

**Weyuker's properties:** Weyuker (1988) has proposed nine properties (also called axioms) for validating complexity metrics. These properties were initially intended to validate only complexity metrics. This is because Weyuker did not attempt to address metrics that are outside the scope of complexity. For this reason, simple one-dimensional metrics such as size or length metrics always show weaknesses when validated with Weyuker's properties. Weyuker's properties have therefore been much criticized for failing to cater for non complexity metrics. They have also been criticized for being unsuitable for validating object oriented metrics. One reason for being unsuitable for object oriented metrics is that object oriented languages themselves have already addressed those structural weaknesses that Weyuker was trying to address.

Despite these criticisms, Weyuker's properties are one of the most widely cited frameworks for theoretical validations of metrics. Weyuker's properties are very much valid for business process metrics validations because business processes such as those created with BPEL language have a similar structure to procedural programs. It is also worth noting that Weyuker's properties have been extended to cater for business processes (Cardoso, 2008). The nine Weyuker's properties are summarized below.

- **Property 1:** 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
- **Property 2:** This property asserts that a changing process must also cause a change to its complexity. A good metric should be able to detect this change
- **Property 3:** This property asserts that there exist two different processes whose data types and values are identical but whose variable names differ. A good metric should return same complexity for such processes

- **Property 4:** 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
- **Property 5:** 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 and a good metric should be able to detect it
- **Property 6:** This property asserts 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. A good metric should be able to discriminate between two such processes
- **Property 7:** This property asserts that the order of statements affects complexity i.e., two identical processes can have different complexity when the order of their statements is changed. A good metric should be able to detect this change
- **Property 8:** If two processes differ only in the choice of names for different elements, then two processes are equal. A good metric should be able to return the same value for such processes.
- **Property 9:** 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. A good metric should be able to detect this change in behavior

**Empirical validation:** Empirical strategies available to software engineers and business process researchers include surveys, case studies and experiments (Wohlin *et al*., 2000; Fenton and Pfleeger, 1997). The main aim of empirical validation is to establish whether the new metrics are measuring what they were intended to measure and is complementary to theoretical validation. Therefore, both theoretical and empirical validations are needed in order to avoid the problem of defining a theoretically sound metric which is otherwise useless in its practical application.

**Surveys:** According to Wohlin *et al*. (2000), surveys are retrospective. This means that they are used to investigate some tool or technique that has been in use for a while. Normally, a sample is taken from some population and later the analyzed results are generalized to the population.

**Case studies:** Case studies are used for monitoring ongoing activities. Wohlin *et al*. (2000) states that the aim of case studies is to track an attribute or to establish a relationship between several attributes, which makes it an observational study.

**Experiments:** Experiments are formal, rigorous and repeatable. Many authors agree that experiments have more control of variables than case studies (Wohlin *et al*., 2000; Fenton and Pfleeger, 1997). This is because they are typically conducted in laboratories, where subjects are assigned to different treatments at random. The main goal of experiments is to keep one or more variables constant while manipulating all other variables. Consequently, the effects of this manipulation are measured and interpreted.

In this study, it was found out that experiments were the most frequently used empirical validation method. By taking advantage of the fact that experiments are repeatable, some authors validated their metrics in families of replica experiments. Replicating experiments in a different setting or with different subjects increases validity of the results and is good for the industry.

**Metrics tool implementation:** Metrics tools are important in that they help to automate the process of metrics collection and calculation. Metrics tools are language recognizers and design considerations vary widely depending on requirements. One strategy is to design a tool with a metrics computation component, a storage component and a presentation component. The presentation component serves as the user interface while the storage component keeps track of metrics data.

In addition, some metrics tools are designed to recognize one modeling language such as the Software Process Measurement Application (SPMA) tool (Atan *et al*., 2007). Other tools such as the ProM framework are more comprehensive and can recognize multiple modeling languages (Van Dongen *et al*., 2005).

## COMPLEXITY METRICS FOR BUSINESS PROCESSES

Several researchers have attempted to classify software metrics in the past. One notable effort to categorize business process complexity metrics was done

by Cardoso (2008). Such an effort is not made here mainly because it was felt that very few metrics exist to warrant it. It was therefore left out for a later date when there will be more metrics and better empirical knowledge in the field of business process measurement.

The following sections present existing metrics grouped by author. Identification of the metrics was based on whether they fell within the scope of business process complexity. Other metrics that are slightly outside the scope of complexity (such as certain error metrics) but which were otherwise intended to serve as pointers to complexity were also included. In most cases, the authors did not indicate whether there is any tool support or not. Also, where empirical studies are recorded, they normally took the form of experiments. To the best of our knowledge, the authors whose metrics are described here are also some of the most frequently cited in the field of business process complexity metrics. Although it is possible that some good papers could have possibly been left out due to limitations in our search criteria, an effort was made to present what has been happening in this field in the last 5 years.

**Cardoso metrics:** Cardoso has proposed the Control-Flow Complexity (CFC) metric for measuring control-flow complexity of business process models (Cardoso, 2006, 2008). CFC is an adaptation of McCabe's cyclomatic complexity metric (McCabe, 1976). While McCabe's cyclomatic complexity assigns same semantics to all decision nodes, CFC distinguishes the various nodes as having different semantics such as AND-splits, XOR-splits, OR-joins etc. CFC is computed by simply adding the CFC of all split constructs (Cardoso, 2008). This metric has been validated with Weyuker's properties (Cardoso, 2008) and with several experiments (Cardoso, 2008; Rolon *et al*., 2008). In the experiments, Spearman's correlation coefficients were used to test the hypothesis and results were significant.

In a separate study, Cardoso *et al*. (2006) have proposed a set of metrics that have been adapted from software engineering. These metrics include the number of activities in a process (NOA), the number of activities and control-flows in a process (NOAC), the number of activities, joins and splits (NOAJS), the Interface Complexity (IC) and Halstead-based Process Complexity (HPC).

The first three metrics are adapted from the Lines of Code (LOC) metric that has been used in software engineering for many years to measure length of software. According to Cardoso *et al*. (2006), these simple process metrics constitute a type of complexity called activity complexity.

Another type of metric proposed by Cardoso *et al*. (2006) is called Interface Complexity (IC). IC is adapted from the information flow metric (Hennry and Kafura, 1981). The IC metric is computed as follows:

$$IC = length^* (No.\ of\ inputs^*\ No.\ of\ outputs)^2$$

where, inputs are incoming data flows and output is the outgoing data flows.

The Halstead metrics model computed from operands and operators present in a program. Cardoso *et al*. (2006) have proposed a metric called Halstead-based Process Complexity (HPC) based on Halstead metrics. The authors of HPC metric replace number of unique operators with number of unique activities, splits and joins and control-flow elements. They also replace number of unique operands with number of unique activities, splits and joins and control-flow elements. Number of unique operands is replaced by number of unique data variables being manipulated by the process and its activities. From these, they derive the equivalent of total number of operator occurrences and total number of operand occurrences as well as length, volume and difficulty.

Cardoso *et al*. (2006) have also proposed three graph-oriented metrics, namely, the Coefficient of Network Complexity (CNC), the Complexity Index (CI) and the Restrictiveness Estimator (RE). CNC measures complexity of a graphic and is defined as the number of arcs divided by number of activities, joins and splits. CI is the minimal number of node reductions that reduce the graph to a single node. RT estimates the number of feasible sequences in a graph. A summary of these metrics is shown in Table 2.

**Gruhn and Laue metrics:** Gruhn and Laue (2006) have proposed the cognitive weight for business process models. This metric is an adaptation of the Cognitive Functional Size (CFS) proposed earlier for software measurement by Shao and Wang (2003). Cognitive complexity metrics are based on cognitive informatics. The proponents of cognitive complexity metrics contend that there are three factors that lead to complexity, namely, internal architecture of software, input data flowing into the module and output data flowing out of the module. This means that cognitive complexity is a function of these three factors (Shao and Wang, 2003). Unlike its software counterpart, this metric is designed for use with graphical business process models that emphasize on visual communication with users but offer minimal formal semantics. As a consequence of this design consideration, the main limitation of this metric is that it

Table 2: Summary of complexity metrics for business processes

| Authors | Metrics | Tool support | Theoretical validation | Significant results | Empirical validation | Significant results |
|---|---|---|---|---|---|---|
| Cardoso (2008) | CFC | Unknown | Yes | Yes | Yes | Yes |
| Cardoso *et al.* (2006) | NOA, NOAC, NOAJS, IC, HPC, CNC, CI, RI | Unknown | No | No | No | No |
| Gruhn and Laue (2006) | CW for BPM, NOA, information flow, max/mean nesting depth, No. of handles, (anti) patterns for BPM | Unknown | No | No | No | No |
| Lassen and van der Aalst (2009) | ECaM, ECyM, SM | Yes | No | No | Yes | Yes |
| Vanderfeesten *et al.* (2008) | CC | Unknown | No | No | Yes | Partial |
| Mendling and Neumann (2007) | $S_N$ (G), $\Pi$ (G), $\Xi$ (G), $\phi_N$, $CYC_N$, TS (G) | No | No | No | Yes | Partial |

Unknown: No clear indication of presence or absence of tools; Yes: Agreement; No: Disagreement; Partial: Some hypotheses were accepted while others were rejected

ignores two of the three factors that comprise cognitive complexity, namely, inputs and output data flows and concentrates only on control-flows.

Gruhn and Laue (2006) have also proposed adapting the information flow metric (Hennry and Kafura, 1981) for business processes. They define information flow as (Fan-in * Fan-out)$^2$. Unlike the case of the IC (Cardoso *et al.*, 2006), this metric does not include length of the process in its formula.

Other metrics proposed by Gruhn and Laue (2006) include number of activities, maximum/mean nesting depth, number of handles (which measures well-structuredness of a model) and (anti)patterns for business process models (which counts usage of anti-patterns in a business process models and requires experience). Gruhn and Laue (2006) also separately proposed the number of activities in a process. This metric was also proposed by Cardoso *et al.* (2006) around the same time. Gruhn and Laue metrics have not yet been validated. A summary of these metrics is shown in Table 2.

**Lassen and Aalst metrics:** Lassen and van der Aalst (2009) have proposed three complexity metrics for a subclass of Petri Nets called Workflow nets. These metrics include extended Cardoso metric (ECaM), extended cyclomatic metric (ECyM) and Structuredness Metric (SM). ECaM extends CFC in that it is tailor-made to support Petri Nets, while ECyM extends cyclomatic metric by measuring graph reachability rather than graph structure. On the other hand, the SM metric measures well-structuredness in the design structure. These metrics have been implemented in the Prom framework (Van Dongen *et al.*, 2005). ProM is a business process metrics tool that focuses on Business Activity Monitoring (BAM). ProM allows many plug-ins, which makes it easy to extend. It also integrates the functionality of several existing process mining tools but also provides additional features. ProM supports several business process modeling languages such as Petri Nets, Event-driven Process Chains (EPC), Social Networks etc. Lassen and Aalst metrics have been validated in an experiment in which Pearson product correlations were used. The results were found to be significant. A summary of these metrics is shown in Table 2.

**Vanderfeesten metric:** Vanderfeesten *et al.* (2007) have proposed a metric called Cross-Connectivity (CC) metric based on cognitive complexity. This is an error prediction metric that measures the strength of the links between process model elements. It is based on the hypothesis that process models are easier to understand and contain fewer errors if they have a high cross-connectivity. In addition to predicting errors, it can also measure understandability of a business process model. This metric has been evaluated empirically using Spearman correlation coefficients and multivariate logistic regression. Results were significant for error prediction hypothesis, but were insignificant for understandability hypothesis. However, the authors argue that CC can explain variation in understandability when combined with existing metrics. A summary of this metric is shown in Table 2.

**Mendling and Neumann metrics:** Mendling and Neumann (2007) have proposed six error metrics that are closely related to complexity. These metrics are based on graph theory and include size, separability, sequentiality, structuredness, cyclicity and parallelism. Size, $S_N$ (G), counts the number of nodes in a graph G and an increase in size increases error probability. Separability, $\Pi$ (G), is the ratio of cut-vertices to the number of nodes. Cut-vertices are those whose deletion separates the process model into multiple components. An increase in separability implies decrease in error probability. Sequentiality, $\Xi$ (G), takes sequences as the building blocks and computes a ratio of arcs of a sequence to the total number of arcs. Increasing sequentiality reduces error probability. Structuredness, $\phi_N$, is how far a process model can be built by nesting blocks of matching join and split connectors. Its increase implies decrease in error probability. Cyclicity, $CYC_N$, relates number of nodes on some cycle with total number of nodes. An increase in

cyclicity implies an increase in error probability. Parallelism, TS (G), is related to the number of concurrent paths that must be synchronized. Increasing parallelism implies increased error probability. The Mendling and Neumann metrics were evaluated empirically and results for size, separability and structuredness metrics were significant. Results for the other three metrics were insignificant. A summary of these metrics is shown in Table 2.

## CONCLUSIONS

The investigation in this study indicates that only a small number of the proposed metrics have been validated either theoretically or empirically. Moreover, most of the authors do not categorically state whether they created any tools to support their metrics. In addition, most of the validations conducted were empirical and not theoretical. Where empirical validation took place, experiments were the main method of choice, with some experiments showing mixed results. This trend is worrying because metrics validation is the only way to ensure that the new metrics are theoretically sound and that they are measuring what they were intended to measure. These two goals of validation can easily be achieved if both theoretical and empirical validations are conducted since they complement each other.

In order to address the issues raised in this study concerning the current state of business process complexity metrics, future research efforts should be directed towards defining new complexity for business processes and then validating them both theoretically and empirically. Another research direction would be to implement metrics tools for automating business process measurement.

## REFERENCES


Al-Hajri, M.A., A.A.A. Ghani, M.H. Selamat and M.N. Sulaiman, 2005. Modification of standard function point complexity weights system. J. Syst. Software, 74: 195-206.

Atan, R., A.A.A. Ghani, M.H. Selamat and R. Mahmod, 2007. Software process modeling using attribute grammar. Int. J. Comput. Sci. Network Sec., 7: 273-281.

BPEL, 2007. Web services business process execution language version 2.0. http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.html.

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

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.

Cardoso, J., 2008. Business process control-flow complexity: Metric, evaluation and validation. Int. J. Web Serv. Res., 5: 49-76.

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

Fenton, N.E. and S.L. Pfleeger, 1997. Software Metrics: A Rigorous and Practical Approach. 2nd Edn., International Thomson Publishing, Boston, ISBN: 0-534-95425-1.

Ghani, A.A.A., K.T. Wei, G.M. Muketha and W.P. Wen, 2008. Complexity metrics for measuring the understandability and maintainability of business process models using Goal-Question-Metric (GQM). Int. J. Comput. Sci. Network Secur., 8: 219-225.

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.

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

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

Koh, T.W., M.H. Selamat and A.A.A. Ghani, 2008b. Review of complexity metrics for object oriented software products. Int. J. Comput. Sci. Network Sec., 8: 314-320.

Lassen, K.B. and W.M.P. van der Aalst, 2009. Complexity metrics for workflow nets. Inform. Software Technol., 51: 610-626.

McCabe, T.J., 1976. A complexity measure. IEEE Trans. Software Eng., 2: 308-320.

Mendling, J. and G. Neumann, 2007. Error metrics for business process models. Proceedings of the 19th International Conference on Advanced Information Systems Engineering, (CAISE'07), USA., pp: 53-56.

Osterweil, L.J., 1997. Software processes are software too revisited: An invited talk on the most influential paper of ICSE 9. Proceedings of the 19th International Conference on Software Engineering, May 17-23, Boston, Massachusetts, United States, pp: 540-548.

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

Qureshi, K. and P. Manuel, 2006. A survey of concurrent object-oriented languages (Cools). Inform. Technol. J., 5: 601-611.

Rolon, E., J. Cardoso, F. Garcia, F. Ruiz and M. Piattini, 2008. Analysis and validation of control-flow complexity measures with BPMN process models. LNBIP, 29: 58-70.

Serrano, M., C. Calero and M. Piattini, 2002. Validating metrics for data warehouses. IEEE Proc. Software, 149: 161-166.

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

Soni, D., R. Shrivastava and M. Kumar, 2009. A framework for validation of object-oriented design metrics. Int. J. Comput. Sci. Network Secur., 6: 46-52.

Van Dongen, B.F., A.K.A. de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters and W.M.P. van der Aalst, 2005. The prom framework: A new era in process mining tool support. LNCS, 3536: 444-454.

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. (Eds.). Future Strategies Inc., Lighthouse Point, FL., USA., pp: 179-190.

Vanderfeesten, I., H.A. Reijers, J. Mendling, W. van der Aalst and J. Cardoso, 2008. On a quest for good process models: The cross-connectivity metric. LNCS, 5074: 480-494.

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

Wohlin, C., P. Runeson, M. Host, M.C. Olsson, B. Regnell and A. Wesslen, 2000. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Hingham, MA, USA.

Zhang, F. and Y.Y. Du, 2009. A process generation approach of dynamic workflows based description logics. Inform. Technol. J., 8: 998-1005.