International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:1, No:8, 2007

Control-flow Complexity Measurement of Processes and Weyuker's Properties

Jorge Cardoso

***Abstract*—**Process measurement is the task of empirically and objectively assigning numbers to the properties of business processes in such a way as to describe them. Desirable attributes to study and measure include complexity, cost, maintainability, and reliability. In our work we will focus on investigating process complexity. We define process complexity as the degree to which a business process is difficult to analyze, understand or explain. One way to analyze a process' complexity is to use a process control-flow complexity measure. In this paper, an attempt has been made to evaluate the control-flow complexity measure in terms of Weyuker's properties. Weyuker's properties must be satisfied by any complexity measure to qualify as a good and comprehensive one.

***Keywords*—**Business process measurement, workflow, complexity.

I. Introduction

BUSINESS Process Management Systems (BPMS) provide a fundamental infrastructure to define and manage business processes. BPMS, such as Workflow Management Systems (WfMS), have become a serious competitive factor for many organizations that are increasingly faced with the challenge of managing e-business applications, workflows, Web services, and Web processes.

Recently, a new field of research for processes has emerged. This new field – termed process measurement – presents a set of approaches to the quantification of specific properties of processes. Important properties to analyze include the estimation of complexity, defects, process size, effort of testing, effort of maintenance, understandability, time, resources, and quality of service. Process measurement is still in its infancy and much work has yet to be undertaken.

Process measurement can and should be used in every phase of the process development life-cycle, including the analysis, design, implementation, testing, and maintenance phases. Process measurement provides business process engineers and managers with a forecast of the characteristics of processes early in the development stage so that corrective actions can be taken, if necessary, when the cost is low.

In [1] we have presented a Control-Flow Complexity (CFC) measure to analyze the degree of complexity of business processes. Process complexity can be defined as the degree to which a business process is difficult to analyze, understand or explain. The use of the CFC measure allow designers to create less complex processes, thus reducing the time spent reading and understanding processes in order to remove faults or adapt the process to changed requirements. Nowadays, complexity analysis has an increased importance since the emergence of processes that span both between and within enterprises have an inherent higher complexity. Therefore, methods should be used to support the design and redesign of processes to reduce their complexity. The CFC can be used to analyze the complexity of business processes, as well as workflow and Web processes.

In this paper, our objective is to evaluate the control-flow complexity measure presented in [1] in terms of Weyuker's properties [2]. Weyuker's properties give an important basis to classify a complexity measure to determine if it can be categorized as a good, structured, and comprehensive one.

J. Cardoso is with the Department of Mathematics and Engineering, University of Madeira, 9050-390 Funchal, Portugal (phone: 291-705-156; fax: 291-705-199; e-mail: jcardoso@uma.pt).

II. Perspectives to Process Complexity

There is no single metric that can be used to measure the complexity of a process. Four main complexity perspectives can be identified (Fig. 4): activity complexity, control-flow complexity, data-flow complexity, and resource complexity. While in this paper we will focus on control-flow complexity, we present the main ideas behind each complexity perspective.

Activity complexity: This view on complexity simply calculates the number of activities a process has. While this complexity metric is very simple, it is very important to complement other forms of complexity. The control-flow complexity of a process can be very low while its activity complexity can be very high. For example, a sequential process that has a thousand activities has a control-flow complexity of 0, whereas its activity complexity is 100.

Control-flow complexity: The control-flow behavior of a process is affected by constructs such as splits, joins, loops, and ending and starting points. Splits allow defining the possible control paths that exist in a process. Joins have a different role; they express the type of synchronization that should be made at a specific point in the process.

Data-flow complexity: The data-flow complexity of a process increases with the complexity of its data structures, the number of formal parameters of activities, and the mappings between activities' data. A data-flow complexity metric can be composed of several sub-metrics which include:

data complexity, interface complexity, and interface integration complexity [3].

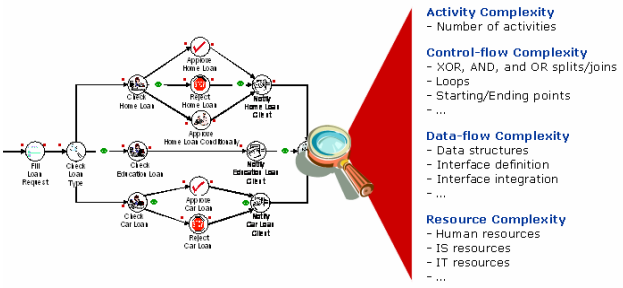

Fig. 1 Types of complexity analysis

Resource complexity: Activities in a process need to access resources during their executions. The different types of resources can be analyzed to determine the complexity of a process. This analysis can help managers to lower administrative costs and better optimize resource utilization.

III. The Control-Flow Complexity Metric

For our investigation of complexity we use the following practical definitions related to the control-flow complexity metric.

Definition 1 (Process):
A process is a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer. A process is a specific ordering of activities across time and place, with a beginning, an end, and clearly identified inputs and outputs.

Definition 2 (Process Property)
A property is a feature, characteristic or attribute of a process, such as complexity, maintainability, cost, reliability, etc. Process properties can be evaluated and quantified using suitable models, methods, and algorithms.

Definition 3 (Process Measure)
A process measure is an empirical assignment of numbers (or symbols) to processes to characterize a specific property.

Definition 4 (Process Measurement)
Process measurement is the task of applying measures to processes in such a way as to describe them.

Definition 5 (Control-flow Graphs)
Control-flow graphs can be used to describe the logic structure of processes. A process is composed of activities and transitions. Activities are represented using circles and transitions are represented using arrows. Transitions express dependencies between activities. An activity with more than one outgoing transition can be classified as an and-split, or-split or xor-split. And-split activities enable all their outgoing transitions after completing their execution. Or-split Activities enable one or more outgoing transition after completing their execution. Xor-split activities enable only one outgoing transition after completing their execution. And-split activities are represented with a '•', or-split are represented with a 'O' and xor-split activities are represented with a '⊕'. An activity with more than one incoming transition can be classified as an and-join, or-join or xor-join. And-join Activities start their execution when all their incoming transitions are enabled. Or-join activities start their execution when a subset of their incoming transitions is enabled. Xor-join activities are executed as soon as one of the incoming transitions is enabled. As with and-split, or-split and xor-split activities, and-join, or-join and xor-join activities are represented with the symbols '•', 'O' and '⊕', respectively.

Definition 6 (Fan-out)
Fan-out is the number of transitions going out of an activity.

Definition 7 (Control-flow induced state)
We map the control-flow complexity into the space of possible execution states of a process. An induced state is a state that can be reached from a particular activity. Splits introduce the notion of states in processes. When a split (XOR, OR, or AND) is added to a process, the activities connected to its outgoing transitions form the states that can be reached from the split.

Definition 8 (XOR-split Control-flow Complexity)
XOR-split CFC ($CFC_{XOR\text{-}split}(activity_i)$) is determined by the number of induced states that are introduced with the split. For XOR-splits, the complexity corresponds to the fan-out of the split, i.e. the number of states that follow the XOR-split that the process designer needs to consider, analyze, and assimilate.

Definition 9 (OR-split Control-flow Complexity)
OR-split CFC ($CFC_{OR\text{-}split}(activity_i)$) is also determined by the number of induced states that are introduced with the split. For OR-splits, the complexity corresponds to 2^n-1, where n is the fan-out of the split. OR-splits lead to higher control-flow complexity than an XOR-split or AND-split since they originate a greater number of induce states.

Definition 10 (AND-split Control-flow Complexity)
As with the previous types of splits, an AND-split CFC ($CFC_{AND\text{-}split}(activity_i)$) is determined by the number of induced states that are introduced with the split. For AND-splits, the complexity is simply 1. The designer constructing a process needs only to consider and analyze one state that may arise from the execution of an AND-split construct since it is assumed that all the outgoing transitions are selected and executed.

Definition 11 (Control-flow Complexity)
The complexity of process is connected to effects such as readability of processes, understandability, effort, testability,

reliability and maintainability. The Control-flow Complexity (CFC) is calculated by adding the CFC of all split constructs presents in a process.

The CFC metric was inspired from the branch of software engineering known as software metrics, namely from the McCabe's Cyclomatic complexity [4]. In processes, the McCabe's Cyclomatic complexity cannot be used directly since the metric ignores the semantics associated with nodes of the graph. While the nodes (i.e. activities) of processes have distinct semantics (e.g. different types of splits and joins), the nodes of a program's flowgraph are undifferentiated. Therefore, we calculated the control-flow complexity for a process *P* as follows:

$$CFC(P) = \sum_{i\in\{\text{XOR-splits of P}\}} CFC_{XOR-split}(i) + \sum_{j\in\{\text{OR-splits of P}\}} CFC_{OR-split}(j) + \sum_{k\in\{\text{AND-splits of P}\}} CFC_{AND-split}(k) \quad (1)$$

The greater the value of the CFC(P) the greater the overall architectural complexity of a process. CFC(P) analysis seeks to evaluate complexity without direct execution of processes.

IV. Control-Flow Complexity and Weyuker's Properties

Weyuker properties have been applied to software engineering and have been seriously discussed in the literature [5-8]. Weyuker properties are a widely known formal analytical approach and were therefore chosen for our analysis since they do provide a basis for some validation of complexity metrics. As shown by Weyuker, with such properties it is possible to filter out measurements with undesirable properties. The majority of these properties are formulated in a clear way. This is an advantage because we are able to discuss them.

A. Summary of Weyuker's Properties

Weyuker's first property states that a metric cannot measure all software programs as being equally complex. The second property states that there are only a finite number of programs of the same complexity. The third property states that each different program may be complex. The fourth property states that the complexity of a program depends on its implementation and that even if two programs solve the same problem, they can have different complexities. Weyuker's fifth property states that the complexity of two programs joined together is greater than the complexity of either program considered separately. The sixth property states that a program of a given complexity when joined to two other programs does not necessarily mean the resulting program will be of equal complexity, even if the two added program are of equal complexity. Weyuker's seventh property states that a permuted version of a program can have a different complexity, so the order of statements matters. The eighth property states that if a program is a straight renaming of another program, its complexity should be the same as the original program. The final property states the complexity of two programs joined together may be greater than the sum of their individual complexities.

B. Concatenation Operations on Processes

Weyuker introduces the concatenation operation (P1;P2) of program blocks. Weyuker defines the concatenation operation in the following way: a program can be uniquely decomposed into a set of disjointed blocks of ordered statements having the property whenever the first statement in the block is executed; the other statements are executed in the given order.

In our approach and since we are dealing with processes, four concatenation operations exist. Processes can be concatenated either sequentially, using an AND, an OR, or a XOR. Every AND/OR/XOR split has also a corresponding AND/OR/XOR join and the different splits do not overlap each other. We have decided to only allow the construction of well structured processes [9] which are based on a set of predefined building blocks. This protects users from designing invalid processes. Aalst [9] has shown that processes that are not well structured contain design errors, such as non-termination, deadlocks, and spliting of instances. We use Weyuker's properties to evaluate the CFC metric assuming that the processes are well-structured for simplicity reasons. The CFC metric can be applied to well-structured and unstructured processes.

In the list of properties below, P, Q and R represent processes and the complexity of P computed by our complexity measure CFC(P) is represented by |P|.

1) When a process P is concatenated sequentially with a process Q, we depict the resulting process as P-Q. This type of concatenation is illustrated in Fig. 2.

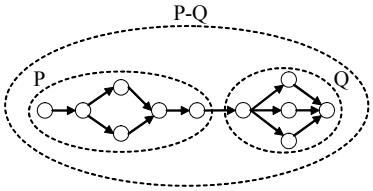

Fig. 2 Sequential concatenation

2) When a process P is concatenated with a process Q using an AND-split and an AND-join, we depict the resulting process as P•Q.This type of concatenation is illustrated in Fig. 3.

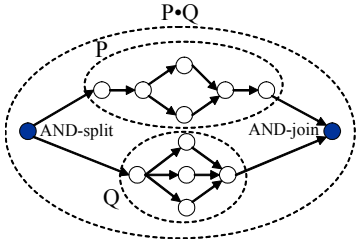

Fig. 3 AND concatenation

3) When a process P is concatenated with a process Q using an OR-split and an OR-join, we depict the resulting process as PoQ. This type of concatenation has the same illustration as the one in Fig. 3, except that the AND-split and the AND-join shown are replaced with an OR-split and an OR-join, respectively.

4) When a process P is concatenated with a process Q using a XOR-split and a XOR-join, we depict the resulting process as P⊕Q. This type of concatenation has also the same illustration as the one in Fig. 3, except that the AND-split and the AND-join shown are replaced with a XOR-split and a XOR-join, respectively.

C. Evaluating the CFC Metric

The nine criteria proposed by Weyuker give a framework to evaluate software metrics' properties using a formal theoretical basis. The properties are intended to evaluate complexity measures on source code metrics. Since there is a strong similarity of source code flowgraphs and processes [1], we will use Weyuker' properties to validate our CFC measure. This widely used criterion will be presented, adapted and applied to processes in the following paragraphs.

Property 1:

There are processes P and Q such that the complexity of P is not equal to the complexity of Q. The property requires that a measure should not produce the same complexity value for every process.

$$(\exists P)(\exists Q)(|P| \neq |Q|).$$

This property is an essential requirement for measures and process measurement. It says that a measure should distinguish between at least two processes. The property stresses that a measure in which all processes are equally complex is not really a measure.

With our measure we can always come up with two processes with two different control-flow complexity values. We can always design a process P which has the same number of split types but with a higher fan-out from those in process Q. As another example, let us take two processes, P and Q, containing only XOR splits. Let us assume that P=Q (the processes are exactly equal). Let us replace the XOR splits of process P with OR splits (for correctness reasons, let us also replace the XOR joins with OR joins). Since $CFC_{XOR\text{-}split}(a)$= fan-out(a) and $CFC_{OR\text{-}split}(a)= 2^{fan\text{-}out(a)}-1$, where a is an activity, then |P|>|Q|. Therefore Property 1 is satisfied.

Property 2:

A measure has to be sufficiently sensitive. A measure is not sensitive enough if it divides all processes into just a few complexity classes

Let c be a nonnegative number. Then there are only finitely many processes for which |P| = c.

Our CFC measure does not follow this property. Therefore, it makes no provision for distinguishing between processes which have a small number of activities (possibly performing very little computation) and those which have a large number of activities (possibly performing substantial amount of computation), provided that they have the same decision structure. The influence of the number of activities is captured by the activity complexity metric.

Property 3:

We have processes which have different degrees of perceived complexity, but map into the same complexity measurement value.

There are distinct processes P and Q such that, |P|=|Q|.

A measure that assigns a distinct value to every process is not much of a measure. It would go against the principle of measurements which requires that the number of objects that can be measured be greater than range of the values of the measure.

Our measure clearly satisfies this property. Let us take two processes, P and Q. Let us assume that P has an AND-split at activity a with a fan-out(a) of two. Let us construct process Q exactly in the same way as process P, but with a fan-out(a) of four at the AND-split activity a. Since $CFC_{AND\text{-}split}(a)= 1$, the complexity of P is equal to the complexity of Q, i.e. |P|=|Q|, but the processes are distinct.

Property 4:

There exist processes P and Q such that P is equivalent to Q but the complexity of P is not equal to the complexity of Q.

$$(\exists P)(\exists Q)(P \equiv Q \text{ and } |P| \neq |Q|).$$

Even though two processes may have the same functionality, it is the details of the design that determine the process's complexity. There are different process designs for the same functionality. For example let us take a business process that makes the backup of a file system composed of four activities that save files at different locations. Two different designs (processes P and Q) with the same functionality of the business process can be constructed. Process P, carries out the four activities sequentially, while process Q uses an AND-split and an AND-join to reduce the time it takes to complete a file system backup. As a result, |P|=0 and |Q| =1, i.e. |P|≠|Q|. Therefore this property is satisfied by our metric.

Property 5:

For any processes P and Q, the complexity of P*Q, $* \in \{-, \circ, \bullet, \oplus\}$, is greater than or equal to the original complexity of P (weak positivity).

Case 1 (-):

$$(\forall P)(\forall Q)(|P - Q| \geq |P|)$$

For the concatenation operation '–', the weak positivity holds. For any two processes P and Q, |P-Q|=|P|+|Q|, thus |P-Q| ≥ |P|.

Case 2 (o):

$$(\forall P)(\forall Q)(|P \circ Q| > |P|)$$

For the concatenation operation 'o', the weak positivity holds. For any two processes P and Q, |PoQ|=|P|+|Q|+2^2-1, thus |PoQ| ≥ |P|. Furthermore, for the concatenation operation 'o' the positivity also holds since |PoQ| > |P|.

Case 3 (●):

$$(\forall P)(\forall Q)(|P \bullet Q| > |P|)$$

For the concatenation operation '●', the weak positivity holds. For any two processes P and Q, |P●Q|=|P|+|Q|+1, thus |P●Q| ≥ |P|. Furthermore, for the concatenation operation '●' the positivity also holds since |P●Q| > |P|.

Case 4 ($\oplus$):

$$(\forall P)(\forall Q)(|P \oplus Q| > |P|)$$

For the concatenation operation ' $\oplus$ ', the weak positivity holds. For any two processes P and Q, |P $\oplus$ Q|=|P|+|Q|+2, thus |P $\oplus$ Q| ≥ |P|. Furthermore, for the concatenation operation ' $\oplus$ ' the positivity also holds since |P $\oplus$ Q| > |P|.

Property 6:

There exist processes P, Q, and R, such that |P|=|Q| and |P*R|≠|Q*R|, where $* \in \{-, \circ, \bullet, \oplus\}$.

$$(\exists P)(\exists Q)(\exists R)\begin{pmatrix} |P| = |Q| \text{ and } |P * R| \neq |Q * R| \\ \text{and } * \in \{-, \circ, \bullet, \oplus\} \end{pmatrix}$$

As with property 5, this property has four distinct cases.

Case 1 (-): |P-R|=|P|+|R| and |Q-R|=|Q|+|R|, since |P|=|Q|, it holds that |P-R|= |Q|+|R|, thus |P-R| = |Q-R|.

Case 2 (o): |PoR|=|P|+|R|+2^2-1 and |QoR|=|Q|+|R|+2^2-1, since |P|=|Q|, it holds that |PoR|= |Q|+|R|+2^2-1, thus |PoR| = |QoR|.

Case 3 (●): |P●R|=|P|+|R|+1 and |Q●R|=|Q|+|R|+1, since |P|=|Q|, it holds that |P●R|= |Q|+|R|+1, thus |P●R| = |Q●R|.

Case 4 ($\oplus$): |P $\oplus$ R|=|P|+|R|+2 and |Q $\oplus$ R|=|Q|+|R|+2, since |P|=|Q|, it holds that |P $\oplus$ R|= |Q|+|R|+2, thus |P $\oplus$ R| = |Q $\oplus$ R|.

As a result, it is clear that our measurement does not follow Weyuker's property 6 in any of the cases presented.

Property 7:

There are processes P and Q such that Q is formed by permuting the order of the activities of P and |P| is not equal to |Q|.

$(\exists P)(\exists Q)$ *If Q is formed by permuting the order of the activities of P, then* $|P| \neq |Q|$.

This property requires that permutation of elements within a process change the metric value. The intent is to ensure that the possibility exists for metric values to change due to permutation of process activities.

Let us assume that we have a process P which contains an AND-split and an OR-split for the activities a_1 and a_2, respectively. Each split has a different fan-out. Activity a_1 has a fan-out of two, while activity a_2 has a fan-out of three. Therefore,

$$|P| = CFC_{AND-split}(a_1) + CFC_{OR-split}(a_2)$$
$$= 1 + 2^3 - 1 = 8$$

Let us assume that Q is a permutation of the activities of process P. More precisely, the activities a_1 and a_2 are exchanged. As a result, activity a_1 has now a fan-out of three, while activity a_2 has a fan-out of two. The complexity of Q becomes,

$$|Q| = CFC_{AND-split}(a_2) + CFC_{OR-split}(a_1)$$
$$= 1+2^2 - 1 = 4$$

Since $|P| \neq |Q|$ (i.e. $8 \neq 4$), it happens that our measurement follows this property.

Property 8:

This property states that uniformly changing activity names should not affect a process complexity.

If P is a renaming of Q, then |P| = |Q|.

This property requires that when the name of the activities or processes changes, the metric should remain unchanged. As the metric being considered in this research does not depend on the name of activities or processes, it satisfies this property.

Property 9:

The complexity of a process formed by concatenating two processes can be greater than the sum of their individual complexities (wholeness property). This property states that the whole must be at least as great as the sum of the parts. The idea behind wholeness is that the whole is more complex than the sum of its components.

$$(\exists P)(\exists Q)(|P * Q| > |P| + |Q|, \text{ and } * \in \{-, \circ, \bullet, \oplus\}),$$

This property states that, at least in some cases, the complexity of a process formed by concatenating two

processes is greater than the sum of their complexities. This reflects the fact that there may be interactions between the concatenated processes.

As with previous properties, this property has four distinct cases.

Case 1 (-): |P-Q|=|P|+|Q|, thus |P-Q| ≥ |P|+|Q|.

Case 2 (o): $|PoQ|=|P|+|Q|+2^2-1$, thus |PoQ| > |P|+|Q|.

Case 3 (●): |P●Q|=|P|+|Q|+1, thus |P●Q| > |P|+|Q|.

Case 4 ($\oplus$): $|P \oplus Q|=|P|+|Q|+2$, thus $|P \oplus Q| > |P|+|Q|$.

As a result, our measurement follows property 9 for case 2, 3, and 4. Case 1 follows a variation of the wholeness property, called the weak wholeness property.

$$(\exists P)(\exists Q)(|P-Q| \geq |P|+|Q|)$$

D. Dealing with Process Loops

Our complexity metric is able to cope with the modeling of loops. When a transition "goes back" to a previous activity, a XOR split as to be place on the activity that will decide if the loop will be taken or not. As presented earlier, our analysis of Weyuker's properties accounts for the existence of XOR splits in a process.

V. Related Work

While a significant amount of research on the complexity of software programs has been done in the area of software engineering, the work found in the literature on complexity analysis for business processes, workflows, and processes in general is almost inexistent.

Research in software engineering has produced various measurements for software. Among others are lines-of-code, the Halstead's measure [10], McCabe's measure [4], the and the COCOMO model [11]. There is a vast literature on software metrics which represents the result from the measurement of the development, operation and maintenance of software in order to supply meaningful and timely management information.

Misra and Misra [12] have evaluated cognitive complexity measure in terms of Weyuker properties and has found that most of Weyuker properties have been satisfied by the cognitive weight software complexity measure and established the cognitive complexity as a well structured one.

In [13] the authors attempt to formalize some properties which any reasonable control-flow complexity measure must satisfy. Their approach is directed to large software programs which are often built by sequencing and nesting of simpler constructs, the authors explore how control-flow complexity measures behave under such compositions.

Please note that these two last fields of research have been carried out in the context of software engineering and not process management.

VI. Conclusions

Most of the work done so far in the business process field has been tool-oriented and technological in nature; the main goal has been during years the definition and development of WfMS including models, modeling languages, correctness analysis, and execution environments. Recently, a new field of research for processes has emerged. This new field – termed process measurement – presents a set of approaches to the quantification of specific properties of processes, such as their complexity.

The process control-flow complexity (CFC) metric is a design-time metric that can be used to evaluate the difficulty of producing business process, Web process, and workflow designs before an actual implementation exist. When process control-flow complexity analysis becomes part of the process development cycle, it has a considerable influence in the design phase, leading to less complex processes.

To increase the confidence, acceptance, and use of the CFC measure we have carried out a serious validation procedure using Weyuker's nine properties. These properties give a formal analytical approach to classify our measure. Since our CFC measure happens to fully satisfy seven of the Weyuker's nine properties and partially satisfies one property it can be considered to have passed a significant part of the theoretically validation process. Therefore, it can be categorized as a good, structured, and comprehensive one.

References

[1] Cardoso, J., Evaluating Workflows and Web Process Complexity, in Workflow Handbook 2005, L. Fischer, Editor. 2005, Future Strategies Inc.: Lighthouse Point, FL, USA. p. 284-290.
[2] Weyuker, E.J., Evaluating software complexity measures. IEEE Transactions on Software Eng., 1988. 14(9): p. 1357-1365.
[3] Cardoso, J. About the Data-Flow Complexity of Web Processes. in 6th International Workshop on Business Process Modeling, Development, and Support: Business Processes and Support Systems: Design for Flexibility. 2005. Porto, Portugal.
[4] McCabe, T., A Complexity Measure. IEEE Transactions of Software Engineering, 1976. SE-2(4): p. 308-320.
[5] Kitchenham, B., S.L. Pfleeger, and N. Fenton, Toward a Framework for Measurement Validation. IEEE Transactions of Software Engineering,, 1996. 21(12): p. 929-944.
[6] Fenton, N., Software Measurement: A Necessary Scientific Basis. IEEE Transactions on Software Engineering, 1994. 20(3).
[7] Morasca, S., et al., Comments on "Towards a Framework for Software Measurement Validation". IEEE Transactions on Software Engineering, 1997. 23(3): p. 187-188.
[8] Zuse, H., A Framework of Software Measurement. 1997, Berlin: Walter de Gruyter Inc.
[9] Aalst, W.M.P.v.d., The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 1998. 8(1): p. 21-66.
[10] Halstead, M.H., Elements of Software Science, Operating, and Programming Systems Series. Vol. 7. 1977, New York, NY: Elsevier.
[11] Boehm, B., Software Engineering Economics. 1981: Prentice Hall.
[12] Misra, S. and A.K. Misra. Evaluating Cognitive Complexity Measure with Weyuker Properties. in Third IEEE International Conference on Cognitive Informatics (ICCI'04). 2004. Victoria, Canada.
[13] Lakshmanan, K.B., S. Jayaprakash, and P.K. Sinha, Properties of Control-Flow Complexity Measures. IEEE Transactions on Software Engineering archive, 1991. 17(12): p. 1289 - 1295.