Application of Metric Measures: From Conventional Software to Expert Systems

Zhisong Chen and Ching Y. Suen
Centre for Pattern Recognition and Machine Intelligence
Concordia University
1455 de Maisonneuve Blvd. West
Montréal, Québec H3G 1M8
Canada
czs@cenparmi.concordia.ca

Abstract

The importance of metric measures has been recognized in almost every scientific and engineering discipline, including software engineering where much progress in this application has been made. In this paper, metric measures are studied for their application to expert systems, for which little work has been done. The characteristics and organization of metric measures are discussed and presented. Due to the analogy between conventional software and expert system, a comparative study is also conducted. A new expert system metric is proposed. Test results indicate that this new measure compares favorably with others.

Introduction

While much progress has been made in quantitative metric measurements of conventional software (CS), little work has been done in this area for expert systems (ESs). Measurement plays a crucial role in the development of every discipline and the metrics used are essential in each scientific and engineering field. Metrics can also play a role in the evolution of knowledge engineering from an art to an engineering and scientific discipline. Expert system is one of the most important applications of knowledge engineering and has been widely used in a variety of domains. However, the lack of formal measurements in ESs has prevented our control over them and increased the risk of applying them. In this paper, we will examine some expert system metric measures based on the analysis of conventional software metrics. A new complexity metric – RC has been developed, which has the best performance. Some problems related to such applications will also be discussed.

Measurements

Measurement is usually described as a process or activity in which a symbol or number is assigned to an object in terms of its properties or attributes [Fenton, 1991; Harrison, 1992]. Basically, the measure which reflects such a assignment can be described as the mapping $\mathcal{F}$ from $\mathcal{S}$ to $\mathcal{N}$, where $\mathcal{S}$ represents the set of objects to be measured or some other measures, and $\mathcal{N}$ is the set of objects in a formal system. For example, in the case of software engineering (SE), $\mathcal{S}$ is often the set of source codes or control flows, $\mathcal{F}$ represents the different kinds of mapping functions (measurements) from $\mathcal{S}$ to numerical set $\mathcal{N}$, such as the number of lines of codes, program volume, effort and level [Halstead, 1977], and cyclomatic complexity [McCabe, 1976].

Measures can be roughly classified as *direct measures* and *indirect measures* [Fenton, 1991]. As the names imply, the direct measures (such as number of variables, number of distinct operators and operants) could be obtained by applying some measuring mechanisms to the objects directly, while the indirect measures (such as product complexity, maintainability, testability) involve at least one of the other measures obtained previously. There exist various mappings for direct and indirect measures, which reflect the different measuring strategies based on different intentions and requirements. In general, metrics can be regarded as the concrete representations of the individual measures, which can be identified by the associated metric names. They could be applied to quantitatively characterize the different attributes of products (such as program size, number of predicates, number of execution paths) and the development process (such as time, programming correctness). Metrics can also be divided into two classes, *static and dynamic metrics.* Static metrics focus on the static characteristics of the products or processes while dynamic metrics concern the dynamic behavior. In this paper, our study is focussed on product-related and static metrics only.

ES versus CS

ESs differ from conventional software (CS) in application domain, development process, system structure,

ESs	CS
rules	conventional statements
rule dependency	data reference
rulebase	source code
inference engine	source code
forward/backward chaining	sequential execution
reasoning	calculation
shells	programming languages
complex application domain	clearly defined domain
expert-level performance	data processing

Table 1: Comparisons between ESs and CS

execution order, and other aspects. However, they also contain some conventional portions, such as the arithmetic operations, function or procedure calls and the employed variables. Table 1 gives a comparison between rule-based ESs and CS on some of these characteristics .

As a result, the development of measuring techniques and methods for ESs falls into two approaches:

1) Develop new approaches to measure the attributes that are inherent or particular to ESs

2) Adapt or extend the established techniques and methods in SE to measure the common features that exist both in conventional software and ESs.

Using the above as a general guideline, the design of the expert system metrics will entail an analysis of the problems and the proper implementation of the measurements.

Expert System Metrics

Due to the analogy between ESs and CS, let's first briefly review the measurements used in SE.

Conventional Software Metrics

In SE, much effort has already been spent in quantifying the different quality-related attributes of software products, where the measured objects $\mathcal{S}$ are the set of source codes and control flows, and the mapping functions $\mathcal{F}$ are different proposed measuring approaches. Considerable progress has been made in the past years. Metrics have been suggested and used as an effective way to predict and measure the characteristics of software products in their entire life-cycle, and the Metrics Guided Methodology has been proposed by Ramamoorthy [Ramamoorthy *et al.*, 1985]. The use of metrics to evaluate the complexity of software products is quite common in the maintenance phase. Actually, many empirical studies have already been conducted using such techniques [Li and Cheung, 1987; Elshoff, 1984; Harrison *et al.*, 1982].

Some of the typical metrics used in SE are: (1) volume-based "software science" [Halstead, 1977], in which the measurement of the program volume, effort and level is based on counting the operators and operands; (2) control flow-based cyclomatic complexity metric [McCabe, 1976], which is defined as the number of decision points plus one to indicate the number of basic paths in the control flow; (3) digraph-based scope and ratio metrics [Harrison and Magel, 1981], which reflect the "influence" upon each selected node by other nodes and "relatedness" among the nodes; and (4) hybrid metric [Li and Cheung, 1987], which combines the scope idea with "software science" to avoid the shortcomings of single-factor measurement. Empirical studies support the notion that hybrid complexity measurements, which take into account different factors of the software product, could give more reliable and valid results for general cases [Li and Cheung, 1987; Tai, 1984; Harrison *et al.*, 1982].

Layer Structure of Metrics

In section 2, direct and indirect measures are described. Software Metrics (conventional software metrics and expert system metrics) for these measures can be further refined. Given S represents the set of the objects, $\mathcal{N}$ is the numerical set, then the layer structure is proposed as follows, which aims to organize them in a more illustrative way:

1. Layer S_1: the set of direct metric measures which involve only one attribute (such as number of lines of codes) of the objects to be measured, that is:

$$S_1 = \{F \mid F : S_0 \longrightarrow \mathcal{N}\}$$

where, S_0 is the set of the attributes of S.

2. Layer S_n ($n \geq 2$): the set of metric measures which involves at least one metric from layer $n-1$, that is:

$$S_n = \{F|F : \underbrace{S_1 \times S_1 \cdots \times S_1}_{k_1} \times \cdots \times \underbrace{S_{n-1} \times S_{n-1} \cdots S_{n-1}}_{k_{n-1}} \longrightarrow \mathcal{N}\}$$

where, $k_{n-1} \neq 0$.

With the defined structure above, some metrics used in SE can be categorized as shown in Table 2, which is one possible classification of the conventional software metrics. Similar metric layers can also be developed for expert system metrics.

It can be seen that the quality metrics such as maintainability usually belong to the higher layers, whereas the first layer metrics consist of the number of lines of codes, number of operators, number of operands, *etc*, which form the fundamental measures. The most important benefit of such layer organization is that some concepts and techniques of *information hiding* can be applied to design and evaluate the different layer metrics.

Product Quality	Layer n
....	......
Maintainability	Layer 4
Understandability, Testability, Modifiability Reliability, Efficiency, Usability	Layer 3
Complexity, Effort, Cost, Volume, Accuracy, Correctness	Layer 2
Number of lines, number of variables, number of operands, number of operators	Layer 1

Table 2: Layer Structure of Metrics

Characteristics of Metric Measurement

The formulation of metric measurements is affected by the following aspects:

- Objectives. The reason for defining metrics is to get some feedbacks from the measures, hence to control the system. The metrics without useful feedbacks are meaningless. However, having different objectives, some of them may be contradictory to each others and require different types of feedbacks, may lead to different measurement approaches, even for the same metric. For example, there exist many complexity metrics for the CS; McCabe's control flow-based cyclomatic complexity [McCabe, 1976] aimed at identifying the basic control paths in the program for "walking through", the data flow and massive computation are not the concern. On the other hand, Henry *et al* defined the complexity metric [Henry and Kafura, 1981] in terms of information flow to reflect the occurrence of changes, which was declared to be important. Based on the claim that a larger program has a higher maintenance cost [Harrison *et al.*, 1982], the complexity measure could simply be defined as the number of source lines of codes if the only concern is maintenance cost.
- Scale. There exist four kind of scales for metric measurements: nominal scales, ordinal scales, interval scales and ratio scales [Harrison *et al.*, 1982]. It is more practical to have the ordinal scale for the higher layer (above first layer) metrics, which provides just the comparisons between different objects. For example, when comparing the understandability of two programs A and B, it is feasible to give the result that A is more difficult than B. The reason is that higher layer metric measures depend on many factors including human aspects about which little is known. It is hard, sometimes impossible, to exactly measure these factors. However, relative comparisons can be made based on some existing attributes of the objects. But this kinds of result has only limited use because some further operations cannot be performed on them [Zuse and Bollmann, 1989].
- Operations. Owing to the different natures of various metrics, it is not always meaningful to have some operations on these metrics, such as the addition of different metrics and statistical calculations. For the former, it should be done in an admissible way, at least it has an intuitive persuasion. Some metrics may have different multi-values depending on different points of views, so simply adding several measures together to obtain a single value for such metrics cannot reflect their real values adequately. For the statistical calculations, even on the same measure, we should still be cautious in applying them, because of restrictions imposed on them [Zuse and Bollmann, 1989].
- Application. Metric measures only give quantitative values for some properties of the objects. They help to analyze the problems but cannot solve them. To further apply such metric measures may involve many other aspects such as the characteristics (object features, maintainer's experience, available tools, analysis of the relationship between the metric measures and the problems, *etc*) of the environment in which such metrics are used and more work needs to be done.
- Precision vs. Expense. There is a trade-off between the precision of the measure and the expense required to obtain such a measure. It is obvious that the more precise the metrics are, the more expensive it usually is to get such metrics, since more information and processing are involved. For higher layer metrics, they are defined in terms of several other measures, so the measure of such higher layer metrics will involve some low layer metric measures as well. If we consider too many factors, hence too many lower layer metrics, for a higher layer metric, then the cost will be high. On the other hand, if too few factors are considered, it is too coarse to be meaningful and useful.

So, even metrics are useful, there exist some conditions for them to be effective. Decision on the above factors must be made before the actual measurement

Metrics for ESs

Having seen many metrics for CS, one wonders whether they can also be applied to ESs? Harrison concluded three steps for the formulation of conventional software metrics: (1) definitions of the properties; (2) method for weighting the interesting properties; (3) method for summarizing the resulting information [Harrison, 1992]. A similar but different body of metric measurement can also be developed for the formal evaluation of ESs [Suen *et al.*, 1990]. Traditional software measuring techniques may be applied to the inference engine [Barrett, 1990; Kiper, 1989], because of its similarities in function to conventional software; that is, both are the implementations of some algorithms for the purpose of control. But these measures cannot be applied to the knowledge-base directly, because of its different nature. For example, the rulebase which represents the knowledge-base of an expert system is made up of many individual rules, each of which expressed as $if < antecedents - list > then < consequent >$ acts like a decision node which is quite different from a conventional statement. There is a strong interrelation among the rules called *dependency*, which is the matching between the antecedents and consequents of different rules. These dependencies decide and constitute the dynamic search paths, sometimes the number of paths may be enormous. In a conventional program, the relationship among the statements is mainly built by the data (variables) references using logical relationship, which usually do not produce a large number of control paths. Also, in ESs, the execution order of rules will not be decided by the physical order of rules listed in the rulebase. It is mainly decided by the input data (facts) and the inference engine or metarules. However, the order in the conventional programs is only determined by the input data and the statements listing. The control and data are mixed in the source code, whereas in ESs, there is a separation between the rulebase and the inference engine.

Already, some work on measuring ESs has begun. For example, Plant presented a rigorous methodology that used a set of formal specifications toward the implementation of knowledge-based systems [Plant, 1991b]. The quality improvement through this methodology was shown by the effects of this methodology on some quality factors such as correctness, reliability, efficiency, integrity, testability, usability, maintainability, which can be evaluated by the metrics [Plant, 1991a]. Kiper attempted to extend McCabe's cyclomatic metric to measure the basic search paths contained in rulebases [Kiper, 1989]. Buchanan suggested the complexity metric of a solution space to be the measurement of the average width and depth of the rulebase [Buchanan, 1987]. Mehrotra defined and used a "distance metric" to group the rulebase so as to increase the expert system's comprehensibility, maintainability and reliability [Mehrotra, 1991]; Preece suggested that attention should also be paid to the differences in data-to-rule ratio [Preece, 1990]. And in the UK, the Gateway project which aims at developing a coherent set of metrics for knowledge-base, is in progress [Behrendt *et al.*, 1991].

To date, several metrics have been proposed to measure rulebases, such as:

- object volume [Kaisler, 1986],
- number of rules [Suen *et al.*, 1990],
- number of antecedents and consequents in a rule [Kaisler, 1986],
- breadth of the knowledge-base [Suen *et al.*, 1990],
- depth of the search space [Suen *et al.*, 1990],
- complexity of individual rules [Miller, 1990],
- vocabulary of the knowledge-base [Buchanan, 1987].

The problem lies in the validity of these measurements; some of them may be highly inter-related and measure the same factors of the rulebase. Also we can see that all the above metrics lie in the first and second layers, this means that the current research is still at its initial state — the study of the lower layer metrics and more research has to be done in this area.

Rulebase Complexity

The complexity of rule-based ESs or rulebases plays an important role in their evaluation. Since little effort has been devoted to measuring it, we will study this subject and propose a new rulebase complexity measure – RC metric which will be described in the following sections.

Model of Complexity

Now the question is to recognize the essence of the complexity of rulebases. First, in order to avoid the confusion which may occur due to the various implications for the complexity, we present the following definition for the rulebase complexity metric:

> *"The measure of the degree of the difficulty in understanding and managing the knowledge structure*[1] *represented by the rulebase."*

The above indicates that our purpose in the measurement mainly concerns with understanding and managing the rulebase. It is related to maintainability which is defined over the complexity and some other factors. The objects $\mathcal{S}$ to be measured is the set of rulebases, and, like the conventional complexity measure, the measure function $\mathcal{F}$ can be constructed at the second layer.

Complexity measure may be affected by (a) *external component* which produces the complexity and (b) *internal component* which reflects the complexity of objects in certain forms to be measured. For rulebases, this form appears to be the rulebase size and the dependency among the rules. The *external component* relates to the experts who provide the expertise and knowledge, the knowledge engineers who acquire and organize the knowledge and some other environment factors. While it seems that the above *external component* is situation-dependent and difficult to measure, actually our focus is on the measurement of the *internal component*.

Formulation of RC Metric

As discussed above, "dependency" and "size" seem to be two major components contributing to the formulation of the rulebase complexity. By intuition, it also follows that a person attempting to understand a rulebase will be mentally affected by the size of the rulebase and all the search paths formed by the rule de-

[1] Here, we are mainly concerned with the difficulty caused by the knowledge-base's syntactic structure, the semantic influence is expected to be studied in the near future.

pendency. Based on this, our new complexity metric RC is defined in terms of the following direct measures.

- Content π
- Connectivity ρ
- Size v

The first component is designed to measure the contents of the rules, which form the different match patterns among the rules and give the "potential chaining"; the second part attempts to evaluate the "relatedness" among the rules, which constitutes the rule chains and search paths. The last factor reflects the effect of the size on the complexity.

Hence, by summarizing the above the three direct measures, the quantitative RC metric on the rulebase R is defined as:

$$RC(R) = [W_1*(1.0-\frac{k_1}{k_1+\sum_{i=1}^{|P|}\rho(r_i)})+W_2*(1.0-\frac{k_2}{k_2+\pi}) + W_3*(1.0-\frac{1.0+k_3}{k_3+v})]^n \qquad (1)$$

where

$$\begin{aligned} P &= \text{Set of connected rules.} \\ & r_i \in P \\ k_1,\ k_2,\ k_3 &= \textit{scale factors} \\ n &= \textit{constant} \\ W_1,\ W_2,\ W_3 &= \textit{weights} \end{aligned}$$

The weights have been set in such a way that $W_1 + W_2 + W_3 = 1$ and they indicate the different contributions of the three components relating to ρ, π, and v.

The rationales for expressing RC as formula (1) are summarized as follows:

- The complexity may be of a higher order than linear in terms of ρ, π, and v.
- As ρ, π and v increase, the RC measurement also increases.
- Reasonable results can be obtained when formula (1) is applied to some sample rulebases.
- Empirical studies and experiments may be used to deduce this kind of measurements.

Further analysis of formula 1 indicates that the RC metric has some interesting properties[2] which are listed below:

Suppose R represents a rulebase.

(a) $0 \leq RC(R) < 1$.

(b) Adding a new node (rule) or edge (connection) to R without reducing the search paths will increase RC(R) except that the added node is isolated.

(c) For any subset $\Theta \subseteq R$, $RC(\Theta) \leq RC(R)$. That is, the complexity of each sub-rulebase is less than or equal to that of the global rulebase.

(d) $RC(R) = 0$ denotes that R contains only isolated nodes in which each node acts as a separate subgraph.

(e) $0.0254 \leq RC(R) < 0.667$ if R forms only one search path.

(f) $RC(R) > 0.0340$ if there are more than two paths in R.

[2]It can be derived from formula 1 based on: $W_1 = W_2 = W_3 = \frac{1}{3}$, $k_1 = k_2 = 100, k_3 = 50, n = 1$.

Rulebases	Purposes
R_1	Animal Identification
R_2	Trip Route Advisor
R_3	Tree Classification
R_4	Undergraduate Course Advisor
R_5	Software Selection
R_6	Tape Product Selection
R_7	Automobile Purchase Advisor
R_8	Hospital Management
R_9	Software Preliminary Evaluation
R_{10}	Selection of Law Universities
R_{11}	Steel Wire Rope Diagnostics
R_{12}	Automobile Diagnostics
R_{13}	Children Health State Advisor
R_{14}	Allergy Self-Help
R_{15}	Career Choices
R_{16}	Computer Purchasing
R_{17}	Diagnosis of Abdominal Pain
R_{18}	On-Line Automobile Selection
R_{19}	Rock Identification
R_{20}	Car Diagnostics
R_{21}	Phone Line Fault Detection

Table 3: Testing Data Used in the Empirical Studies

Some Results

The 21 rulebases presented in Table 3 are used as the testing data to compute three metrics: *number of rules (NR), Buchanan's solution space complexity (BSSC) and the new RC metric.* The measuring results are shown in Figure 4.

A preliminary analysis of the results indicates the following:

- RC measure has quite good correlation with NR and BSSC which have been considered as indicators of rulebase complexity. This correlation is an inherent property of RC measurement, determined by our consideration in its formulation. Therefore, it shows that RC can indeed act as an indicator of the rulebase complexity. Imperfect correlation, which makes

Rulebases	NR	$BSSC$	RC
R_1	10	2.59	0.300
R_2	12	3.0	0.441
R_3	14	3.4	0.276
R_4	16	4.3	0.338
R_5	16	4.6	0.402
R_6	23	9.8	0.352
R_7	25	6.5	0.457
R_8	38	16.2	0.507
R_9	44	15.4	0.494
R_{10}	47	11.0	0.407
R_{11}	53	28.1	0.486
R_{12}	62	34.5	0.570
R_{13}	74	43.7	0.572
R_{14}	83	50.8	0.528
R_{15}	91	79.7	0.663
R_{16}	112	81.4	0.627
R_{17}	114	76.0	0.517
R_{18}	117	88.2	0.701
R_{19}	123	89.7	0.643
R_{20}	127	72.2	0.617
R_{21}	131	90.3	0.654

Table 4: Measuring Results for NR, BSSC and RC Metrics

our measurement more reasonable from the viewpoint of rulebase complexity, can be attributed to the following reasons:

- In some applications, the rulebases, especially those with a large number of rules such as the R_{17} with 114 rules in our testing data, can be degraded into different subsets of rules without any connection to each other. As a result, their complexity is decreased because of this separation, and it may not look as high as the number of rules may indicate. One extreme situation is that all rules in a large rulebase are isolated from each other, that is, each rule forms a separate subset without connecting to the others. Therefore the overall complexity is in fact very low[3], however, simply counting the number of rules without consideration of their internal relations among rules could give the wrong information and lead to contradictory decision. Our measurement reflects this intuitive feature, for example, the RC measurement for the rulebase R_{17} in the testing data is only 0.517 which is lower than those of the rulebases R_{16}, R_{15}, R_{14}, R_{13}, R_{12} which contain less rules. So, from this point of view, *RC* as a complexity indicator appears to be more accurate than simply the number of rules.
- The number of variables contained in the matching patterns in each rule is also an important aspect which should not be ignored when comparing different rulebases. However, BSSC reflects the size of the abstract solution space only; it does not account for the contents of the rulebases. It only concerns with the rule chains formed by the rulebases, and regards the complexities of $P \rightarrow B$ and $P(Y,X) \rightarrow B(X,Y)$ as the same even though the latter case usually implies a more complicated matching. This is one of the reasons that for some rulebases with a high Buchanan's solution space complexity, RC measure is not so high. This indicates that the RC measurement can preform better than the solution space complexity.

[3] In our measurement, we will count it as zero.

Evaluating Expert System Metrics

In order to validate the existing metrics and formulate new metric definitions, it is essential to establish some basic properties against which the measure could be evaluated. Besides the general requirements for the metric measurements such as meaningful, reasonable, reliable and cost-effective, we present the following desired properties for evaluating the metric behavior.

Suppose in the following discussions, the capital letters denote different rulebases, $\theta(\mathcal{R})$ represents the metric θ measured on rulebase $\mathcal{R}$, then we have:

Property 1: $\exists\mathcal{P}\,\exists\mathcal{R}\cdot(\theta(\mathcal{P}) \neq \theta(\mathcal{R}))$.

This property requires the metric to be able to scale and compare the rulebases.

Property 2: $\forall\mathcal{P}\,\forall\mathcal{R}\cdot(\mathcal{P}\subset\mathcal{R}\rightarrow\theta(\mathcal{R}_{\prime}) < \theta(\mathcal{R}))$.

This property means that the measure on a sub-rulebase shall be smaller than that of the rulebase as a whole.

Property 3: For a nonnegative constant c, there are only finite number of rulebases with the measure c.

Property 4: $\exists\mathcal{P}\,\exists\mathcal{R}\cdot(\mathcal{P} /= \mathcal{Q} \wedge \theta(\mathcal{P}) = \theta(\mathcal{R}))$.

Here, the $/=$ sign denotes the unequal dependency structures of different rulebases.

This property indicates that rulebases with different structures may have the same measure.

Property 5: $\exists\mathcal{P}\,\exists\mathcal{R}\cdot(\mathcal{P} <> \mathcal{Q} \wedge \theta(\mathcal{P}) = \theta(\mathcal{R}))$.

The "<>" sign means the unequal functions of different rulebases.

This property indicates that rulebases with different functions may have the same measure.

Property 6: $\exists\mathcal{P}\,\exists\mathcal{R}\cdot(\mathcal{P} \equiv \mathcal{Q} \wedge \theta(\mathcal{P}) \neq \theta(\mathcal{R}))$.

The " $\equiv$" sign means the rulebases with the same function.

This property indicates that rulebases having the same behavior may not have the same measure.

Property 7: $\exists\mathcal{P}\,\exists\mathcal{R}\cdot(\theta(\mathcal{P}) = \theta(\mathcal{R}) \wedge \exists\mathcal{Q}\cdot(\theta(\mathcal{P}\cup\mathcal{Q}) \neq \theta(\mathcal{R}\cup\mathcal{Q})))$

Property Number	1	2	3	4	5	6	7	8	9	10	11
RC	1	1	1	1	1	1	1	1	1	1	1
NR	1	1	1	1	1	1	0	1	0	0	0
BSSC	1	0	0	1	1	1	1	1	1	1	0
KC	1	0	0	1	1	1	1	1	1	1	0
ADSS	1	0	0	1	1	1	1	1	1	1	0
ABSS	1	0	0	1	1	1	1	1	1	1	0
NAC	1	1	1	1	1	1	0	1	0	0	1

Table 5: Evaluation of Different Metrics

This property means that the conjunction of rulebases may have different effects.

Property 8: If $\mathcal{P}$ is a renaming of $\mathcal{R}$, that is, if there exists a sequence $\mathcal{R}= P_1, P_2, \ldots\ldots, P_n = \mathcal{P}$, P_i is obtained by replacing all the instances of an identifier x in P_{i-1} by y where y doesn't appear in $\mathcal{P}_{i-1}$, then for the metric θ, which is based on the syntactic structures of rulebases, $\theta(\mathcal{P}) = \theta(\mathcal{R})$.

This property points out that in general renaming will not change the measure.

Property 9: $\exists\mathcal{P}\ \exists\mathcal{R} \cdot (\theta(\mathcal{P}) + \theta(\mathcal{R}) > \theta(\mathcal{P} \cup \mathcal{R}))$

This property asserts that in some cases, the conjunction of rulebases decreases the measure.

Property 10: $\exists\mathcal{P}\ \exists\mathcal{R} \cdot (\theta(\mathcal{P}) + \theta(\mathcal{R}) < \theta(\mathcal{P} \cup \mathcal{R}))$

This property asserts that in some cases, the conjunction of rulebases increases the measures.

Property 11: $\exists\mathcal{P}\ \exists\mathcal{R} \cdot (\mathcal{P} \Leftrightarrow \mathcal{R} \wedge \theta(\mathcal{P}) \neq \theta(\mathcal{R}))$

The "$\Leftrightarrow$" sign means the equal dependency graphs of the rulebases.

This property indicates that for some rulebases, even they have the same dependency structures, the types of dependencies contained in them may be different. So the metric measures for them should be different, i.e., the metric measures should be sensitive to these differences.

Based on the properties introduced above, we have compared several metrics: *our proposed complexity metric (RC), Buchanan's solution space complexity measure (BSSC), number of rules (NR), Kiper's complexity measure (KC), average depth of search space (ADSS), average breadth of search space (ABSS), and number of antecedents and consequents (NAC).* The result is listed in Table 5, where the " 1 " sign denotes the metric with the property while the " 0 " sign indicates the metric without the property.

From the above, we can see that the performances of the four metrics: BSSC, KC, ADSS and ABSS are the same, this is because they were all based on the abstract search space and the search paths in the space. So, they, as the measures, have the same effect. Also, NR and NAC metrics perform quit similarly in the measurements. They are the measures of the size. All

RC	Good Performance
BSSC, KC, ADSS, ABSS	Moderate Performance
NR, NAC	Weak Performance

Table 6: Some Metric Groups According to Their Performance

the above measures lack the insight into the rule interrelation[4]. This is why the *RC* metric performs best in all these measures. It takes into account the space, the size and the matching patterns, and it proves again that the hybrid metric usually gives more reliable and adequate results. According to the performance of these metrics, they can be ranked as shown in Table 6.

Summary

We have studied and discussed several problems concerning metric measurements on expert systems, which are still in their infant state. Several proposed metrics have been examined. As a concrete example, a new complexity measure — RC metric is proposed, which shows some promise.

So far, our metric study is still focusing on the low layer (less than 3) metrics. Building a system of metrics ranging from the first layer to the top one is the goal of our study. We believe this kind of study and research is expected to contribute to the assurance of the qualities of the products developed in the knowledge engineering environment.

Acknowledgment

This research was supported by Bell Canada. The encouragement of Messrs. J.N. Drouin and F. Coallier is much appreciated.

References

Barrett, Britt W. 1990. A software quality specification methodology for knowledge-based systems. In Culbert, Chris, editor 1990, *AAAI-90 Workshop on Knowledge Based Systems Verification, Validation and Testing.* AAAI. Unpublished Workshop Notes.

Behrendt, W.; Lambert, S. C.; Ringland, G. A.; Hughes, P.; and Poulter, K. 1991. Gateway: Metrics for knowledge based systems. In Liebowitz, Jay, editor 1991, *Proceedings of the World Congress on Expert Systems*, volume 2, New York. Pergamon Press. 1056–1067.

Buchanan, Bruce G. 1987. Artificial intelligence as an experimental science. Technical Report KSL 87-03, Knowledge Systems Laboratory, Stanford University, Stanford, CA.

[4]This is also the reason why *the number of antecedents and consequents* performs a little better than *the number of rules.*

Elshoff, James L. 1984. Characteristics of program complexity measurement. In *Proc. of the Int. Conference on Software Engineering.* 288–293.

Fenton, Norman E. 1991. *Software Metrics: A Rigorous Approach.* Chapman & Hall, 2-6 Boundary Row London SE1 8HN.

Halstead, M.H. 1977. *Elements of Software Science.* North-Holland, New York.

Harrison, Warren and Magel, Kenneth 1981. A graph-theoretic complexity measure. In *ACM Computer Science Conf.*, St. Louis.

Harrison, Warren; Magel, Kenneth; and Raymond Kluczny, Arlan Dekoch 1982. Applying software complexity metrics to program maintenance. *Computer* 15:65–79.

Harrison, Warren 1992. Software measurement and metrics. *Encyclopedia of Computer Science and Technology* 26(Supplement 11):363–372.

Henry, Sallie and Kafura, Dennis 1981. Software structure metrics based on information flow. *IEEE Trans. on Software Engineering* SE-7(5):510–518.

Kaisler, Stephen H. 1986. Expert system metrics. In *Proc. 1986 IEEE International Conference on Systems, Man, and Cybernetics*, volume 1. IEEE. 114–120.

Kiper, James D. 1989. Structural testing of rule-based expert systems. In *IJCAI-89 Workshop on Verification, Validation and Testing of Knowledge-Based Systems.* IJCAI.

Li, H. F. and Cheung, W.K. 1987. An empirical study of software metrics. *IEEE Trans. on Software Engineering* SE-13(6):697–708.

McCabe, Thomas J. 1976. A complexity measurement. *IEEE Trans. on Software Engineering* SE-2(4):308–320.

Mehrotra, Mala 1991. Rule grouping: A software engineering approach towards verification of expert system. NASA Contract Report 4372, Vigyan Inc., NASA Langley, Hampton VA.

Miller, Lance A. 1990. Dynamic testing of knowledge bases using the heuristic testing approach. *Expert Systems with Applications (US)* 1(3):249–269.

Plant, R. T. 1991a. Factors in software quality for knowledge-based systems. *Information and Software Technology (UK)* 33(7):527–536.

Plant, R. T. 1991b. Rigorous approach to the development of knowledge-based systems. *Knowledge Based Systems* 4(4):186–196.

Preece, Alun D. 1990. The role of specifications in expert system evaluation. In Culbert, Chris, editor 1990, *AAAI-90 Workshop on Knowledge Based Systems Verification, Validation and Testing.* AAAI. Unpublished Workshop Notes.

Ramamoorthy, C. V.; Tsai, Wei-Tek; Yamaura, Tsuneo; and Blide, Anupam 1985. Metrics guided methodology. In *Proc. of IEEE Computer Society's Ninth International Computer Software & Application Conference*, Chicago. 111–120.

Suen, Ching Y.; Grogono, Peter D.; Shinghal, Rajjan; and Coallier, François 1990. Verifying, validating, and measuring the performance of expert systems. *Expert Systems with Applications (US)* 1(2):93–102.

Tai, Kui-Ching 1984. A program complexity metric based on data flow information in control graphs. In *Processing of 7th International Conference on Software Engineering*, Orlando. 239–248.

Zuse, Horst and Bollmann, Peter 1989. Software metrics: Using measurement theory to describe the properties and scales of static software complexity metrics. *SIGPLAN Notices* 24(8):510–518.