A VARIANT OF THE RESIDUAL COMPLEXITY METRIC

By

ZAIN SAIFULLAH

Sarjana Matematika

Institut Teknologi Bandung

Bandung, Indonesia

1986

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July 1995

OKLAHOMA STATE UNIVERSITY

A VARIANT OF THE RESIDUAL

COMPLEXITY METRIC

Thesis Approved:

Mansur Samadzadeh

Thesis Adviser

Blayne E. Mayfield

H. Lu

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my appreciation to and thank my thesis advisor Dr. Mansur H. Samadzadeh for his advisement, guidance, and instruction throughout my thesis research work.

I extend my appreciation to Drs. Blayne E. Mayfield and Huizhu Lu for serving on my graduate committee.

I also would like to thank the Government of Indonesia for providing funds for my graduate study through the Science and Technology for Industrial Development Program.

Finally, I would like to thank my family, my wife Maimun M. Naseeh and my son Husamuddin Barroq, for their patience during my graduate studies.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1

II. SOFTWARE COMPLEXITY METRICS 3

2.1 Size Metric 3
2.2 Halstead's Software Science 4
2.3 Cyclomatic Complexity 5
2.4 Residual Complexity Metric (RCM) 5
2.5 Measurement Scales 6

III. WEYUKER'S EVALUATION CRITERIA 7

3.1 Definition and Properties 7
3.2 Sample Evaluation 8
3.3 Criticisms 9

IV. A VARIANT OF THE RESIDUAL COMPLEXITY METRIC (VRCM) 12

4.1 Evaluation of the Residual Complexity Metric 12
4.2 Definition of a Variant of RCM (VRCM) 14
4.3 Properties of VRCM 16
4.3.1 Evaluation of VRCM 16
4.3.2 Growth of RCM and VRCM 19
4.3.3 Range of VRCM 20
4.4 Approximation of the Value of the Number of Consecutive Tokens 22
4.4.1 An Approach to Approximate the Value of p 23
4.4.2 Relationships among N, p, q, k_p 24
4.4.3 Constraints of the Approach 25
4.4.4 Analysis of the Tables 27

V. SUMMARY, CONCLUSIONS, AND FUTURE WORK 30
5.1 Summary 30
5.2 Conclusions 30
5.3 Future Work 31

REFERENCES 33

APPENDICES 36

A. Relationship Tables Among $N, p, q, {}_k p$ 37

A.1 Type 1 Approach 1 37
A.2 Type 2 Approach 1 48
A.3 Type 2 Approach 2 59

B. Glossary 70

LIST OF TABLES

Table Page

I. Weyuker's evaluation criteria applied to some software complexity metrics .. 9

II. Evaluation of RCM: $R_1, R_1U, R_2, R_2U, R_3, R_3U, R_4, R_4U$ 14

CHAPTER I

INTRODUCTION

Software metrics work can be defined as a method of evaluating the structural attributes of software. Various software metrics have been proposed by researchers and practitioners in the past 20 years. Generally, metrics are classified as product or process metrics. Product metrics are indicators that measure some aspects of a software product, such as a syntactic or structural aspect of a software product. Process metrics are measurable indicators of the software development process, such as the number of errors or the number of revisions. In addition to this classification, Fenton [Fenton91] considers resource metrics separately from process metrics. Resource metrics are indicators of the inputs to / outputs from processes, both in the direct sense of input/output values and in the indirect sense of environmental assumptions.

Attributes to be measured can be classified as internal and external attributes [Fenton91]. Internal attributes are attributes that are measured based on one of three entities: process, product, resources. External attributes are attributes that are measured based on how the three entities relate to their environment. A number of metrics that measure internal attributes relating to complexity metrics are discussed in the next chapter.

In the last few years, more industrial companies have considered the usage of some software metrics to evaluate their projects. Motorola reported that there have been some benefits gained by applying software metrics [Daskalantonakis92]. Also, in three years of

the Hewlett-Packard's metrics program, Grady and Caswell concluded that there had been fifteen advantages in using software metrics [Grady87].

With the existing large number of proposed metrics and new metrics being defined, the general properties of software metrics that might lead metrics into becoming 'good engineering tools' must be defined. Weyuker's evaluation criteria of software complexity measures provided some of these properties [Weyuker88].

Chapter II of this thesis discusses some software complexity metrics, including the Residual Complexity Metric (RCM), and their characteristics. Chapter III discusses Weyuker's evaluation criteria of software complexity measures. Chapter IV describes a Variant of the Residual Complexity Metric (VRCM), its properties, and some approaches to solve a problem that can occur in using VRCM. Finally, Chapter V contains the summary, conclusions, and some future areas of work.

CHAPTER II

SOFTWARE COMPLEXITY METRICS

The definition of complexity differs among researchers. Fenton defines complexity as a term to capture all internal attributes of software [Fenton91]. This term is used by a number of researchers to measure a number of internal (structural) product attributes. For instance, a complexity metric can measure the size, modularity, or control-flow of a program.

Much effort has been spent in attempting to capture software complexity. In general, quantifying a software product is carried out by creating mappings from the internal product attributes of the program into the real numbers. Some specific popular and well-known software complexity metrics are described below. Other work in software metrics aims to capture interconnectivity [Kafura81] [Robillard89], nesting depth [Harrison81] [Piwowarski82], knot measure [Woodward79], or other control flow or data flow attributes [Magel81] [Oviedo80].

2.1 Size Metric

The size metric is one of the metrics generally used in software evaluation. This metric measures various forms of size (magnitude) of a program. It can represent the lines of code, number of tokens, number of functions, or other measures that are based on these basic counts [Conte86].

Line of code (LOC) is a simple and widely-used metric. HP's metrics program considers LOC in its metric establishment [Grady87]. The definition of LOC will affect

the complexity. For instance, whether comment lines, blank lines, or declaration lines in a program should be considered as a factor in determining the complexity.

2.2 Halstead's Software Science

Halstead's metrics were developed using token counts as basic units. Halstead [Halstead77] did his work on the basis of the number of operators and operands. As one of the first pioneers in measuring complexity through software code, his publications have received a lot of attention by researchers. Although researchers have performed many validation studies on his results, there are some criticisms to these metrics [Shepperd94]. Some of Halstead's metrics are given below.

- Estimated Program Length

$$\hat{N} = \eta_1 \log \eta_1 + \eta_2 \log \eta_2$$

where η_1: number of unique operators

η_2: number of unique operands

- Program Volume

$$V = (N_1 + N_2) \log (\eta_1 + \eta_2)$$

where N_1: total number of operators

N_2: total number of operands

- Programming Effort

$$E = \frac{\eta_1 N_2 (N_1 + N_2) \log(\eta_1 + \eta_2)}{2 \eta_2}$$

2.3 Cyclomatic Complexity

Cyclomatic complexity was proposed by McCabe [McCabe76]. He developed the metric based on program control structure by applying a number of graph theoretic concepts. Cyclomatic complexity for a (program flow) graph G with n vertices, e edges, and p connected components is defined as

$$v(G) = e - n + 2p$$

McCabe demonstrated that

$$v(G) = \Pi + 1$$

where Π is the number of binary predicates in a program.

2.4 Residual Complexity Metric

Residual complexity metric (RCM) was proposed by Samadzadeh and Edwards [Samadzadeh88]. Experimental validation of this metric in software maintenance phase using industrial data was performed by Samadzadeh and Nandakumar [Samadzadeh91]. Their approach uses the operator and operand token classification, as well as their refinements into statement type token such as sequential, conditional, iterative, input, output, and processing tokens. All tokens are also classified in term of unique and non-unique tokens.

Understanding a software document can be modeled by token partitioning [Samadzadeh88]. The residual complexity metric was presented in order to measure the remaining complexity that remained uncovered by a specific level of partitioning or categorization of the tokens. The metric is derived based on the concept of computational work that can be measured in terms of the entropy function as used in information theory.

The residual complexity is defined [Samadzadeh88] as

$$R = N_1 \log N_1 + N_2 \log N_2 + \ldots + N_q \log N_q$$

where N_i stands for number of tokens in the ith block of the partition, and there are a total of q partitions of tokens.

Halstead's estimated program length can be directly derived from RCM. By choosing a value of 2 for q, RCM becomes the estimated program length [Samadzadeh 88]. Thus RCM can be viewed as a generalization of Halstead's estimated program length.

2.5 Measurement Scales

A definition of measurement is offered by Fenton [Fenton94] as follows.

> Measurement is defined as the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.

By assigning numbers or symbols, measurement must preserve observations about objects to be measured. Measurement activities must also have clear objectives. The Goal/Question/Metric paradigm (GQM) of Basili and Rombach [Basili88] provides that important point of a measurement system.

One important issue in the theory of measurement is scale. Measurement scale is classified to clarify the different possible measurement representations of objects' attribute [Fenton94]. There are four scale types (in increasing order): nominal, ordinal, interval, and ratio [Conte86] [Fenton94] [Zuse89]. As an example, addition and subtraction operations can be performed on measures belonging to the interval scale.

CHAPTER III

WEYUKER'S EVALUATION CRITERIA

3.1 Definition and Properties

Weyuker proposed several evaluation criteria for syntactic or structural complexity measures [Weyuker88]. A program in her paper consists of a program statement followed by a program body, which in turn is followed by an output statement. To simplify the discussion, the program body is called the program. The program is the object to be measured in the evaluation process.

She developed a system to evaluate complexity measures by defining a general set of properties for complexity measures that will help "to clarify the strengths and weaknesses of existing and proposed complexity measures" [Weyuker88]. The main goal of the evaluation criteria is to define rigorous properties for a formal definition of software complexity measures.

The proposed properties and some of the definitions dealing with Weyuker's evaluation criteria follow [Weyuker88]. Let P, Q, and R be programs. |P| denotes the complexity of program P. P;Q denotes the concatenation of programs P and Q.

<Property 1> $(\exists P)\ (\exists Q)\ (|P| \neq |Q|)$
There are programs that have different complexities.

<Property 2> Let c be a nonnegative number, there are only finitely many programs of complexity c.

<Property 3> There are distinct programs P and Q such that $|P| = |Q|$.

<Property 4> $(\exists P)\ (\exists Q)\ (P \equiv Q\ \&\ |P| \neq |Q|)$.
There are programs that compute the same function, but have different complexities.

<Property 5> $(\forall P)\ (\forall Q)\ (|P| \leq |P;Q|$ and $|Q| \leq |P;Q|)$
Complexity of a program P is less than or equal to that of the concatenation of P and another program.

<Property 6a> $(\exists P)\ (\exists Q)\ (\exists R)\ \ (|P| = |Q|\ \ \&\ \ |P;R|\ \neq |Q;R|)$
<Property 6b> $(\exists P)\ (\exists Q)\ (\exists R)\ \ (|P| = |Q|\ \ \&\ \ |R;P|\ \neq |R;Q|)$
There are programs that have the same complexity, but the concatenation of those programs and another program give different results in terms of their complexities.

<Property 7> There are program bodies P and Q such that Q is formed by permuting the order of the statements of P, and $|P| \neq |Q|$.

<Property 8> If P is a renaming of Q, then $|P| = |Q|$.

<Property 9> $(\exists P)\ (\exists Q)\ (|P| + |Q| < |P;Q|)$
There are programs where the complexity of their concatenation is less than that of the same programs computed separately and added together.

Properties 1 through 3 are about properties of measures. Property 4 says that the complexity of a function relies on its implementation. Property 5 states the monotonicity property of a measure. Properties 6, 7, and 9 deal with interactions among subprograms. Property 8 states the effect of changing variable names.

3.2 Sample Evaluation

Weyuker performed her evaluation on a number of complexity metrics [Weyuker88]. The metrics which were covered in her evaluation were statement count, cyclomatic complexity, Halstead's programming effort, and Oviedo's data flow complexity. The evaluation of the metrics are given in Table I.

Cyclomatic complexity cannot fulfill Property 2. The cyclomatic complexity is based on the control statement of a program. It does not consider other statement types. For a particular value of the cyclomatic complexity, we can create infinitely many instances of

other statement types. Therefore, infinitely programs can be created with the same cyclomatic complexity.

Table I. Weyuker's evaluation criteria applied to some software complexity metrics (Source: [Weyuker88])

Property	Statement Count	Cyc.Comp.	Effort	Data Flow Comp.
1	Y	Y	Y	Y
2	Y	N	Y	N
3	Y	Y	Y	Y
4	Y	Y	Y	Y
5	Y	Y	N	N
6	N	N	Y	Y
7	N	N	N	Y
8	Y	Y	Y	Y
9	N	N	Y	Y

Table I also shows that statement count, cyclomatic complexity, and effort cannot satisfy Property 7 that covers statement arrangement or token order. As a general rule, every token count measures do not consider arrangement or order [Weyuker88]. This property will be stressed in this thesis work.

3.3. Criticisms

An earlier attempt to define software complexity measures axiomatically was performed by Prather [Prather84]. The three axiomatic criteria proposed by Prather govern the structural behavior of software measures. These evaluation criteria are less restrictive than those of Weyuker's [Shepperd93]. This generalization can lead to the overall weakness of the evaluation criteria.

There are a number of criticisms of these evaluation criteria. First, There is no compatibility of the scale of measurement in all properties of these evaluation criteria. As an example, two properties (Properties 5 and 6) have contradictory scales of measurement

[Fenton94]. Property 5 of Weyuker's evaluation criteria requires the ratio scale, but Property 6 excludes the ratio scale.

The second criticism is in regard to the generalization of all attributes of software complexity measures. Although she tries to define the properties in general terms (not many details), in order to cover the general properties of software complexity measures, Weyuker's work cannot capture all attributes of software complexity measures. As an example, Property 5 has much to do with the size rather than the comprehensibility in determining complexity. On the other hand, in Property 6, comprehensibility contributes more to the complexity than software size [Fenton94].

Another criticism is that these evaluation criteria work at the code level. They cannot work at the design level. Property 2 says that from a certain value of complexity, we can create finitely many programs. In design level, given a particular complexity level, we can create infinitely many program designs. As a result, infinitely many programs can be created [Shepperd93].

From the previous two criticisms, it can be inferred that the criteria are not general (complete) for evaluating software complexity measures. They evaluate some important aspects of software complexity measures. More specifically, on the completeness of Weyuker's axioms, Fenton [Fenton94] states that:

> More importantly, what they fail to observe, is that Weyuker did not propose that the axioms were sufficient; she only proposed that they were necessary.

Another aspect of this criticism, because of the weakness of the generalization of axioms, is that the complexity of a software document cannot be determined by mapping a software document directly to a single real number [Fenton94]. A set of mappings is more realistic. Some hybrid metrics, which combine two or more software metrics, are alternatives to eliminate this weakness [Harrison81] [Hansen78] [Ramamurthy88]. A combination of a number of metrics using factor analysis was proposed by Munson and

Khoshgoftaar [Munson89] to address the generalization weakness of Weyuker's evaluation criteria. This proposed hybrid metrics together with its empirical validation was primarily concerned with the dimensionality of metrics. From the orthogonal metrics resulting from factor analysis, Munson and Khoshgoftaar constructed a relative complexity metric that measures the overall program complexity by creating a mapping from the representative metrics of each orthogonal metric into a single real number. This relative complexity metric is calculated using the covariance matrix of the representative metrics and their corresponding eigenvalues [Munson90] [Khosgoftaar94].

It can be argued that the meaning of attributes to be measured is more important than merely satisfying the properties of Weyuker's evaluation criteria. Cherniavsky and Smith demonstrated that they can create a metric that satisfies all properties, but is not a sensible measure of complexity [Cherniavsky91]. McColl and McKim [McColl92] developed a metric that satisfies all nine properties of Weyuker's evaluation criteria, and which works on a structured language known as WHILE language.

CHAPTER IV

A VARIANT OF THE RESIDUAL COMPLEXITY METRIC (VRCM)

4.1 Evaluation of the Residual Complexity Metric

The tokens used in the experimental validation of RCM by Samadzadeh and Nandakumar [Samadzadeh91] were the operator and operand tokens (R_1); the sequential, conditional, and repetitive tokens (R_2); the input, output, and processing tokens (R_3); and the refinement of operator (arithmetic, logical, and system) tokens and the refinement of operand (constant, and variable) tokens (R_4); as well as their non-unique tokens (R_1U, R_2U, R_3U, and R_4U).

RCM was evaluated using Weyuker's criteria. Properties 1, 3, 4, and 8 of Weyuker's evaluation criteria are clearly satisfied by RCM. The result of the study on the rest of the properties applied to RCM are reported below.

First, let's examine the applicability of Property 2 to RCM. If a particular value of RCM and the number of unique tokens are given, the total number of tokens N and the possible total number of tokens that belongs to each particular block of the partition N_i depend on these values. The last two values are finite because, from the definition of RCM, these values are less than a given particular value of RCM, i.e. $N_i \leq N < R$ for $N > 1$. For $N = 1$, those values are also clearly finite. From this finite total number of tokens, we can build finitely many programs of certain complexity. Therefore, Property 2 holds.

It is clear from the definition of RCM that additional tokens can increase the complexity as measured by RCM. If an additional token is a new token type, the

complexity remains the same. Otherwise, the complexity becomes larger. Therefore, Property 5 holds.

To satisfy Property 6, for unique and non-unique cases of RCM, we choose two programs P and Q that have the same complexity. Let $q = 2$ and let P consist of n tokens of the first type and n+1 tokens of the second type. Q consist of n+1 first type tokens and n second type tokens. Let R contain 1 first type tokens. The complexity of P and Q is the same, but $|P;R| \neq |Q;R|$. Hence, Property 6a holds.

Using the same reason stated by Weyuker [Weyuker88], the complexity calculated by RCM is independent from the placement of tokens in a program. Position of tokens in a program does not affect the complexity as calculated by RCM. Permuting the order of tokens does not change the complexity. Therefore, Property 7 cannot be satisfied by the RCM.

To satisfy Property 9, choose programs P and Q such that the complexity of P and Q are given by

$$|P| = A \log A + B \log B$$

$$|Q| = C \log C + D \log D$$

where A, B, C, D > 0 and they are the number of tokens in the corresponding classes. The complexity of the concatenation of programs P and Q is given by

$$\begin{aligned}
&(A + C) \log (A + C) + (B + D) \log (B + D) \\
&\quad = A \log (A + C) + C \log (A + C) + B \log (B + D) + D \log (B + D) \\
&\quad > A \log A + C \log C + B \log B + D \log D \\
&\quad > |P| + |Q|
\end{aligned}$$

Therefore, Property 9 holds. Table II shows the results.

Table II shows that the original RCM satisfies 8 of the 9 properties of Weyuker's evaluation criteria for software complexity metrics. Property 7, which RCM does not satisfy, was originally defined based on the assumption that the complexity becomes larger (or at least does not decrease) with additional program bulk because of the potential

interactions among the units. It also shows that RCM with the above token categorizations is not responsive to unit arrangement and token order.

Table II. Evaluation of RCM: $R_1, R_1U, R_2, R_2U, R_3, R_3U, R_4, R_4U$

Property	R_1	R_1U	R_2	R_2U	R_3	R_3U	R_4	R_4U
1	Y	Y	Y	Y	Y	Y	Y	Y
2	Y	Y	Y	Y	Y	Y	Y	Y
3	Y	Y	Y	Y	Y	Y	Y	Y
4	Y	Y	Y	Y	Y	Y	Y	Y
5	Y	Y	Y	Y	Y	Y	Y	Y
6	Y	Y	Y	Y	Y	Y	Y	Y
7	N	N	N	N	N	N	N	N
8	Y	Y	Y	Y	Y	Y	Y	Y
9	Y	Y	Y	Y	Y	Y	Y	Y

4.2 Definition of a Variant of RCM (VRCM)

Every bulk or token count measure, which categorizes the tokens of a program directly, is not responsive to the unit arrangement or token order property of Weyuker's evaluation criteria. To satisfy Property 7, a token categorization of RCM has to consider the token order. This consideration leads to the definition of a variant of RCM hereafter referred to as VRCM.

A variant of the residual complexity metric is developed by applying the concept of context locality, i.e., the original locality of tokens, to the original residual complexity metric definition. Partition classes for calculating VRCM are created based on the number of consecutive token. It can be observed that a token must consider a number of previous and subsequent tokens as its context.

Locality of tokens is bounded by how far (measured by "distance") each token can interact with other tokens as far as program comprehensibility is concerned. The maximum distance, at which a token is allowed to interact with other tokens in this thesis work, is

called the range of consecutive tokens. Let p and q be the range of consecutive tokens and the number of original token types, respectively. At the beginning and at the end of a program, there are classes that have lengths less than p (because there are no other previous or subsequent tokens to be observed). For this reason and for token concatenation, a null token is added to the original set of tokens. For $p = 1$, the number of possible partition classes is equal to the number of different token types, i.e., $q + 1$. Constant 1 represents the additional null token. For $p > 1$, the order of the original token types affects the forming of the partition classes of VRCM. Positions of the original token types, within the range of consecutive tokens, determine the total number of possible classes that can be created. For each position of an original token type, as mentioned earlier for $p = 1$, the number of possible partition classes is $q + 1$. If p is given, the number of all possible classes that can be created is the product of the number of all possible classes for each position, i.e., $(q + 1)\ (q + 1)\ ...(q + 1) = (q+1)^p$. Omitting a class all of whose elements are null, the total number of possible classes will be $(q+1)^p - 1$. As an example, let the original token set be {X,Y} and $p = 2$. The set of all possible classes that can be created is {XX, XY, Xe, YY, YX, Ye, ee, eX, eY}, where e represents the null token.

Based on the definition of RCM, which is

$$R = \sum_{i=1}^{q} (N_i \log N_i)$$

VRCM is defined as

$$R_V = \sum_{i=1}^{s} (N_i \log N_i)\ , \qquad s = (q+1)^p - 1, \qquad N_i \neq 0$$

But the elements of the classes that contain null tokens do not contribute to the complexity because each class only has one member . Therefore, VRCM can be written as

$$R_V = \sum_{i=1}^{s} (N_i \log N_i), \qquad s = q^p, \qquad N_i \neq 0$$

where p, q, R, and R_V stand for the range of consecutive tokens, the number of original token partitions, RCM, and VRCM, respectively.

4.3 Properties of VRCM

4.3.1 Evaluation of VRCM

The formal definition of VRCM is the same as the formal definition of RCM, except for the total number of possible classes. Some of the observations and results about VRCM are similar to those of RCM.

Properties 1, 3, 4, and 8 of Weyuker's evaluation criteria are clearly satisfied by VRCM. The results of the study on the other five properties applied to VRCM are presented below.

Using the same reasoning as in the evaluation of RCM, VRCM satisfies Properties 2 and 5. For Property 2, given a particular value of R_V, $N_i \leq N < R_V$ for $N > 1$. For $N = 1$, those values are also clearly finite. Therefore, Property 2 holds.

The proof of Theorem 1 (in Subsection 4.3.2 below) provides an argument for satisfying Property 5. It states that additional tokens at the end of a program cannot decrease the complexity as measured by VRCM. Therefore, Property 5 holds.

To satisfy the rest of the properties (Properties 6a, 7, and 9), we use proof by examples due to the use of existential quantifiers in some of the properties. Some of the properties in Weyuker's evaluation criteria indicate that we need the availability of at least one program that can satisfy those properties. For Property 6a, Example 1 below shows another approach that differs from that of the evaluation of RCM. This example also shows that Property 9 is satisfied by VRCM. Furthermore, Example 2 below shows that VRCM satisfies Property 7. Thus VRCM satisfies all of the 9 properties of Weyuker's evaluation criteria.

In the next two examples, consider a token classification that consists of three types of token (q=3): conditional (C), iterative (I), and sequential (S) [Samadzadeh91]. Let p = 2, then there are $q^p = 9$ possible partition classes that can contribute to the complexity. The partition classes are {CC, CI, CS, II, IC, IS, SS, SC, SI}, where CC stands for the first and the second token being conditional tokens, CI stands for the first token being a conditional token and the second token being an iterative token, etc. Examples below show that Properties 6, 7, and 9 hold. The partition class names, e.g., SS or SC, are used in the following two examples to refer to the size of each respective partition class as well.

Example 1:

Program P	Token Type	Program Q	Token Type	Program R	Token Type
x = 3;	S	x = 3;	S	for(j=0;j<=2;j++){	I
y = 7;	S	y = 7;	S	if (j==0) {	C
for(i=0;i<=3;i++){	I	if(x==3) {	C	j = 7;	S
if (x==3) {	C	for(i=0;i<=3;i++){	I	printf("%d",j);	S
y = x + y;	S	y = x + y;	S	} }	
if (y > 0) {	C	if (y > 0) {	C		
x = 3x + 7;	S	x = 3x + 7;	S		
y = 0;	S	y = 0;	S		
} } }		} } }			

|P| = SS log SS + SI log SI + IC log IC + CS log CS + SC log SC
= 2 log 2 + 1 log 1 + 1 log 1 + 2 log 2 + 1 log 1
= 4 log 2

|Q| = SS log SS + SC log SC + CI log CI + IS log IS + CS log CS
= 2 log 2 + 2 log 2 + 1 log 1 + 1 log 1 + 1 log 1
= 4 log 2

|R| = IC log IC + CS log CS + SS log SS
= 1 log 1 + 1 log 1 + 1 log 1
= 0

|P;R| = SS log SS + SI log SI + IC log IC + CS log CS + SC log SC
= 3 log 3 + 2 log 2 + 2 log 2 + 3 log 3 + 1 log 1
= 6 log 3 + 4 log 2

|Q;R| = SS log SS + SC log SC + CI log CI + IS log IS + CS log CS + SI log SI + IC log IC
= 3 log 3 + 2 log 2 + 1 log 1 + 1 log 1 + 2 log 2 + 1 log 1 + 1 log 1
= 3 log 3 + 4 log 2

|P| = |Q|, but |P;R| ≠ |Q;R|. Therefore, Property 6a holds.

|P| + |R| < |P;R|. Therefore, Property 9 holds.

Example 2:

Program P	Token Type	Program Q	Token Type
x = 3;	S	x = 3;	S
y = 7;	S	for(i=0;i<=3;i++) {	I
for(i=0;i<=3;i++) {	I	y = 7;	S
if (x==3) {	C	if (x==3) {	C
y = x + y;	S	y = x + y;	S
if (y > 0) {	C	if (y > 0) {	C
x = 3x + 7;	S	x = 3x + 7;	S
y = 0;	S	y = 0;	S
z = 2x;	S	z = 2x;	S
} } }		} } }	

Q is formed by interchanging the second and third lines in P.

|P| = SS log SS + SI log SI + IC log IC + CS log CS + SC log SC
= 3 log 3 + 1 log 1 + 1 log 1 + 2 log 2 + 1 log 1
= 3 log 3 + 2 log 2

|Q| = SI log SI + IS log IS + SC log SC + CS log CS + SS log SS
= 1 log 1 + 1 log 1 + 2 log 2 + 2 log 2 + 2 log 2
= 6 log 2

|P| ≠ |Q|

Therefore, Property 7 holds.

4.3.2 Growth of RCM and VRCM

As a variant of RCM, VRCM should retain specific properties of RCM. Concatenation of programs is one of the issues in Weyuker's evaluation criteria. This issue leads to another aspect of programs, i.e., the size growth. The theorem below addresses the growth of token counts for RCM and VRCM.

Theorem 1:

VRCM maintains the growth direction of RCM under concatenation of tokens.

Proof:

Let P and Q be sequences of tokens.

For RCM,

$$R(P) = \sum_{i=1}^{q} (N_i \log N_i)$$

Additional tokens will increase N, the total number of tokens. If additional tokens belong to empty classes, then R(P;Q) = R(P). If there is at least one additional token that belongs to a non-empty class, we have R(P;Q) > R(P). Therefore R(P) ≤ R(P;Q).

The same reasoning holds for VRCM. With additional tokens at the end of P, the classes that have null tokens at the end of P become classes that have no null token. If additional tokens belong to empty classes, then $R_V(P;Q) = R_V(P)$. Otherwise, $R_V(P) < R_V(P;Q)$. Therefore, $R_V(P) \leq R_V(P;Q)$.

♦

This theorem says that VRCM maintains the same growth direction as RCM. It also says that VRCM is not really a new metric. It is a modification of the RCM then considers token order.

4.3.3 Range of VRCM

To determine the range of VRCM, the maximum and minimum values of VRCM must be examined. Lemma 1 below gives the maximum value of both RCM and VRCM. This lemma also indicates a critical point that yields the maximum value.

Lemma 1:

The maximum value of RCM and VRCM is $N \log N$ which occurs when all tokens belong to one partition class.

Proof:

Assume that the maximum value occurs when tokens belong to more than one partition class. Then, by definition,

$$R = \sum_{i=1}^{q} (N_i \log N_i)$$

where $q>1$, $N_i \neq 0$, and $\sum_{i=1}^{q} N_i = N$

But,

$$\begin{aligned} N \log N &= (N_1 + N_2 + \ldots + N_q) \log (N_1 + N_2 + \ldots + N_q) \\ &= N_1 \log (N_1 + N_2 + \ldots + N_q) + N_2 \log (N_1 + N_2 + \ldots + N_q) \\ &\quad + \ldots + N_q \log (N_1 + N_2 + \ldots + N_q) \\ &> N_1 \log N_1 + N_2 \log N_2 + \ldots + N_q \log N_q \end{aligned}$$

Hence, the assumption that the maximum value occurs when more than one partition class is created is wrong. Therefore, the maximum value occurs when the partitions contain one class. By letting $q = 1$, the maximum value is $N \log N$.

♦

Without loss of generality, it is assumed that the domain of N_i is a positive real number for each i. This assumption is required to fulfill a necessary condition that a

function has partial derivatives (derivatives of a function on its particular independent variable assuming that other independent variables are constant). Lemma 2 gives the minimum value of this function.

Lemma 2:

If q is given, the minimum value of $\sum_{i=1}^{q}(N_i \log N_i)$ with constraints $\sum_{i=1}^{q} N_i = N$ for $N_i \geq 1$ and $N_i \in$ positive real number, is $\sum_{i=1}^{q}\left(\frac{N}{q}\log\frac{N}{q}\right)$.

Proof:

Using the Lagrange Multiplier [Protter64],

$$L(N_i) = \sum_{i=1}^{q}(N_i \log N_i) + \lambda(N - \sum_{i=1}^{q} N_i)$$

If L_{N_i} denotes the partial derivative of L respect to N_i, then

$$L_{N_i} = \log N_i + 1 - \lambda = 0 \quad \text{for each } i$$

or

$$N_i = N_j$$

Hence,

$$N_i = \frac{N}{q}$$

or

$(N/q, N/q, \ldots, N/q)$ is a critical point.

But,

$$L_{N_i N_j} = \begin{cases} 1/N_i, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

The quadratic form matrix of its partial derivative is not zero on its diagonal. The value of elements on its diagonal is q/N. The eigenvalues of this matrix, λs, are calculated [Scanlon67] as

$$\prod_{i=1}^{q} (L_{N_i N_i} - \lambda) = 0$$

Because $L_{N_i N_i} = q/N$ is positive for each i, λ is also positive for all i. Therefore, $(N/q, N/q, \ldots, N/q)$ is a point that gives a minimum value. By setting $N_i = N/q$, the minimum value is $\sum_{i=1}^{q} \left(\frac{N}{q} \log \frac{N}{q} \right)$.

♦

Theorem 2 below gives the possible range of values for VRCM. Using the same value of q, the range of VRCM is wider than that of RCM. This is caused by the relatively larger number of classes that VRCM has over RCM.

Theorem 2:

If N, q, and p are given, then $\sum_{i=1}^{q^p} \left(\frac{N}{q^p} \log \frac{N}{q^p} \right) < \text{VRCM} < N \log N$.

Proof:

From the definition of VRCM, the total number of partition classes that contribute to the complexity is q^p. Using Lemmas 1 and 2, the theorem holds.

♦

4.4 Approximation of the Value of the Number of Consecutive Tokens

Calculation of RCM depends only on the total number of tokens N, the number of tokens in each partition block N_i, and the number of partition classes q. In the calculation of VRCM, there is an additional parameter p (the number of consecutive tokens). With this additional parameter, there is a problem that can occur in calculating VRCM. The problem is how to determine the value of p.

The larger the value of p, the larger the total number of partition classes that can be created. On the other hand, the total number of tokens N increases at a much slower rate. Hence, a larger value of p tends to cause the distribution of tokens to spread evenly in the new token categorization. This spread lowers the value of VRCM. For example, a class

that has $(a + b)$ members spreads its member into two new partition classes. The first class get a members and the second class get b members. But $(a + b) \log (a + b) \geq a \log a + b \log b$. Therefore a larger the value of p tends to decrease the value of VRCM (decrease the remaining complexity that remains uncovered by the current token partitioning). But, there is still a limitation, in that the value of VRCM becomes zero, in the extreme case, if so many partition classes are created.

4.4.1 An Approach to Approximate the Value of p

To solve the problem of calculating token context in terms of the number of consecutive tokens to be considered, we need a representation of p that has the ability to measure the changes that are caused by the changes in the value of p. Let k_p be a constant that corresponds to a particular value of p assuming that N and q are given. Determining the value of p amounts to choosing the value of k_p.

In this approach, the changes of the value of R_V will be measured. The change can be measured in terms of p. To make it work, a representative value of R_V on each p has to be determined. In this approach, the average value of R_V on each p, R_p, will be used as a representative of R_V on each p.

Two methods of determining a constant, k_p, are chosen to measure the change of the value of R_p. The constants are defined below.

- Relative difference of two consecutive values for R_p, i.e., R_p and R_{p+1}

$$k_p = \frac{R_p - R_{p+1}}{R_p}$$

- Relative difference between R_p and the maximum value of VRCM

$$k_p = \frac{N \log N - R_p}{N \log N}$$

The first constant will be useful if R_p is a monotonic non-increasing function of p. It can measure the maximum tolerance of the changes of the average value of R_v. The second constant does not need the assumption that R_p is a monotonic non-increasing function of p. It measures the percentage of the average understanding a document as captured by token categorizations. For ease of reference, the first method of determining the constant is called Type 1 and the second one is called Type 2.

4.4.2 Relationships among N, p, q, k_p

As mentioned in the last subsection, the value of k_p will determine the relationships among N, p, and q. To calculate the value of k_p, the value of R_p must be determined or approximated. Based on this calculation or approximation, the appropriate constant type k_p can then be chosen.

Theorem 2 provides a basis for finding an approximate value for R_p. The average of the two bounds for VRCM is chosen as an approximate value for R_p (determining better approximate values for R_p is an area of future work). R_p can then be written as

$$
\begin{aligned}
R_p &= 1/2\ N \log N + 1/2 \sum_{i=1}^{q^p} \left(\frac{N}{q^p} \log \frac{N}{q^p} \right) \\
&= 1/2\ N \log N + 1/2\ q^p \left(\frac{N}{q^p} \log \frac{N}{q^p} \right) \\
&= 1/2\ N \log N + 1/2\ N \log \frac{N}{q^p} \\
&= 1/2\ N \log N + 1/2\ N \log N - 1/2\ N \log q^p \\
&= N \log N - 1/2\ Np \log q
\end{aligned}
$$

Clearly R_p is a linear function of p. On the basis of this linearity, we can use both of the constants to measure the relative changes. Type 1 is calculated as

$$k_p = \frac{R_p - R_{p+1}}{R_p}$$

$$= \frac{(N\log N - 1/2\,Np\log q) - (N\log N - 1/2\,N(p+1)\log q)}{(N\log N - 1/2\,Np\log q)}$$

$$= \frac{-1/2\,Np\log q + 1/2\,Np\log q + 1/2\,N\log q}{(N\log N - 1/2\,Np\log q)}$$

$$= \frac{1/2\,N\log q}{N\log N - 1/2\,Np\log q}$$

$$= \frac{\log q}{\log N^2 - \log q^p}$$

Type 2 is calculated as

$$k_p = \frac{N\log N - R_p}{N\log N}$$

$$= \frac{N\log N - (N\log N - 1/2\,Np\log q)}{N\log N}$$

$$= \frac{1/2\,p\log q}{\log N}$$

$$= \frac{\log q^p}{\log N^2}$$

4.4.3 Constraints of the Approach

There are two main constraints in this approach. First, in determining the value of R_p, it is assumed that $\frac{N}{q^p}$ is a real number. In fact, the number of tokens belonging to each partition class is an integer. To address this constraint, another method is applied, i.e., using the nearest integer numbers. If $f = \left\lfloor \frac{N}{q^p} \right\rfloor$ and $c = \left\lceil \frac{N}{q^p} \right\rceil$, and assuming that the

probability of f and c in substituting the value of $\frac{N}{q^p}$ is the same (equally probable), then the minimum value of VRCM becomes

$$\sum_{i=1}^{q^p}\left(\frac{N}{q^p}\log\frac{N}{q^p}\right) = \sum_{i=1}^{1/2\,q^p}(c\log c)+\sum_{i=1}^{1/2\,q^p}(f\log f)$$

$$= \frac{q^p}{2}[(c\log c)+(f\log f)]$$

$$= \frac{q^p}{2}[(f+1)\log(f+1)+f\log f]$$

As a result, the value of R_p becomes

$$\frac{1}{2}N\log N+\frac{1}{4}q^p[(f+1)\log(f+1)+f\log f]$$

In this case, R_p is not a monotonic function of p. This is caused by the non-monotonic behavior of the floor function. Based on this reason, Type 2 is an appropriate constant that measures the changes. The constant can be written as

$$k_p = \frac{N\log N - R_p}{N\log N}$$

$$= \frac{N\log N - \{1/2\,N\log N + 1/4\,q^p[(f+1)\log(f+1)+f\log f]\}}{N\log N}$$

$$= \frac{1/2\,N\log N - 1/4\,q^p[(f+1)\log(f+1)+f\log f]}{N\log N}$$

For ease of reference, this approximation is called Approach 2 and the previous approximation is called Approach 1.

Another constraint of having three approximate values is the possible values of p and q. From the value of R_p, the value of q^p is bounded above by the total number of tokens, N, because of the behavior of the logarithmic function. In fact, if particular values of N and q are given, we can assign the value of p up to N. This constraint says that the approaches cannot measure the change of the value of VRCM on a large number of possible partition classes.

4.4.4 Analysis of the Tables

The relationships among the parameters of VRCM are shown in Appendix A. The relationships are given by choosing some values for the parameters. With the assumption that there is a maximum total number of token N corresponding to a program module, in this analysis we choose the total number of tokens to be up to 150. Because of the monotonicity property of R_p as function of p, for any particular values of N and q, the value of k_p also inherits that monotonicity property, as shown in mathematical expressions of this constant in Subsection 4.4.2. This indicates that regardless of the value of N, the monotonicity property is preserved. For the example in Appendix A, we assume 150 is large enough. By repeatedly adding 5 to N, starting from 5, the tables are created. For a particular value of N, all possible values of p and q are considered in this analysis. Appendix A.1 covers Type 1 and Approach 1. Appendix A.2 covers Type 2 and Approach 1. Finally, Appendix A.3 covers Type 2 and Approach 2.

For Type 2, after choosing a particular value of k_p , for a particular value of N and q, the value of p can be determined from the corresponding table, and vice versa. For Type 1, it can be done in the same manner, but it does not determine the value of p directly. It works based on the consecutive values of p, i.e., p and $p + 1$. As mentioned in Section 4.4, Type 1 can measure the maximum tolerance of the changes of the average value of VRCM. Therefore, $p + 1$ corresponds to the value of k_p.

Based on the tables in Appendix A, the properties of the approaches are described below.

• For Approach 1, if p and q become larger, the value of the relative difference (k_p) becomes larger too. It shows that k_p is a monotonic non-decreasing function of p. For Type 1, q has a greater influence on the relative difference than p does. For Type 2, we cannot conclude which one has a greater influence. At some points, q has a greater influence than p does, and vice versa.

• For Approach 1, Type 1 the relative difference becomes smaller with the increasing value of N, for particular values of p and q. At some values of N, the values of k_p are almost the same. Furthermore, at large values of N, the relative difference between two consecutive values of p is also almost the same. Conversely, at the small values of N, these values have large differences.

• For Type 2, the values of the relative difference at the largest possible values of p at some points are 0.5. The definition of the relative difference of Type 2 says that this value can be achieved whenever the value of VRCM is 0. At these points, the boundary constraint is not a problem. In fact, the value of 0.5 is achieved whenever $N = q^p$. From the tables, generally, those values are around 0.4. This fact indicates, although there are some boundary constraints, that the approaches can capture a large percentage of average of understanding a document. From the tables, the range of Approach 1 for this type is 0.34 - 0.50. The range of Approach 2 is 0.32 - 0.46.

• In general, the properties of Approach 2, Type 2 are the same as the properties of Approach 1, Type 2. But, because of the behavior of the floor function, there are exceptions on some points. There can arise cases where with a larger value of q, a smaller value of the relative difference would result. For example, let $N = 120$ and $p = 2$. The

values of the relative differences for q = 8, 9, 10 are 0.46, 0.45, and 0.44, respectively. These exceptions violate the expected behavior of R_p that says, from the mathematical expression of R_p, that the larger the value of q, the smaller the value of R_p, and from the definition of Type 2, this increases the value of k_p.

CHAPTER V

SUMMARY, CONCLUSIONS, AND FUTURE WORK

5.1 Summary

In Chapter I, a general classification of software metrics and a classification of software attributes to be measured were introduced. Chapter II discussed the definition of software complexity and some specific software complexity metrics, especially the Residual Complexity Metric (RCM). Chapter III described Weyuker's evaluation criteria that contain 9 properties. Some criticisms of these evaluation criteria were also presented. Two main criticisms were the omission of the consideration of the scales of measurement and the incapability of the evaluation criteria in becoming a general measurer. Chapter IV described the evaluation of RCM using Weyuker's evaluation criteria. A variant of RCM was proposed in that chapter in order to satisfy Property 7 that covers token order. The properties of VRCM and two approaches to determine the ranges of consecutive tokens were also presented in that chapter.

5.2 Conclusions

The proposed VRCM is developed by applying the concept of context locality to the original RCM definition. VRCM satisfies all properties of Weyuker's evaluation criteria. VRCM also maintains the same growth direction as RCM. Determining the value of the number of consecutive tokens p is carried out by determining a constant that can represent p. Two approaches and two types of constants are proposed. These constants,

referred to as the relative differences, measure the changes that are caused by the changes in the value of p.

The relationships among the parameters of VRCM are presented in tables in Appendix A. Approach 2, using the fact that VRCM is actually computed based on integers, has a property that at some points possibly violates behavior of R_p (average value of understanding a document) defined on section 4.4. Approach 1 has no violation of behavior of R_p, but it works on real numbers as a generalization of VRCM. This generalization is a weak point of Approach 1.

There is a boundary constraint of the parameters of VRCM. The tables show that the theoretical approach of Type 2 can cover a large portion of the possible values of complexity (or a user's current level of a document's understanding). In reality, the full or almost full coverage of that understanding, by choosing a large value for p, will cause the metric, its distinguishing power, in that it will lose its ability to differentiate documents.

5.3 Future Work

Theorem 1 in Chapter IV says that VRCM is not a new metric. It also shows that concatenation of programs cannot decrease the complexity as measured by both RCM and VRCM. However, inserting additional tokens in the middle of a program can possibly decrease the value of VRCM. This phenomenon can be explained by the following example. Program P in Example 1 in Subsection 4.3.1 can be written as SSICSCSS. The complexity of this program is 4 log 2. Suppose token I is inserted between the sixth token (C) and seventh token (S). This new program can be written as SSICSCISS. This token insertion causes class CS to have only one member (previously it had 2 members), thus decreasing the value of VRCM. On the other hand, this token insertion causes the classes CI and IS both to have one member (previously they had no members), thus adding nothing to the value of VRCM. Therefore, the net result decreases the complexity as

measured by VRCM. In this example, the value of VRCM becomes 2 log 2. This anecdote suggests that experimental validation of VRCM is needed as an area of future work. It is also worth noting that Weyuker's evaluation criteria do not include the case of additional tokens inserted in the middle of a program.

In the approaches to determine the value of p, the minimum value of VRCM is used. This value is achieved based on the assumption that all tokens belong to all possible classes with the same probability. In fact, there are classes that have no members in VRCM. For example, in program P in Example 1 in Section 4.3, there are only 5 partition classes out of 9 that have members. Hence, an experimental work that studies the distribution of the non-empty and non-singleton classes as a function on p is needed. This distribution can determine the 'practical' minimum value of VRCM on each p.

The two approaches in determining the values of p (as mentioned in Section 4.4) have weaknesses. Specifically, in Approach 2, an improvement of the approach is required. The improvement can be done on a theoretical basis or through some adjustments such as an adjustment to the value k_p in the points that violate the behavior of R_p.

Instead of using the relative difference of Type 2, the ratio of R_p and the maximum value of VRCM can be used. This constant measures the residual understanding of a software document.

RCM is derived based on the concept of computational work that can be measured in terms of the entropy function as used in information theory. Information theory analysis of VRCM is also an area of future work.

Finally, as mentioned in Section 4.4, determining the approximate values of R_p, which can represent the average value of VRCM better, is an area of future work.

REFERENCES

[Basili88] V. Basili and D. Rombach, "The tame project: Towards improvement-orientated software environments", *IEEE Transactions on Software Engineering*, Vol. 14, No. 6, pp. 758-773, June 1988.

[Cherniavsky91] J. C. Cherniavsky and C. H. Smith, "On Weyuker's Axioms For Software Complexity Measures", *IEEE Transactions on Software Engineering*, Vol. 17, No. 6, pp. 636-638, June 1991.

[Conte86] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, "*Software Engineering Metrics and Models*", The Benjamin/Cummings Publishing Co., Menlo Park, CA, 1986.

[Daskalantonakis92] M. K. Daskalantonakis, "A Practical View of Software Measurement and Implementation Experiences Within Motorola", *IEEE Transactions on Software Engineering*, Vol. 18, No. 11, pp. 998-1010, November 1992.

[Ejiogu91] L. E. Ejiogu, "*Software Engineering with Formal Metrics*", QED Technical Publishing Group, Wellesley, MA, 1991.

[Fenton91] N. E. Fenton, "*Software Metrics: A Rigorous Approach*", Chapman & Hall, London, UK, 1991.

[Fenton94] N. E. Fenton, "Software Measurement: A Necessary Scientific Basis", *IEEE Transactions on Software Engineering*, Vol. 20, No. 3, pp. 199-206, March 1994.

[Grady87] R. B. Grady and D. L. Caswell, "*Software Metrics: Establishing a Company-Wide Program*", Prentice-Hall Inc., Englewood Cliffs, NJ, 1987.

[Halstead77] M. H. Halstead, "*Elements of Software Science*", Elsevier, New York, NY, 1977.

[Halstead79] M. H. Halstead, "Advances in Software Science", *Advances in Computers*, Vol. 18, Academic Press Inc., New York, NY, pp. 119-172, 1979.

[Hansen78] W. J. Hansen, "Measurement of Program Complexity by the Pair", *ACM SIGPLAN Notices*, Vol. 13, No. 3, pp. 29-33, March 1978.

[Harrison81] W. Harrison and K. Magel, "A Complexity Measure Based on Nesting Level", *ACM SIGPLAN Notices*, Vol. 16, No. 3, pp. 63-74, March 1981.

[Kafura81] D. Kafura and S. Henry, "Software Quality Metrics Based on Interconnectivity", *J. Systems and Software*, Vol. 2, No. 2, pp. 121-131, June 1981.

[Khoshgoftaar94] T. M. Khoshgoftaar, J. C. Munson, and D. L. Lanning, "Alternative Approaches for the Use of Metrics to Order Programs by Complexity", *J. Systems and Software*, Vol. 24, No. 3, pp. 211-221, March 1994.

[Magel81] K. Magel, "Regular Expressions in a Program Complexity Metric", *ACM SIGPLAN Notices*, Vol. 16, No. 7, pp. 61-65, July 1981.

[McCabe76] T. J. McCabe, "A Complexity Measure", *IEEE Transactions on Software Engineering*, Vol. SE-2, pp. 308-320, December 1976.

[McColl92] R. B. McColl and J. C. McKim, Jr., "Evaluating and Extending NPath as a Software Complexity Measure", *J. Systems and Software*, Vol. 17, No. 3, pp. 275-279, March 1992.

[Munson89] J. C. Munson and T. M. Khoshgoftaar, "The Dimensionality of Program Complexity", *Proceedings of the 11th International Conference on Software Engineering*, Pittsburgh, PA, pp. 245-253, 1989.

[Munson90] J. C. Munson and T. M. Khoshgoftaar, "Applications of a Relative Complexity Metric for Software Project Management", *J. Systems and Software*, Vol. 12, No. 3, pp. 283-291, July 1990.

[Nandakumar89] C. K. Nandakumar, *Quantifying the Software Maintenance Task: An Empirical Study of Complexity Metrics Across Versions*, MS Thesis, Computer Science Department, Oklahoma State University, Stillwater, OK, May 1989.

[Oviedo80] E. I. Oviedo, "Control Flow, Data Flow and Program Complexity", *Proceedings of COMPSAC80*, Chicago, IL, pp. 146-152, 1980.

[Piwowarski82] P. Piwowarski, "A Nesting Level Complexity Measure", *ACM SIGPLAN Notices*, Vol. 17, No. 9, pp. 44-50, September 1982.

[Prather84] R. E. Prather, "An Axiomatic Theory of Software Complexity Measure", *The Computer Journal*, Vol. 27, No. 4, pp. 340-347, November 1984.

[Protter64] M. H. Protter and C. B. Morrey, Jr., "*Modern Mathematical Analysis*", Addison-Wesley Publishing Co., Inc., Reading, MA, pp. 176-183, 1964

[Ramamurthy88] B. Ramamurthy and A. Melton, "A Synthesis of Software Science Measures and the Cyclomatic Number", *IEEE Transactions on Software Engineering*, Vol. 14, No. 8, pp. 1116-1121, August 1988.

[Robillard89] P. N. Robillard and G. Boloix, "The Interconnectivity Metrics: A New Metric Showing How a Program is Organized", *J. Systems and Software*, Vol. 10, No. 1, pp. 29-39, July 1989.

[Samadzadeh88] M. H. Samadzadeh and W. R. Edwards, Jr., *A Classification Model of Software Comprehension*, Computer Science Department, Oklahoma State University, Stillwater, OK, OSU-CIS-TR-88-01, 30 pages, 1988.

[Samadzadeh91] M. H. Samadzadeh and C. K. Nandakumar, "A Study of Software Metrics", *J. Systems and Software*, Vol. 16, No. 3, pp. 229-234, November 1991.

[Scanlon67] J. C. Scanlon, "*Advanced Calculus*", D. C. Heath and Co., Boston, MA, pp. 119-124, 1967.

[Shepperd93] M. Shepperd and D. Ince, "*Derivation and Validation of Software Metrics*", Oxford University Press Inc., New York, 1993.

[Shepperd94] M. Shepperd and D. C. Ince, "A Critique of Three Metrics", *J. Systems and Software*, Vol. 26, No. 3, pp. 197-210, September 1994.

[Weyuker88] E. J. Weyuker, "Evaluating Software Complexity Measures", *IEEE Transactions on Software Engineering*, Vol. 14, No. 9, pp. 1357-1365, September 1988.

[Woodward79] M. R. Woodward, M. A. Hennell, and D. Hedley, "A Measure of Control Flow Complexity in Program Text", *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 1, pp. 45-50, January 1979.

[Zuse89] H. Zuse and P. Bollmann, "Software Metrics: Using Measurement Theory to Describe the Properties and Scales of Static Software Complexity Metrics", *ACM SIGPLAN Notices*, Vol. 24, No. 8, pp. 23-33, August 1989.

APPENDICES

APPENDIX A: RELATIONSHIP TABLES AMONG N, p, q, k_p

APPENDIX A.1: TYPE 1, APPROACH 1

Relative difference for $N =$ 5, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.27						
3							
4							
5							
6							
7							
8							
9							
10							

Relative difference for $N =$ 10, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.18	0.22					
3	0.31						
4							
5							
6							
7							
8							
9							
10							

Relative difference for $N = 15$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.15	0.17					
3	0.25						
4							
5							
6							
7							
8							
9							
10							

Relative difference for $N = 20$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.13	0.15	0.18				
3	0.22						
4	0.30						
5							
6							
7							
8							
9							
10							

Relative difference for $N = 25$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.12	0.14	0.16				
3	0.21						
4	0.27						
5	0.33						
6							
7							
8							
9							
10							

Relative difference for $N = 30$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.11	0.13	0.15				
3	0.19	0.24					
4	0.26						
5	0.31						
6							
7							
8							
9							
10							

Relative difference for $N = 35$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.11	0.12	0.14	0.16			
3	0.18	0.22					
4	0.24						
5	0.29						
6							
7							
8							
9							
10							

Relative difference for $N = 40$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.10	0.12	0.13	0.15			
3	0.17	0.21					
4	0.23						
5	0.28						
6	0.32						
7							
8							
9							
10							

Relative difference for N = 45, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.10	0.11	0.13	0.14			
3	0.17	0.20					
4	0.22						
5	0.27						
6	0.31						
7							
8							
9							
10							

Relative difference for N = 50, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.10	0.11	0.12	0.14			
3	0.16	0.20					
4	0.22						
5	0.26						
6	0.30						
7	0.33						
8							
9							
10							

Relative difference for N = 55, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.09	0.10	0.12	0.13			
3	0.16	0.19					
4	0.21						
5	0.25						
6	0.29						
7	0.32						
8							
9							
10							

Relative difference for $N = 60$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.09	0.10	0.11	0.13			
3	0.15	0.18					
4	0.20						
5	0.24						
6	0.28						
7	0.31						
8							
9							
10							

Relative difference for $N = 65$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.09	0.10	0.11	0.12	0.14		
3	0.15	0.18					
4	0.20	0.25					
5	0.24						
6	0.27						
7	0.30						
8	0.33						
9							
10							

Relative difference for $N = 70$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.09	0.10	0.11	0.12	0.14		
3	0.15	0.17					
4	0.19	0.24					
5	0.23						
6	0.27						
7	0.30						
8	0.32						
9							
10							

Relative difference for $N = 75$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.09	0.10	0.11	0.12	0.13		
3	0.15	0.17					
4	0.19	0.24					
5	0.23						
6	0.26						
7	0.29						
8	0.32						
9							
10							

Relative difference for $N = 80$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.09	0.09	0.10	0.12	0.13		
3	0.14	0.17					
4	0.19	0.23					
5	0.22						
6	0.26						
7	0.29						
8	0.31						
9							
10							

Relative difference for $N = 85$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.08	0.09	0.10	0.11	0.13		
3	0.14	0.16	0.20				
4	0.18	0.23					
5	0.22						
6	0.25						
7	0.28						
8	0.31						
9	0.33						
10							

Relative difference for $N = 90$, Type 1, Approach 1

p ⇒ / q ⇓	12	23	34	45	56	67	78
2	0.08	0.09	0.10	0.11	0.13		
3	0.14	0.16	0.19				
4	0.18	0.22					
5	0.22						
6	0.25						
7	0.28						
8	0.30						
9	0.32						
10							

Relative difference for $N = 95$, Type 1, Approach 1

p ⇒ / q ⇓	12	23	34	45	56	67	78
2	0.08	0.09	0.10	0.11	0.12		
3	0.14	0.16	0.19				
4	0.18	0.22					
5	0.21						
6	0.24						
7	0.27						
8	0.30						
9	0.32						
10							

Relative difference for $N = 100$, Type 1, Approach 1

p ⇒ / q ⇓	12	23	34	45	56	67	78
2	0.08	0.09	0.10	0.11	0.12		
3	0.14	0.16	0.19				
4	0.18	0.22					
5	0.21						
6	0.24						
7	0.27						
8	0.29						
9	0.31						
10	0.33						

Relative difference for $N = 105$, Type 1, Approach 1

p ⇒ / q ⇓	12	23	34	45	56	67	78
2	0.08	0.09	0.10	0.11	0.12		
3	0.13	0.15	0.18				
4	0.18	0.21					
5	0.21						
6	0.24						
7	0.26						
8	0.29						
9	0.31						
10	0.33						

Relative difference for $N = 110$, Type 1, Approach 1

p ⇒ / q ⇓	12	23	34	45	56	67	78
2	0.08	0.09	0.09	0.10	0.12		
3	0.13	0.15	0.18				
4	0.17	0.21					
5	0.21						
6	0.24						
7	0.26						
8	0.28						
9	0.31						
10	0.32						

Relative difference for $N = 115$, Type 1, Approach 1

p ⇒ / q ⇓	12	23	34	45	56	67	78
2	0.08	0.09	0.09	0.10	0.12		
3	0.13	0.15	0.18				
4	0.17	0.21					
5	0.20						
6	0.23						
7	0.26						
8	0.28						
9	0.30						
10	0.32						

Relative difference for $N = 120$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.08	0.08	0.09	0.10	0.11		
3	0.13	0.15	0.17				
4	0.17	0.20					
5	0.20						
6	0.23						
7	0.26						
8	0.28						
9	0.30						
10	0.32						

Relative difference for $N = 125$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.08	0.08	0.09	0.10	0.11		
3	0.13	0.15	0.17				
4	0.17	0.20					
5	0.20	0.25					
6	0.23						
7	0.25						
8	0.27						
9	0.29						
10	0.31						

Relative difference for $N = 130$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.08	0.08	0.09	0.10	0.11	0.12	
3	0.13	0.15	0.17				
4	0.17	0.20					
5	0.20	0.25					
6	0.23						
7	0.25						
8	0.27						
9	0.29						
10	0.31						

Relative difference for $N = 135$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.08	0.08	0.09	0.10	0.11	0.12	
3	0.13	0.14	0.17				
4	0.16	0.20					
5	0.20	0.24					
6	0.22						
7	0.25						
8	0.27						
9	0.29						
10	0.31						

Relative difference for $N = 140$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.08	0.08	0.09	0.10	0.11	0.12	
3	0.13	0.14	0.17				
4	0.16	0.19					
5	0.19	0.24					
6	0.22						
7	0.25						
8	0.27						
9	0.29						
10	0.30						

Relative difference for $N = 145$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.07	0.08	0.09	0.10	0.11	0.12	
3	0.12	0.14	0.17				
4	0.16	0.19					
5	0.19	0.24					
6	0.22						
7	0.24						
8	0.26						
9	0.28						
10	0.30						

Relative difference for $N = 150$, Type 1, Approach 1

p ⇒	12	23	34	45	56	67	78
q ⇓							
2	0.07	0.08	0.09	0.10	0.11	0.12	
3	0.12	0.14	0.16				
4	0.16	0.19					
5	0.19	0.24					
6	0.22						
7	0.24						
8	0.26						
9	0.28						
10	0.30						

APPENDIX A.2: TYPE 2, APPROACH 1

Relative difference for $N = 5$, Type 2, Approach 1

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.43						
3							
4							
5							
6							
7							
8							
9							
10							

Relative difference for $N = 10$, Type 2, Approach 1

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.30	0.45					
3	0.48						
4							
5							
6							
7							
8							
9							
10							

Relative difference for N = 15, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.26	0.38					
3	0.41						
4							
5							
6							
7							
8							
9							
10							

Relative difference for N = 20, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.23	0.35	0.46				
3	0.37						
4	0.46						
5							
6							
7							
8							
9							
10							

Relative difference for N = 25, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.22	0.32	0.43				
3	0.34						
4	0.43						
5	0.50						
6							
7							
8							
9							
10							

Relative difference for $N = 30$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.20	0.31	0.41				
3	0.32	0.48					
4	0.41						
5	0.47						
6							
7							
8							
9							
10							

Relative difference for $N = 35$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.19	0.29	0.39	0.49			
3	0.31	0.46					
4	0.39						
5	0.45						
6							
7							
8							
9							
10							

Relative difference for $N = 40$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.19	0.28	0.38	0.47			
3	0.30	0.45					
4	0.38						
5	0.44						
6	0.49						
7							
8							
9							
10							

Relative difference for $N = 45$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.18	0.27	0.36	0.46			
3	0.29	0.43					
4	0.36						
5	0.42						
6	0.47						
7							
8							
9							
10							

Relative difference for $N = 50$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.18	0.27	0.35	0.44			
3	0.28	0.42					
4	0.35						
5	0.41						
6	0.46						
7	0.50						
8							
9							
10							

Relative difference for $N = 55$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.17	0.26	0.35	0.43			
3	0.27	0.41					
4	0.35						
5	0.40						
6	0.45						
7	0.49						
8							
9							
10							

Relative difference for $N = 60$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.17	0.25	0.34	0.42			
3	0.27	0.40					
4	0.34						
5	0.39						
6	0.44						
7	0.48						
8							
9							
10							

Relative difference for $N = 65$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.17	0.25	0.33	0.42	0.50		
3	0.26	0.39					
4	0.33	0.50					
5	0.39						
6	0.43						
7	0.47						
8	0.50						
9							
10							

Relative difference for $N = 70$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.16	0.24	0.33	0.41	0.49		
3	0.26	0.39					
4	0.33	0.49					
5	0.38						
6	0.42						
7	0.46						
8	0.49						
9							
10							

Relative difference for $N = 75$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.16	0.24	0.32	0.40	0.48		
3	0.25	0.38					
4	0.32	0.48					
5	0.37						
6	0.42						
7	0.45						
8	0.48						
9							
10							

Relative difference for $N = 80$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.16	0.24	0.32	0.40	0.47		
3	0.25	0.38					
4	0.32	0.47					
5	0.37						
6	0.41						
7	0.44						
8	0.47						
9							
10							

Relative difference for $N = 85$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.16	0.23	0.31	0.39	0.47		
3	0.25	0.37	0.49				
4	0.31	0.47					
5	0.36						
6	0.40						
7	0.44						
8	0.47						
9	0.49						
10							

Relative difference for $N = 90$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.23	0.31	0.39	0.46		
3	0.24	0.37	0.49				
4	0.31	0.46					
5	0.36						
6	0.40						
7	0.43						
8	0.46						
9	0.49						
10							

Relative difference for $N = 95$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.23	0.30	0.38	0.46		
3	0.24	0.36	0.48				
4	0.30	0.46					
5	0.35						
6	0.39						
7	0.43						
8	0.46						
9	0.48						
10							

Relative difference for $N = 100$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.23	0.30	0.38	0.45		
3	0.24	0.36	0.48				
4	0.30	0.45					
5	0.35						
6	0.39						
7	0.42						
8	0.45						
9	0.48						
10	0.50						

Relative difference for $N = 105$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.22	0.30	0.37	0.45		
3	0.24	0.35	0.47				
4	0.30	0.45					
5	0.35						
6	0.38						
7	0.42						
8	0.45						
9	0.47						
10	0.49						

Relative difference for $N = 110$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.22	0.29	0.37	0.44		
3	0.23	0.35	0.47				
4	0.29	0.44					
5	0.34						
6	0.38						
7	0.41						
8	0.44						
9	0.47						
10	0.49						

Relative difference for $N = 115$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.22	0.29	0.37	0.44		
3	0.23	0.35	0.46				
4	0.29	0.44					
5	0.34						
6	0.38						
7	0.41						
8	0.44						
9	0.46						
10	0.49						

Relative difference for $N = 120$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.14	0.22	0.29	0.36	0.43		
3	0.23	0.34	0.46				
4	0.29	0.43					
5	0.34						
6	0.37						
7	0.41						
8	0.43						
9	0.46						
10	0.48						

Relative difference for $N = 125$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.14	0.22	0.29	0.36	0.43		
3	0.23	0.34	0.46				
4	0.29	0.43					
5	0.33	0.50					
6	0.37						
7	0.40						
8	0.43						
9	0.46						
10	0.48						

Relative difference for $N = 130$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.14	0.21	0.28	0.36	0.43	0.50	
3	0.23	0.34	0.45				
4	0.28	0.43					
5	0.33	0.50					
6	0.37						
7	0.40						
8	0.43						
9	0.45						
10	0.47						

Relative difference for $N = 135$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.14	0.21	0.28	0.35	0.42	0.49	
3	0.22	0.34	0.45				
4	0.28	0.42					
5	0.33	0.49					
6	0.37						
7	0.40						
8	0.42						
9	0.45						
10	0.47						

Relative difference for $N = 140$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.14	0.21	0.28	0.35	0.42	0.49	
3	0.22	0.33	0.44				
4	0.28	0.42					
5	0.33	0.49					
6	0.36						
7	0.39						
8	0.42						
9	0.44						
10	0.47						

Relative difference for $N = 145$, Type 2, Approach 1

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.14	0.21	0.28	0.35	0.42	0.49	
3	0.22	0.33	0.44				
4	0.28	0.42					
5	0.32	0.49					
6	0.36						
7	0.39						
8	0.42						
9	0.44						
10	0.46						

Relative difference for $N = 150$, Type 2, Approach 1

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.14	0.21	0.28	0.35	0.42	0.48	
3	0.22	0.33	0.44				
4	0.28	0.42					
5	0.32	0.48					
6	0.36						
7	0.39						
8	0.42						
9	0.44						
10	0.46						

APPENDIX A.3: TYPE 2, APPROACH 2

Relative difference for $N = 5$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.33						
3							
4							
5							
6							
7							
8							
9							
10							

Relative difference for $N = 10$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.30	0.38					
3	0.36						
4							
5							
6							
7							
8							
9							
10							

Relative difference for $N = 15$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.28	0.43					
3	0.42						
4							
5							
6							
7							
8							
9							
10							

Relative difference for $N = 20$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.19	0.34	0.41				
3	0.32						
4	0.41						
5							
6							
7							
8							
9							
10							

Relative difference for $N = 25$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.20	0.28	0.43				
3	0.37						
4	0.43						
5	0.39						
6							
7							
8							
9							
10							

Relative difference for $N = 30$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.20	0.33	0.45				
3	0.31	0.41					
4	0.45						
5	0.42						
6							
7							
8							
9							
10							

Relative difference for $N = 35$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.21	0.28	0.35	0.41			
3	0.34	0.42					
4	0.35						
5	0.43						
6							
7							
8							
9							
10							

Relative difference for $N = 40$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.17	0.25	0.37	0.42			
3	0.29	0.44					
4	0.37						
5	0.44						
6	0.42						
7							
8							
9							
10							

Relative difference for $N = 45$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.17	0.28	0.39	0.44			
3	0.25	0.45					
4	0.39						
5	0.45						
6	0.43						
7							
8							
9							
10							

Relative difference for $N = 50$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.18	0.25	0.32	0.44			
3	0.28	0.45					
4	0.32						
5	0.35						
6	0.44						
7	0.41						
8							
9							
10							

Relative difference for $N = 55$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.18	0.28	0.34	0.45			
3	0.25	0.36					
4	0.34						
5	0.37						
6	0.44						
7	0.42						
8							
9							
10							

Relative difference for $N = 60$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.25	0.36	0.45			
3	0.28	0.37					
4	0.36						
5	0.38						
6	0.45						
7	0.43						
8							
9							
10							

Relative difference for $N = 65$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.16	0.23	0.30	0.36	0.42		
3	0.25	0.38					
4	0.30	0.42					
5	0.39						
6	0.45						
7	0.44						
8	0.42						
9							
10							

Relative difference for $N = 70$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.16	0.26	0.32	0.37	0.43		
3	0.27	0.39					
4	0.32	0.43					
5	0.40						
6	0.46						
7	0.44						
8	0.43						
9							
10							

Relative difference for $N = 75$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.17	0.24	0.33	0.38	0.43		
3	0.25	0.40					
4	0.33	0.43					
5	0.33						
6	0.37						
7	0.45						
8	0.43						
9							
10							

Relative difference for $N = 80$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.22	0.29	0.39	0.44		
3	0.27	0.41					
4	0.29	0.44					
5	0.34						
6	0.38						
7	0.45						
8	0.44						
9							
10							

Relative difference for $N = 85$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.24	0.30	0.40	0.44		
3	0.24	0.34	0.43				
4	0.30	0.44					
5	0.35						
6	0.39						
7	0.46						
8	0.44						
9	0.43						
10							

Relative difference for $N = 90$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.15	0.22	0.31	0.41	0.45		
3	0.23	0.35	0.43				
4	0.31	0.45					
5	0.36						
6	0.40						
7	0.46						
8	0.45						
9	0.43						
10							

Relative difference for $N = 95$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.16	0.24	0.33	0.41	0.45		
3	0.24	0.36	0.44				
4	0.33	0.45					
5	0.37						
6	0.40						
7	0.46						
8	0.45						
9	0.44						
10							

Relative difference for $N = 100$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.14	0.23	0.29	0.35	0.45		
3	0.23	0.37	0.44				
4	0.29	0.45					
5	0.32						
6	0.41						
7	0.38						
8	0.45						
9	0.44						
10	0.42						

Relative difference for $N = 105$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.14	0.21	0.30	0.36	0.45		
3	0.24	0.38	0.44				
4	0.30	0.45					
5	0.33						
6	0.41						
7	0.38						
8	0.45						
9	0.44						
10	0.43						

Relative difference for $N = 110$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.23	0.31	0.36	0.46		
3	0.23	0.32	0.45				
4	0.31	0.46					
5	0.34						
6	0.35						
7	0.39						
8	0.46						
9	0.45						
10	0.43						

Relative difference for $N = 115$, Type 2, Approach 2

p ⇒	2	3	4	5	6	7	8
q ⇓							
2	0.15	0.22	0.28	0.37	0.46		
3	0.24	0.33	0.45				
4	0.28	0.46					
5	0.34						
6	0.35						
7	0.39						
8	0.46						
9	0.45						
10	0.44						

Relative difference for $N = 120$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.14	0.20	0.29	0.38	0.46		
3	0.22	0.34	0.45				
4	0.29	0.46					
5	0.35						
6	0.36						
7	0.40						
8	0.46						
9	0.45						
10	0.44						

Relative difference for $N = 125$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.14	0.22	0.30	0.38	0.46		
3	0.24	0.35	0.45				
4	0.30	0.46					
5	0.31	0.43					
6	0.37						
7	0.40						
8	0.46						
9	0.45						
10	0.44						

Relative difference for $N = 130$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.14	0.21	0.27	0.33	0.38	0.43	
3	0.22	0.36	0.46				
4	0.27	0.38					
5	0.31	0.43					
6	0.37						
7	0.41						
8	0.38						
9	0.46						
10	0.45						

Relative difference for $N = 135$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.14	0.22	0.28	0.34	0.39	0.43	
3	0.21	0.31	0.46				
4	0.28	0.39					
5	0.32	0.43					
6	0.38						
7	0.41						
8	0.39						
9	0.46						
10	0.45						

Relative difference for $N = 140$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.13	0.21	0.29	0.34	0.39	0.44	
3	0.22	0.32	0.46				
4	0.29	0.39					
5	0.33	0.44					
6	0.38						
7	0.42						
8	0.39						
9	0.46						
10	0.45						

Relative difference for $N = 145$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.14	0.20	0.26	0.35	0.40	0.44	
3	0.21	0.32	0.46				
4	0.26	0.40					
5	0.34	0.44					
6	0.33						
7	0.42						
8	0.40						
9	0.46						
10	0.45						

Relative difference for $N = 150$, Type 2, Approach 2

p ⇒ / q ⇓	2	3	4	5	6	7	8
2	0.14	0.21	0.27	0.36	0.40	0.44	
3	0.22	0.33	0.46				
4	0.27	0.40					
5	0.30	0.44					
6	0.34						
7	0.36						
8	0.40						
9	0.46						
10	0.45						

APPENDIX B

GLOSSARY

Complexity:	A measure of a number of internal (structural) product attributes.
External Attributes:	Attributes that are measured based on how process, product, and resources relate to their environment.
Internal Attributes:	Attributes that are measured based on one of three entities: process, product, resources.
Partition Classes:	Classes resulting from token categorization.
Process Metrics:	Measurable indicators of the software development process, such as the number of errors and the number of revisions.
Product Metrics:	Indicators that measure some syntactic or structural aspects aspects of a software product.
RCM:	Residual Complexity Metric, a metric that measure the remaining complexity of a software document that has been subjected a token categorization.
Resource Metrics:	Indicators of the inputs to / outputs from processes.
Scale Types of Measurement:	Classification of measurement scales that consists of the the nominal, ordinal, interval, and ratio scales.
Software Metrics:	Indicators of the structural attributes of software.
VRCM:	A variant of that RCM that applies the concept of context locality to the original definition of RCM.

VITA

Zain Saifullah

Candidate for the Degree of

Master of Science

Thesis: A VARIANT OF THE RESIDUAL COMPLEXITY METRIC

Major Field: Computer Science

Biographical:

Personal Data: Born in Jakarta, Indonesia, On December 22, 1961, son of Mr.M. Zainuddin Abdullah and Mrs. Siti Aminah.

Education: Received Sarjana Matematika degree from Institut Teknologi Bandung, Bandung, Indonesia in June 1986; completed the requirements for the Master of Science Degree at Oklahoma State University in July 1995.

Professional Experience: From July 1987 to June 1991 worked as Systems Programmer for Directorate for the Assessment of Technology in Electronics and Informatics, the Agency for Assessment and Application of Technology (BPPT), Jakarta, Indonesia.