Measuring Complexity, Development Time and Understandability of a Program: A Cognitive Approach

Amit Kumar Jakhar
Department of CSE, BIT, Mesra, Ranchi-835 215, Jharkhand, India
Email: amitkumar.cs08@pec.edu.in

Kumar Rajnish
Department of CSE, BIT, Mesra, Ranchi-835 215, Jharkhand, India
Email: krajnish@bitmesra.ac.in

Abstract — One of the central problems in software engineering is the inherent complexity. Since software is the result of human creative activity and cognitive informatics plays an important role in understanding its fundamental characteristics. This paper models one of the fundamental characteristics of software complexity by examining the cognitive weights of basic software control structures. Cognitive weights are the degree of the difficulty or relative time and effort required for comprehending a given piece of software, which satisfy the definition of complexity. Based on this approach a new concept of New Weighted Method Complexity (NWMC) of software is developed. Twenty programs are distributed among 5 PG students and development time is noted of all of them and mean is considered as the actual time needed time to develop the programs and Understandability (UA) is also measured of all the programs means how much time needed to understand the code. This paper considers Jingqiu Shao et al Cognitive Functional Size (CFS) of software for study. In order to validate the new complexity metrics we have calculated the correlation between proposed metric and CFS with respect to actual development time and performed analysis of NWMC with CFS with Mean Relative Error (MRE) and Standard Deviation (Std.). Finally, the authors found that the accuracy to estimate the development time with proposed measure is far better than CFS.

Index Terms — Program Complexity, Development Time, Understandability, Cognitive Weight, Basic Control Structures, Cognitive Functional Size, Lines of Code

I. INTRODUCTION

With the growth in technology most of the businesses are now being controlled by software’s, making the cost, schedule and quality estimation are critical concerns of businesses. Thus, the development of software has also become increasingly sophisticated over the years. With the intention of finding the most appropriate attributes such as control flow [1], operator and operand count [2], information flow [3], data flow [4-5], identifier density [6], spatial complexity [7-8] and cognitive complexity [9-11] that measure the development and maintenance cost associated with developing software. Although, each of these proposed complexity metrics have their advantages and disadvantages, most of them are useful only in a limited environment. Thus, in addition to proposing new complexity metrics, some computer scientists and researchers have attempted to identify metrics that addresses most of the characteristics of a software programs. However, only a few studies have been conducted to test the relationship between the proposed complexity metrics and their applicability in the real world. To identify the effectiveness and the practicality of a metric a number of evaluation frameworks have been proposed in the past [12-15].

The complexity measures based on cognitive informatics is in development phase. Cognitive complexity measures represent the human effort needed to perform a task or difficulty in understanding the software code. In cognitive informatics, it is found that the functional complexity of software in design and comprehension is dependent on three fundamental factors: internal processing, input and output [16-17]. NWMC is the new measure for measuring software complexity presented in this paper. It is a measure of the cognitive and psychological complexity of software as a human intelligence artefact. Cognitive complexity takes into account both internal structures of software and I/Os it processes.

The main aim of this paper is to measure the cognitive complexity with NWMC, analyzing the development time of programs and the measure the understandability of a program. To accomplish this work, the authors compared the proposed cognitive complexity (NWMC) and Cognitive Functional Size (CFS). The development time of NWMC and CFS is analyzed with actual development time and calculate their MRE and Standard Deviation. At last, understandability of a program is also measured, that can help to reduce difficulty which is tackled at the maintenance phase.

The rest of the paper is organized as follows: Section 2 deals with CFS presented by Shao et al., the basic control structure of a program and their corresponding weight is also shown in this section. Section 3 present a new complexity measure based on cognitive informatics and finding development time and Understandability of
programs. The comparison of proposed measure with CFS and LOC has been done in Section 4. Section 5 deals with conclusion and the future works of this research paper.

II. CFS of Shao and Wang

Over the years the term software complexity has been defined in several ways by a number of authors and researchers. The IEEE definition of software complexity is “the degree to which a system or component has a design or implementation that is difficult to understand and verify [18]”. Basili defined software complexity as “a measure of the resources expended by a system while interacting with a piece of software to perform a given task. If the interacting system is a computer, then complexity is defined by the execution time and storage required to perform the computation. If the interacting system is a programmer, then complexity is defined by the difficulty of performing tasks such as coding, debugging, testing or modifying the software [19].

The cognitive weight of software is defined “as the extent of difficulty or relative time and effort for comprehending given software modelled by a number of Basic Control Structures (BCS) [9].” A BCS is the basic building block of any software regardless of the underline technology. According to C.A.R. Hoare et al., there are five types of BCSs: sequential, branch, iteration, recursion and parallel [20]. In 2002, Y. Wang proposed two more BCSs: function call and interrupt [17]. Each BCS has been allocated with a unique cognitive weight [9]. The cognitive weights for BCS are as under:

<table>
<thead>
<tr>
<th>Category</th>
<th>BCS</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>Sequence (SEQ)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>If-Then-Else (ITE)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Case</td>
<td>3</td>
</tr>
<tr>
<td>Iteration</td>
<td>For-do</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Repeat-until</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>While-do</td>
<td>3</td>
</tr>
<tr>
<td>Embedded</td>
<td>Function Call (FC)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Recursion (REC)</td>
<td>3</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Parallel (PAR)</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Interrupt (INT)</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1. Basic Control Structures and their weights

The cognitive weight of a block changes depending on the existence of layers. If the BSCs are not nested and is in same layer, i.e., if there are no nested ‘if’ conditions, ‘for’ or ‘while’ loops, the cognitive weight is calculated by taking the sum of all the BSCs within that block. However, in the case of existence of nesting, the cognitive weight is calculated by multiplying the cognitive weight of the inner BSCs with cognitive weight of outer BSCs. Hence, for a component with q linear blocks which consist of m layers of nesting BCSs with each layer having n linear BCSs, the total cognitive weight (Wc) is defined as [9]:

\[ W_c = \sum_{j=1}^{q} \prod_{k=1}^{m} \sum_{i=1}^{n} W_c(j,k,i) \]  

(1)

If the q blocks do not contain any embedded BCSs, then the above equation can be simplified as follows [9]:

\[ W_c = \sum_{j=1}^{q} \sum_{k=1}^{m} (j,k) \]  

(2)

The CFS of software with N inputs, N outputs and a single method is defined as [9]:

\[ S_i = N_{io} \ast W_c \]  

(3)

where, CWU is the unit of cognitive weight of software.

If the software consists of n methods, then CFS of the \( e \)th method is defined as [9]:

\[ S_f(c) = \sum_{c=1}^{n} S_f(c) \]  

[CWU]  

(4)

Thus, for a component-based software system with p components, CFS is defined as [9]:

\[ \hat{S}_f = \sum_{p=1}^{n_c} S_f(p) \]  

\[ = \sum_{p=1}^{n_c} \sum_{c=1}^{n_c} S_f(p,c) \]  

(5)

III. NEW WEIGHTED METHOD COMPLEXITY (NWMC)

3.1 Definition

By considering the above theories of CFS presented by Shao et al [9], the authors had incorporated additional parameters in proposed cognitive weight measurement for measuring the complexity of program. In fact, cognitive weights correspond to the number of executed instructions and using this complexity number the authors make a relation to calculate the development time of the program, and compare the calculated time with the experimental developmental time which was observed when five users are trying to develop the code and mean of the five is consider as the actual development time. MRE is calculated for the both the proposed technique and CFS. Standard Deviation is also calculated of all 20 programs. At last, the understandability of a program is measured empirically with the help of NWMC that can be used to decrease the difficulty level and increase the quality, because UA is the quality indicator.

The NWMC metric may be defined as:

\[ NWMC = N_{parameter} \ast W_c \]  

(6)

Copyright © 2014 MECS
I.J. Information Technology and Computer Science, 2014, 12, 53-60
where,
\[ N_{\text{parameters}} = (N_i + N_o + N_{fp} + N_{lp}) \]
(7)
where,
- \( N_i \): individual number of inputs of the main program and other program which is called from the main program and some other program.
- \( N_o \): individual number of outputs of the main program and other program which is called from the main program and some other program.
- \( N_{fp} \): number of local parameters other than \( N_i \) used in a main program and other program which is called from the main program and some other program.
- \( N_{lp} \): number of formal parameters during function call and recursive call from the main program and other program which is called from the main program and some other program.
- \( W_i \): calculated from equation (1).

After calculating the complexity, the authors measure the development time which is observed from experiments with five different students in the institute lab. The time has taken by the students to develop the program was calculated and the average of all five students development time is considered as actual development time. The proposed work is incorporated with a new formula i.e.

Development Time = 
\[ a + b \times (\text{Cognitive Complexity number}) \]
(8)
where the value of \( a \) and \( b \) are calculated with the help of regression. The same formula is also applied to calculate the development time with the help of CFS.

The value of \( a \), \( b \) for NWMC are given below:

\( a = 0.9793 \)
\( b = 0.0964 \)
and the value of \( a \), \( b \) for CFS are:

\( a = 0.0004 \)
\( b = 0.7454 \)

Development time is calculated with these constant putting into the Development Time formula for NWMC and CFS. Then compare the result with actual time refer Table 3, 4 and Fig. 5, 6, 7.

3.2 Example for Illustration
A program to sort an array using selection sort is shown in Fig. 1. An algorithm contains 3 program bodies: one main program (P1), another program P2 (called from P1), and another program P3 (directly called from P2 and indirect call to P1). CFS and NWMC of Fig. 1 is calculated as follows:

**Calculation of CFS**

The cognitive weights for the three programs (P1, P2, and P3) are as follows:

- For program P1, there are four internal structures: a sequence, two iterations and a function call BCSs. The cognitive weights for these three BCSs are determined as under:
  - BCS1 (sequence): \( W_1 = 1 \)
  - BCS2 (iteration): \( W_2 = 2 * 3 * 2 = 6 \)
  - Total cognitive weight for program P1 is: \( 1 + 6 = 7 \).
- For program P2, there are two internal structures: iteration BCS and embedded component. The cognitive weight for Program P2 is determined as under:
  - BCS1 (iteration): \( W_1 = 3 + 7 = 10 \)
  - For program P1, there are four internal structures: a sequence, two iterations and a function call BCSs.

The cognitive weights and CFS for program P1 is calculated as under:

- BCS1 (sequence): \( W_1 = 1 \)
- BCS2 (iteration): \( W_2 = 3 \)
- BCS3 (function call): \( W_3 = 2 + 10 = 12 \)
- BCS4 (iteration): \( W_4 = 3 \)
- Total cognitive weight for program P1 is:
  \[ 1 + 3 + 12 + 3 = 19 \]
  \[ CFS = (N_i + N_o) \times 19 = (2 + 1) \times 19 \]
  \[ = 3 \times 19 = 57 \text{ [CWU]} \]

The above result of CFS shows that when both the internal architectural complexity and I/O turnover are considered, this program’s complexity is equivalent to 57 CWU.

**Calculation of NWMC**

For Program P3, \( N_{lp} = 3, N_{fp} = 3, \)
\[ N_{parameters} = 3 + 3 = 6 \]
\[ NWMC (P3) = 6 \times W_i = 6 \times 7 = 42 \text{ [CWU]} \]

For Program P2, \( N_{lp} = 3, N_{fp} = 2, \)
\[ N_{parameters} = 3 + 2 = 5 \]
\[ NWMC (P2) = 5 \times W_i = 5 \times 45 = 225 \text{ [CWU]} \]

For Program P1, \( N_i = 2, N_o = 1, N_{lp} = 4 \)
\[ N_{parameters} = 7 \]
\[ NWMC (P1) = 7 \times (1 + 3 + 227 + 3) \]
\[ = 7 \times 234 = 1872 \text{ [CWU]} \]

The above result of NWMC shows that when both the internal architectural complexity and \( N_{parameters} \) turnover are considered, this program’s complexity is equivalent to 1872 CWU.

It is observed from above solved example that NWMC gives better prediction for the program’s complexity as compared to CFS. Because the CFS considered only input, output and BCS, but internal complexity is more than this. So, in proposed metric different internal parameters are considered. That’s why the value of both measures are different. Now the development time of this example is calculated by the constant \( a, b \).
Development time of Fig. 1 with NWMC:
DT = 0.9793 + (0.0964 * 1872) = 181 Minutes
Development time of Fig. 1 with CFS:
DT = 0.0004 + (0.7454 * 57) = 42 Minutes

The development time of Fig. 1 with NWMC and CFS is shown above, it is clearly indicated that the difference is around 5 times greater than CFS. So to understand the problem, design, code, and test the program, the 42 minutes are very less for the beginners who do not have any knowledge about problem, but they have programming skills. So to develop the code like Fig. 1 takes three hours approximately. But the 42 minutes are very less to develop such complex code, since the NWMC predict the development time better than CFS.

The UA factor of Fig. 1 is calculated as follows:
UA = (1872 ** 0.48) * 0.62 = 23 minutes
So, according to the observation 23 minutes has been required to understand the code.

### Table 2. Analysis of the NWMC, CFS and LOC

<table>
<thead>
<tr>
<th>Program No.</th>
<th>CFS</th>
<th>NWMC</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33</td>
<td>44</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>24</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>28</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>88</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>75</td>
<td>200</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>65</td>
<td>169</td>
<td>33</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>200</td>
<td>34</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>120</td>
<td>35</td>
</tr>
<tr>
<td>9</td>
<td>57</td>
<td>413</td>
<td>36</td>
</tr>
<tr>
<td>10</td>
<td>76</td>
<td>190</td>
<td>37</td>
</tr>
<tr>
<td>11</td>
<td>34</td>
<td>205</td>
<td>40</td>
</tr>
<tr>
<td>12</td>
<td>40</td>
<td>140</td>
<td>44</td>
</tr>
<tr>
<td>13</td>
<td>66</td>
<td>420</td>
<td>46</td>
</tr>
<tr>
<td>14</td>
<td>280</td>
<td>820</td>
<td>60</td>
</tr>
<tr>
<td>15</td>
<td>84</td>
<td>128</td>
<td>60</td>
</tr>
<tr>
<td>16</td>
<td>136</td>
<td>708</td>
<td>61</td>
</tr>
<tr>
<td>17</td>
<td>569</td>
<td>697</td>
<td>65</td>
</tr>
<tr>
<td>18</td>
<td>174</td>
<td>5859</td>
<td>66</td>
</tr>
<tr>
<td>19</td>
<td>99</td>
<td>704</td>
<td>71</td>
</tr>
<tr>
<td>20</td>
<td>312</td>
<td>4664</td>
<td>159</td>
</tr>
</tbody>
</table>

---

### Fig. 1. A Program to sort an array using selection sort
```c
#include<stdio.h>
#include<conio.h>

int smallest(int arr[], int k, int n);

void selection_sort(int arr[], int n);

void main()
{
    clrscr();
    printf("Enter the number of elements in the array);
    scanf("%d", &n); //BCS1
    printf("Enter the elements of the array
                arr[%d] = ", i);
    scanf("%d", &arr[i]);
    selection_sort(arr, n); //BCS2
    printf("Sorted Array is:
                %d
    " ), arr[i]);
    getch();
}
```

### Fig. 2. Analysis of NWMC and CFS for 20 Programs

---

### Fig. 3. Analysis of CFS with LOC for 20 Programs
Measuring Complexity, Development Time and Understandability of a Program: A Cognitive Approach

Table 3. Average development time of 20 programs in Minutes

<table>
<thead>
<tr>
<th>Program No.</th>
<th>U 1</th>
<th>U 2</th>
<th>U 3</th>
<th>U 4</th>
<th>U 5</th>
<th>Avg. Time (Min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>9</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>6.2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3.6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4.2</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>16</td>
<td>9</td>
<td>12</td>
<td>8</td>
<td>11.8</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>28</td>
<td>22</td>
<td>21</td>
<td>23</td>
<td>23.8</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
<td>25</td>
<td>22</td>
<td>19</td>
<td>23</td>
<td>22.4</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>26</td>
<td>22</td>
<td>19</td>
<td>21</td>
<td>22.6</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>24</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>42</td>
<td>44</td>
<td>32</td>
<td>33</td>
<td>30</td>
<td>36.2</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>25</td>
<td>17</td>
<td>14</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>28</td>
<td>31</td>
<td>25</td>
<td>20</td>
<td>21</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>26</td>
<td>28</td>
<td>22</td>
<td>21</td>
<td>18</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>61</td>
<td>56</td>
<td>43</td>
<td>41</td>
<td>38</td>
<td>47.8</td>
</tr>
<tr>
<td>14</td>
<td>80</td>
<td>92</td>
<td>64</td>
<td>66</td>
<td>58</td>
<td>72</td>
</tr>
<tr>
<td>15</td>
<td>124</td>
<td>136</td>
<td>84</td>
<td>79</td>
<td>63</td>
<td>97.2</td>
</tr>
<tr>
<td>16</td>
<td>119</td>
<td>122</td>
<td>88</td>
<td>77</td>
<td>58</td>
<td>92.8</td>
</tr>
<tr>
<td>17</td>
<td>108</td>
<td>112</td>
<td>76</td>
<td>75</td>
<td>80</td>
<td>90.2</td>
</tr>
<tr>
<td>18</td>
<td>448</td>
<td>489</td>
<td>386</td>
<td>342</td>
<td>358</td>
<td>404.6</td>
</tr>
<tr>
<td>19</td>
<td>117</td>
<td>126</td>
<td>85</td>
<td>82</td>
<td>72</td>
<td>96.4</td>
</tr>
<tr>
<td>20</td>
<td>480</td>
<td>435</td>
<td>423</td>
<td>380</td>
<td>358</td>
<td>415.2</td>
</tr>
</tbody>
</table>

Table 4. Comparison of development time of CFS and NWMC with actual time

<table>
<thead>
<tr>
<th>S. No</th>
<th>CFS</th>
<th>NWMC</th>
<th>AE</th>
<th>UA</th>
<th>MRE of CFS</th>
<th>MRE of NWMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td>5</td>
<td>6</td>
<td>3.8</td>
<td>-297</td>
<td>15.8</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>3</td>
<td>4</td>
<td>2.9</td>
<td>-397</td>
<td>8.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3.1</td>
<td>-42</td>
<td>12.4</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>9</td>
<td>12</td>
<td>5.3</td>
<td>-13.7</td>
<td>19.8</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>20</td>
<td>24</td>
<td>7.9</td>
<td>-135</td>
<td>14.9</td>
</tr>
<tr>
<td>6</td>
<td>48</td>
<td>17</td>
<td>22</td>
<td>7.3</td>
<td>-116</td>
<td>22.9</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>20</td>
<td>23</td>
<td>7.9</td>
<td>-64.9</td>
<td>10.4</td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>13</td>
<td>20</td>
<td>6.2</td>
<td>-19.3</td>
<td>37.3</td>
</tr>
<tr>
<td>9</td>
<td>42</td>
<td>41</td>
<td>36</td>
<td>11.2</td>
<td>-17.4</td>
<td>-12.7</td>
</tr>
<tr>
<td>10</td>
<td>57</td>
<td>19</td>
<td>18</td>
<td>7.7</td>
<td>-215</td>
<td>-7.2</td>
</tr>
<tr>
<td>11</td>
<td>25</td>
<td>21</td>
<td>25</td>
<td>8</td>
<td>-1.4</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>30</td>
<td>15</td>
<td>23</td>
<td>6.7</td>
<td>-29.6</td>
<td>35.4</td>
</tr>
<tr>
<td>13</td>
<td>49</td>
<td>41</td>
<td>48</td>
<td>11.3</td>
<td>2.9</td>
<td>13.2</td>
</tr>
<tr>
<td>14</td>
<td>209</td>
<td>51</td>
<td>72</td>
<td>12.5</td>
<td>190</td>
<td>29</td>
</tr>
<tr>
<td>15</td>
<td>33</td>
<td>71</td>
<td>97</td>
<td>14.7</td>
<td>66.3</td>
<td>26.8</td>
</tr>
<tr>
<td>16</td>
<td>101</td>
<td>69</td>
<td>93</td>
<td>14.4</td>
<td>9.2</td>
<td>25.9</td>
</tr>
<tr>
<td>17</td>
<td>275</td>
<td>68</td>
<td>90</td>
<td>14.4</td>
<td>205</td>
<td>24.4</td>
</tr>
<tr>
<td>18</td>
<td>130</td>
<td>407</td>
<td>405</td>
<td>39.9</td>
<td>67.9</td>
<td>-0.5</td>
</tr>
<tr>
<td>19</td>
<td>74</td>
<td>69</td>
<td>96</td>
<td>14.4</td>
<td>23.4</td>
<td>28.6</td>
</tr>
<tr>
<td>20</td>
<td>233</td>
<td>451</td>
<td>415</td>
<td>35.8</td>
<td>44</td>
<td>-8.5</td>
</tr>
</tbody>
</table>

Fig. 4. Analysis of NWMC with LOC for 20 Programs

Fig. 5. A plot of actual and calculated development time

Fig. 6. MRE of CFS and NWMC with Actual time

Fig. 7. Correlation of CFS and NWMC with actual time
IV. COMPARATIVE STUDY OF NWMC AND CFS

This section presents the analysis of NWMC, CFS, lines of codes (LOC) of 20 programs and also focus on measuring the development time of 20 programs and correlates with actual development time and find the MRE and Standard Deviation of all programs. An attempt has also been made to calculate the UA of a program with the help of proposed measure by using an empirical formula. The actual development time is observed from five student of the institute and their mean is considered as actual time because the experience of five students is different with programming language. Each program is analysed in terms of code (LOC), CFS and NWMC (with the unit of cognitive weight or CWU) as shown in Table 2 and illustrated in Fig. 2, Fig. 3 and Fig. 4.

The LOC of software, or program length, can be used as a predictor of program characteristics such as effort and difficulty of maintenance. However, it characterizes only one specific aspect of size, namely the static length, because it takes no account of functionality. The CFS and NWMC is concerned with functional and cognitive complexity.

Following observations made from Table 2, Table 3, Table 4, Fig. 2, Fig. 3 and Fig. 4, Fig. 5, Fig. 6, Fig. 7 which are as follows:

- From Fig. 2, it is observed that for each program NWMC gives better cognitive weight measurement for complexity value calculation than CFS. Because CFS has only takes input, output and weight of BCS in account, but the internal data elements are also part of complexity that are excluded by the CFS, but the proposed metric consider these factors.

- Certain interesting observation made from Table 2, Fig. 3 and Fig. 4. From Table 2 it is observed that, the trends for LOC, CFS and NWMC follow basically the same pattern. As the LOC increases, so does the corresponding CFS and NWMC. It is noteworthy that there are four points for which the CFS and NWMC grows-up sharply (see bold italic mentioned for CFS, NWMC and LOC in Table 2). This indicates that these four programs have higher code cognitive complexity, using fewer lines of code to implement more complex program. As indicated in Fig. 3 and Fig. 4, LOC is not a good measure for predicting program complexity. As mentioned in Table 2, there are two cases with similar LOC first case is around 60/61 LOC (painted with Gray) but one has CFS = 44 CWU below, NWMC =728 CWU above, while other is CFS= 136 CWU above, NWMC = 708 CWU below and in the second case is around 65/66 LOC (painted with Dark Yellow) but one has CFS = 369 CWU above, NWMC =697 CWU below, while other is CFS= 174 CWU below, NWMC = 5859 CWU above. Hence the proposed metric shows the accurate complexity of programs than others measures.

- Table 3 shows the development time of 20 programs with time taken by five different students of our institute. The development time of all five students is mostly different from others, this is due to their programming skills. Some students are very experienced and some are less experienced. The average time of all five students are calculated in the last column that is considered as the actual time in minutes that is required to develop the given program. The unit of the measuring developing time is in Minutes.

- The Table 4 describes the development time of 20 programs which is calculated with the help of NWMC and CFS and a comparison is also done with the actual time which is measured from 5 PG students. The development time of all 20 programs are calculated with the regression, the authors found the two constant a, b and the cognitive weight of individual. The value of a, and b are shown in the section 3.1 for both of measures. The MRE of both the techniques NWMC and CFS are calculated that is shown in the last two column of the Table 4. The MRE of the NWMC is very close to 0, means that the difference between the actual development time and the calculated time in very less, but MRE of CFS shown very large value in both the sides (in +ve side and –ve side) means the calculation is not much accurate. The result of this study shows that the development time is measured by the NWMC is very close to the actual development time than the CFS, this can be observed from mean of MRE (indicated in yellow colour) of both measures, CFS overestimate the development time that’s why the mean of MRE is greater than proposed measure. The standard deviation of both the measure is shown in the last row of the Table 4, CFS also does not do well with standard deviation the average standard deviation is 125.74 with the smallest error being 2.9. But the standard deviation of proposed measure has 24.23 with 0.5 smallest error (indicated in green colour) and the small value is desirable. So the proposed measure the development time better than CFS. UA is also calculated that is very important factor to test and maintain the software. The UA is helpful when the code is need to understand, because the maintaining the code is difficult than developing. Clock Time required to understand the code, but how much, this is question is solved by the understandability factor shown in Table 4. By using the Table 4 the Fig. 5 and Fig. 6 are generated.

- Fig. 7 represents the Pearson and Spearman’s rho Correlation of NWMC and CFS with actual time of 20 Programs. (** Correlation is significant at the 0.01 level (2-tailed).). It can be seen that NWMC has a better correlation with actual time than CFS. These firmly believe that NWMC is better cognitive complexity metric that can be used in the real world.

Therefore, the NWMC and cognitive weights provide an objective, logical, and comparative measure for quantitatively analyzing and predicting program complexity and development time and understandability of software in software engineering.
V. CONCLUSION AND FUTURE WORK

A complexity measure based on cognitive weight and a formula to calculate the development time and understandability of programs is proposed. It is found that cognitive weight complexity measure is the most suitable measure, when it is compared with other similar measures. Software complexity measures serve both as an analyzer and a predictor in quantitative software engineering. This paper has developed the New Weighted Method Complexity (NWMC) on the basis of cognitive weights, permitting determination of program complexity from cognitive aspects. Cognitive weights for basic control structures (BCSs) have been introduced to measure the complexity of logical structures of program. A large set of case studies has been carried out to analyze the relationship between NWMC with CFS and LOC of a program. The NWMC has been shown to be a fundamental measure of program complexity based on the cognitive weight. The development time of a program is calculated with the help of cognitive complexity that is measured by NWMC and CFS. The result shows that the proposed technique has better capability to measure the complexity and development time of a program than the CFS.

This work has produced four substantial findings:

(a) The NWMC of software in design and comprehension is dependent on five factors: internal processing structures, as well as the number of inputs, number of local parameters, number of formal parameters used during call and number of outputs.

(b) The NWMC is more robust than the CFS and LOC measure and independent of language/implementation.

(c) The development time help to better estimate the cost and the deadline required for releasing the software and ensuring that the risk has been reduced.

(d) Understandability factor help to maintain and test the software, and can also be used at early stage of the software life cycle to reduce the complexity that increase the Understandability ratio.

The future work includes on some fundamental issues:

(1) Programs used for the study were very small as compared to large system. The same will be carried out with large system along with some other cognitive measures to see the effect on program complexity.

(2) The proposed measure needs further verification with some software engineering measurement (as for example, Weyuker Properties, Briand’s Property) for analyzing the nature of the metric.

(3) Furthermore, the time required to find and fix the bugs can be observed with the help of UA factor.

REFERENCES


Authors’ Profiles

Mr Amit Kumar Jakhar: PhD. student in the Department of Computer Science & Engineering, at Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India. His research area is software
Mr Kumar Rajnish: Assistant Professor in the Department of Computer Science & Engineering, at Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India. He received his PhD in Engineering from BIT Mesra, Ranchi, Jharkhand, India. His Research area is Object-Oriented Metrics, Object-Oriented Software Engineering, Software Quality Metrics, Programming Languages, and Software Estimation.