Software Complexity for Computer Communication and Sensor Networks Using Binary Decision Diagrams

Harpreet Singh, Adam Mustapha, Arati M Dixit, Kuldip Singh, Grant R. Gerhart

Abstract—Software complexity has been a topic of interest for software engineers for the last three decades. Various definitions of software complexity have appeared in literature from time to time. This has resulted in the development of software metrics by different investigators. With the new developments in the area of computer communication and sensor networks, software complexity metrics need an improvement so that they can be used for internet, computer communication networks, and sensor networks. It is difficult to apply the existing definitions of software complexity to internet applications. The objective of the present paper is to propose a definition which could be used for internet and sensor networks applications. In this paper a new definition of software complexity is utilized for the proposed applications. It is suggested to use Binary Decision Diagrams for this purpose as these diagrams result in disjoint expression which is needed in the calculation of the software complexity. Simple cases of software complexity for series, parallel, series-parallel and non series-parallel networks are investigated. The simulation of the proposed algorithm is given in this paper. The complexity between two nodes of a graph can be determined with the help of path expressions between the two nodes of interest of a graph. Binary Decision Diagram approach is then utilized to determine the complexity between two terminal nodes of a graph. The approach is applicable to any number of nodes and branches of a graph. It is hoped that this approach will be extensively used by software engineers who are developing software for internet, computer communication networks, and sensor networks.

Index Terms—Software complexity, software reliability, fuzzy logic, binary decision diagrams, boolean algebra, MATLAB.

I. INTRODUCTION

Software complexity is a very important metric used in software engineering. While developing software it is important to predict the complexity of software in advance. If the complexity is more than the desired complexity, changes can be made in the development process to obtain the prescribed complexity. It is important to know this metric in the early stages of the software development. Knowing complexity at later stages results in high development costs of the software. Because of these reasons there has been ongoing research in determining various definitions of complexity. With increasing popularity of internet, the complexity definitions have to be modified so as to meet the increasing demand of software development for internet applications. The existing definitions of complexity are not suitable for internet application. In this paper, a new definition of software complexity is suggested so that this definition can be applied to internet software development [1]. Here in the software complexity is defined from zero to one rather than the previous definition of number of lines of code, number of operators and operands, number of loops, etc. Halstead [2], [28] defined software complexity in terms operators and operands. McCabe [3] defined the software complexity in terms of the number of loops in the software. This was followed by a number of other workers [4]-[6] who gave different definitions of software complexity. Table I has various definitions of software complexity.

In order to define software complexity between zero and one, we suggest Boolean algebra approach here. We essentially suggest the same approach which has been used by several investigators in determining the reliability of computer communication network [7]-[16]. The same approach can be useful for internet and
sensor network applications, so long as the problem can be represented as a graph, these approaches can be applied. In this paper we suggest Binary Decision Diagrams approach to determine the complexity of sensor network. Binary Decision Diagrams became popular when the conventional truth table approaches could not be applied for a large number of variables. With the coming of internet and sensor network, a number of nodes and branches of under consideration have become extremely large. Because of the fact that Binary Decision Diagrams approach can be used for a large number of variables, hence such an approach is suggested in the present work.

### TABLE 1 SOFTWARE COMPLEXITY

<table>
<thead>
<tr>
<th>Authors/Year</th>
<th>Software Complexity</th>
<th>In Terms of</th>
</tr>
</thead>
<tbody>
<tr>
<td>McCabe 1976</td>
<td>Cyclomatic complexity</td>
<td>Loops</td>
</tr>
<tr>
<td>Halstead 1977</td>
<td>Length of program and Effort</td>
<td>Operands and Operators</td>
</tr>
<tr>
<td>Ramamorthy and Melton 1988</td>
<td>Family of measures</td>
<td>Weighted measures</td>
</tr>
<tr>
<td>Lisa Anneberg 1991</td>
<td>Patri net 'A' matrix complexity</td>
<td>Order of 'A' matrix</td>
</tr>
<tr>
<td>Munson and Khoshgoftaar 1992</td>
<td>Relative complexity</td>
<td>Factor Analysis</td>
</tr>
</tbody>
</table>

### II. A BRIEF REVIEW OF BINARY DECISION DIAGRAMS

Binary Decision Diagrams (BDD) [16]-[19] have become popular for the last three decades. The importance of Binary Decision Diagram was observed especially with the development of new techniques for finding faults for very large VLSI circuits. However, these diagrams have attracted the attention of investigators in different fields. In this paper we develop a procedure so as to illustrate how Binary Decision Diagrams can be useful in determining the software complexity for computer communication networks. Complexity using Binary Decision Diagrams will be of interest to logic circuit fault tolerant and VLSI design community. These diagrams are useful for both combination circuits and sequential circuits.

### III. REVIEW OF SOFTWARE COMPLEXITY OF SIMPLE NETWORKS

The software complexity of circuits has been studied for the last four decades extensively in the literature [2]-[6], [10], [20]-[22]. The software complexity of branches in series, branches in parallel, and branches in non-series parallel has been the subject of research papers [1].

Currently, sensors networks have become very popular in literature because of their applications in different areas. The purpose of this paper is to utilize how Binary Decision Diagrams can be useful in determining the software complexity for computer communication and sensors networks. For completeness, a review of Binary Decision Diagrams is briefly given first. Fig. 2 illustrates the software complexity of paths in series, in parallel, in series-parallel and non-series parallel. Both paths and binary decision diagrams for each case are given.

More complex software is less reliable and vice versa. The software reliability has a range of zero to one; hence software complexity is also defined as to have a range of zero to one. Further, these definitions are extended so that the complexity of interconnected network can be defined.

Consider two branches in series having software complexity as 0.9 and 0.8. The uncomplexity (reliability) of these branches will be 0.1 and 0.2 respectively. The reliability (uncomplexity) of series path will be equal to 0.1* 0.2 = 0.02. Hence the complexity will be of the series path will be 1-0.02 = 0.98.

Next consider two parallel paths having complexity 0.9 and 0.8. Their uncomplexity (reliability) will be 0.1 and 0.2 respectively. So the reliability (uncomplexity) of the two paths in parallel will be equal to 0.1 + 0.2 – 0.1* 0.2 which will be equal to 0.28. Hence the software complexity of two parallel paths is 1- 0.28 = 0.72.

Next consider two series parallel paths (x1 and x2 in parallel with x3 and x4) having complexities equal to 0.9 and 0.8 for the series path and 0.7 and 0.6 for the other path in parallel. The complexity of the series path will be 0.98 and 0.88. The
uncomplexity of the two parallel paths are 0.2 and 0.12 respectively. The overall uncomplexity is 0.02 + 0.12 – 0.02 * 0.12 = 0.1376. Hence the overall series parallel complexity will be equal to 1 – 0.1376 = 0.8624.

We cannot find out complexity of non-series parallel network by simple calculation. However by using Boolean Algebra technique and by finding a disjoint (non-overlapping) expression one can determine the terminal complexity of the non-series parallel network as 0.835.

The terminal software complexity value of each network is given in a Table 2. These values can be easily determined with the help of conventional techniques given in reference [1]. However, in this paper these values can be determined through Binary Decision Diagrams and with the software developed using Fuzzy Logic [24]-[26] MATLAB toolbox [27].

<table>
<thead>
<tr>
<th>Graph</th>
<th>Terminal software complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 \rightarrow c_2 )</td>
<td>0.98</td>
</tr>
<tr>
<td>( c_1 \rightarrow c_2 \rightarrow )</td>
<td>0.72</td>
</tr>
<tr>
<td>( c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow c_4 )</td>
<td>0.8624</td>
</tr>
<tr>
<td>( c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow c_4 )</td>
<td>0.835</td>
</tr>
</tbody>
</table>
III. TERMINAL SOFTWARE COMPLEXITY (TSC) USING BINARY DECISION DIAGRAMS (BDD)

The terminal software complexity is the software complexity between two terminal nodes. Given a network, it is important to determine branch software complexity between various nodes of a network. The problem is a general one and has applications in several areas such as communication systems and computer programs. Knowing the complexity of each branch, one can determine the terminal software complexity between any two nodes. The Boolean algebra approach has been used by several investigators [10], [14], in determining the terminal reliability of a network. The Boolean algebra technique basically consists of the following steps:

1. Determine simple paths between two nodes of a graph.
2. Write down the Boolean expression corresponding to the paths where the Boolean variables correspond to the different branches.
3. Determine a disjoint expression corresponding to the Boolean expression given in step 2.
4. Given the disjoint Boolean expression, substitute the corresponding values of branch software uncomplexity.
5. The terminal software complexity = 1 - uncomplexity

V. ALGORITHM FOR TERMINAL SOFTWARE COMPLEXITY USING BINARY DECISION DIAGRAMS

The following steps are used for drawing Binary decision Diagrams.
1. Determine simple paths.
2. Determine the Boolean expressions which correspond to the simple paths.
3. Use Shannon’s expansion theorem to mark the paths.
4. Determine the non-overlapping expressions.
5. Substitute branch software uncomplexity values
6. The terminal software complexity = 1 - uncomplexity.

VI. SOFTWARE IMPLEMENTATION

We have developed software which can determine the software complexity between any two terminal nodes of graph. The approach is applicable for a graph having any number of nodes and branches. The software has been developed based on the concept of Binary Decision Diagrams.

The form below in Fig. 3 is the starting form of the application. This form allows the user to enter
- number of wires
- number of nodes
- number of bidirectional wires
- begin node of the desired path
- end node of the desired path

Desired path is the path for which the complexity is to be calculated.

Fig. 3 shows MATLAB displays for terminal complexity for series parallel network with 4 nodes 4 wires example.
VI. CONCLUSION

In this paper a new technique defining software complexity has been proposed. Contrary to the previous definitions, we suggest a new definition of software complexity where the values of complexity lie between zero and one. Such definition helps in determining the software complexity for internet application. This approach how to determine complexity requires disjoint Boolean expression. In this paper BDD are suggested to obtain disjoint or non-overlapping expression. The proposed procedure has been implemented using MATLAB. The procedure is applicable for any number of nodes. It is hoped that this approach of determining software complexity will be utilized by several investigators who will be developing software for internet and sensor applications. Determining software complexity at the earlier stages of the software development goes a long way in reducing the overall cost of software development.

REFERENCES


