Complexity Metrics for Component Based Software — A Comparative Study

Sonal Gehlot¹, PoojaRana¹, Rajender Singh²*
¹Maharshi Dayanand University, Rohtak, Haryana, India.
²Department of Computer Science & Application, Maharshi Dayanand University, Rohtak, Haryana, India.

*Corresponding author: Tel. 08199023624, email: chhillar02@gmail.com
Manuscript submitted March 10, 2019; accepted May 20, 2019.

Abstract: The Component Based Software Development (CBSD) approach is becoming the trend for software development and is based on developing the software from existing components instead of developing software from the scratch level. Measuring the software complexity is an important aspect during software development as it is an important determinant of software development effort, testing effort, cost, maintainability etc. Interactions/interfaces among components play an important role in contributing complexity to a component based software. In this paper a comparison between different complexity metrics developed by different authors is performed. These metrics are performed by taking different factors to calculate the complexity of the components based software, these factors are instance variables, instance methods, control flow and interface methods etc. The comparison is performed by taking some quality factors into consideration like maintainability, Integrity, complexity, testability, customizability etc.

Key words: CBAD, coupling, weighted assignment technique, complexity metrics, cyclometric complexity, and black box component.

1. Introduction

Software metrics play a very important role in assessing and predicting various attributes of software such as complexity, reusability, maintainability, testability etc. Among these attributes complexity affects all other attributes of the software [1]. Software Complexity measures have great importance because it indicates scope of further improvement in software development. Higher value of complexity increases efforts of testing, maintenance and also difficult to reuse.

The component based software development (CBAD) is one of the most important paradigms. CBAD approach is increasingly being adopted for software development. This approach uses reusable components as building blocks for constructing software application. The main aim of this approach is to minimize the development cost, time and efforts by mean of reuse [1].

The major problem faced in Component development is its complexity. So it is necessary to measure the software complexity and reduced it to achieve the maximum benefits of CBAD with minimum cost and efforts. There are several metrics which are available for measuring software complexity but they are not suitable for CBAD.

Software complexity cannot be removed completely but can be controlled only. For controlling of complexity, from time to time many researchers have proposed various metrics for evaluating, predicting and controlling software complexity. Traditional software metrics are usually applicable to small programs,
whereas the metrics for object-oriented and component based software applications should depend mainly on the granularity and interoperability aspects of the classes and components. The major factor influencing the CBAD is the dependency among software components, which is necessary and desirable because one component may provide the services to another component. Then there should be an interface between the components.

The following figure (Fig. 1) shows the technique for developing software application from existing components (cp).

![Components Shelf](image)

**Fig. 1.** Component based software development technique.

The paper is divided into sections. First section is introduction; second section has comparison of different complexity metrics. Section 3 has key observation and Section 4 concludes the paper.

### 2. Complexity Metrics for Component Based Application

The dependency among components may be defined as the reliance of a component on others to support a specific functionality or configuration. In CBSE application the components interact with other components by sharing information in order to provide application functionalities. This composition creates interaction that promotes dependencies among components. Application functionalities cannot solely encapsulate within one component. Therefore changing a component may affect that composite functionality, which is reflected in different components. In addition, replacing a new version of a specific component might involve replacing the component on which it depends, in order to preserve a specific application’s functionality.

The component complexity closely depends on what contributes to develop the components. Thus there are four elements that affect the component complexity. First element is Variable factor that tells complexity of the variables defined in the component. The variables may consist of member variables of a class having scope for the entire class and the parameters, which are local to a particular method. The second element is interfaces, which are the access points of component, through which a component can request a service declared in an interface of the service providing component.

Interface complexity is defined as sum of complexity of the interface methods of the class. Third element is coupling factor that tells rate of coupling of the methods in the component. Fourth element is cyclometric complexity of the methods of the component.

In this paper we will review the different types of complexity metrics and compare them.

**Nael Salman et al. [2] 2006** author in this paper developed several complexity metrics for component based system. The main focus of the author is to find out the strength of the software, by defining metrics on its structural complexity [2]. The main factors that determine the complexity are components, connectors and composition tree.
For components three metrics are defined such as TNC, ANMC and TNIC, for connectors three metrics are
developed like TNL, ANLC, ANLI [2]. the author defined a suite of metrics for component based software and the
definition of component based software is adopted by author which is given by Szyperski (1999) [2].

Gill and Balkishan et al. [3] 2008 their attempt was to identify the impact of dependency among components
on software complexity. They proposed two metrics CDM and CIDM. Both the metrics can be applied once the
directed graph and adjacency matrix of the design is found [3]. The result of these metrics is analyzed in order to
determine as how the interaction among components and number of components affect the complexity of
component based application. These metric shows higher interaction between components increases the
complexity because of more coupling among component. Higher complexity means more expensive software and
less maintainability [3].

Gui Gui and Paul. D. Scott et al. [4] 2009 in this paper the author develop new metrics for coupling and
cohesion to measure the reusability of a component. These metrics are different from other metrics in three aspects,
first is degree of resemblance of one component with other component, degree of coupling, direct coupling and
cohesion relationship. Author measures quantitatively measure the complexity by taking these factors into
consideration [4]. A comparative analysis is performed between new metrics and existing metrics and found that
new metrics are much superior then existing metrics [4].

Narasimhan and Hendradiya’s metric suite et al. [5] 2009 in this paper author defined a metric suite for
component based software. Author defined two sets of metrics for measuring complexity and criticality. First set
of metrics are Component Packing Density (CPD) which describe the binding of the components. These metrics
relate all component constituents to the number of integrated component. Another set of metrics are Component
Interaction Density (CID), these metrics relates interaction between components and available number of
interaction in the system [5].

Kumari and Bhasin et al. [6] 2011 the authors aimed to design a composite complexity measure to quantify
important aspects of complexity of a component based application. Two major complexity metric of CBS are one
due to individual component named as TC (CBS) and other due to its interaction with other component named as
IACC [6]. They take different factors like size, type of variables, nesting level of control structure to calculate the
individual component complexity. Graph theoretic notions and concept of weights have been used for interface
component complexity. The result shows that the effect of these parameters (Size, Nesting Structure, Control
Structure etc.) on complexity of a CBS is quite significant. The results also show that higher interaction between
components increases the complexity because of more coupling among the components [6].

<table>
<thead>
<tr>
<th>Sr.No</th>
<th>Metric</th>
<th>Formula Used</th>
<th>Description</th>
<th>Author/s</th>
<th>External Quality Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Average no of methods per component (ANMC) [2]</td>
<td>ANMC=total no of methods/ total no of component [2]</td>
<td>This metrics is estimated by dividing the total number of methods by the total number of components [2].</td>
<td>SALMAN et.al. (2006) [2]</td>
<td>Integrability and maintainability</td>
</tr>
</tbody>
</table>

Table 1. Metrics of Different Authors and Comparisons
<table>
<thead>
<tr>
<th>Sr.No</th>
<th>Metric</th>
<th>Formula Used</th>
<th>Description</th>
<th>Author/s</th>
<th>External Quality Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.</td>
<td>ClassCohT = (\frac{\sum_{i,j=1}^{m} \text{Sim}(i,j)}{m^2 - m}) [4]</td>
<td>ClassCohT = (\frac{\sum_{i,j=1}^{m} \text{Sim}(i,j)}{m^2 - m}) [4]</td>
<td>Gui and Paul et.al. (2009) [4]</td>
<td>Complexity Reusability</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>ClassCohD = (\frac{\sum_{i,j=1}^{m} \text{Sim}(i,j)}{m^2 - m}) [4]</td>
<td>ClassCohD = (\frac{\sum_{i,j=1}^{m} \text{Sim}(i,j)}{m^2 - m}) [4]</td>
<td>Gui and Paul et.al. (2009) [4]</td>
<td>Complexity Reusability</td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Packing Density [5]</td>
<td>CPD(_{\text{constituent type}}) #\text{Constituent} \text{# Component} [5]</td>
<td>It is a ratio between Number of constituents and Number of components [5].</td>
<td>Narasimhan et.al. (2009) [5]</td>
<td>Complexity Criticality</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Incoming Interaction Density of Component [5]</td>
<td>IIDC= # I_{IN} \text{# } I_{\text{max}}[IN] [5]</td>
<td>It is a ratio between number of incoming interaction used and available number of incoming interaction [5].</td>
<td>Narasimhan et.al. (2009) [5]</td>
<td>Complexity Interdependenc e Interface</td>
<td></td>
</tr>
<tr>
<td>Outgoing Interaction Density of Components [5]</td>
<td>OIDC= # I_{OUT} \text{# } I_{\text{max}}[OUT] [5]</td>
<td>It is a ratio between number of outgoing interaction used and available number of outgoing interaction [5].</td>
<td>Narasimhan et.al. (2009) [5]</td>
<td>Complexity Interdependenc e Interface</td>
<td></td>
</tr>
<tr>
<td>Total Complexity of a Component Based System (TC(CBS)) [6]</td>
<td>TC(CBS)= \sum_{j=1}^{n}((ICC)j * (Wj)) * (IACC)j [6]</td>
<td>This composite metric takes different attributes of complexity. The result shows the effect of these parameters on complexity of a CBS [6].</td>
<td>Kumari and Bhasin et.al. (2011) [6]</td>
<td>Complexity Testing Maintainability</td>
<td></td>
</tr>
<tr>
<td>Interaction Among Components Complexity (IACC) [6]</td>
<td>IACC= IIC+OIC [6]</td>
<td>This metric shows the interaction with other component. The concept of link is used to quantify interface aspect of a component [6].</td>
<td>Kumari and Bhasin et.al. (2011) [6]</td>
<td>Complexity Testing Maintainability</td>
<td></td>
</tr>
<tr>
<td>Average Incoming Interactions Complexity (AIIC) [7]</td>
<td>AIIC= \sum_{i=1}^{m} I_{II} \text{# m} [7]</td>
<td>This metric shows the average of the incoming interactions of one component [7].</td>
<td>Kumari and Upadhyaya et. al. (2011) [7]</td>
<td>Complexity Reliability</td>
<td></td>
</tr>
<tr>
<td>Average Outgoing Interactions Complexity (AOIC) [7]</td>
<td>AOIC= \sum_{i=1}^{m} O_{II} \text{# m} [7]</td>
<td>This metric shows the average of the outgoing interactions of one component [7].</td>
<td>Kumari and Upadhyaya et. al. (2011) [7]</td>
<td>Complexity Reliability</td>
<td></td>
</tr>
<tr>
<td>Average Interface Complexity of a Component Based System (AIC(CBS)) [7]</td>
<td>AIC(CBS)= \sum_{i=1}^{m} I_{II} + \sum_{i=1}^{m} O_{II} \text{# m} [7]</td>
<td>This Metric shows the average interface metric by summation of incoming interface and outgoing interface metrics [7].</td>
<td>Kumari and Upadhyaya et.al. (2011) [7]</td>
<td>Complexity Reliability</td>
<td></td>
</tr>
<tr>
<td>Complexity of Interface Component (CI) [8]</td>
<td>CI= CSI+ICC(\text{internal})+ICC(\text{external}) [8]</td>
<td>This metric measured in term of size of interface, interface coupling with internal subcomponents and interface coupling with outer components [8].</td>
<td>Chilla r and Ahlawat et.al. (2012) [8]</td>
<td>Complexity Reusability Customizability</td>
<td></td>
</tr>
<tr>
<td>Average Complexity of CBS [8]</td>
<td>AC(CBS)= \sum_{i=1}^{n} CI/i [8]</td>
<td>This complexity measured the average complexity of interface component [8].</td>
<td>Chilla r and Ahlawat et. al. (2012) [8]</td>
<td>Complexity Reusability Customizability</td>
<td></td>
</tr>
</tbody>
</table>
Kumari and Upadhyaya et al. [7] 2011 they attempted to design an interface complexity metric for black box components to quantify an important aspect of complexity of a CBS. In CBAD a component is linked with other component and has interfaces with them. A link means that a component submits an event and other component receive it. Interface between two components can be through incoming and outgoing interactions. They proposed AIIC, AOIC and AIC (CBS) which calculate average incoming interaction complexity, average outgoing interactions complexity and average interface complexity of a component based application [7]. The result shows that the effect of interface parameter on complexity of CBS is quite significant. They propose that average no of interaction per component in CBS should not be greater than five otherwise that CBS would be highly complex and will be more prone to errors and hence unreliable [7].

Chillar and Ahlawat et al. [8] (2012) they proposed two metrics CI and IDM for measuring complexity of interface and interface dependency of CBS. These metrics are based on different constituents of an interface like interface methods and interface variables with different weights assigned to them. Strength of proposed metrics is computed using weighted assignment technique. Interface methods are classified according to data type of return type and data type of arguments [8]. These metrics shows that higher dependency among components increases complexity because of more coupling. There is a positive relation between complexity of interface metric and interface dependency metric. It is clear that complexity of interface and dependency of interface increases with increase in parameter involved [8].

### 3. Key Observations

Conducted a systematic study of the literature available for the interface metrics for component based applications. These included research publications involving validations, proposals and all other studies related to interface metrics [9]. The search for relevant publications was conducted through various ACM journals. Reference checking was used to make sure that no relevant work was being left out [9]. The contribution of different interface metrics based on the mapping level addressed by them was also studied, which revealed that most of the interface metrics proposed to measure the interdependence of different component using interface component [9].

Further analysis shows that most of the interface metrics address the maintainability, reusability and testability quality factors. Hence a lot of work is still left to be done to prove these metrics are good indicators of the overall software quality.

### 4. Conclusion and Future Scope of Work

This paper provides a thorough survey of interface metrics for Component Based application. The survey conducted covers all the aspects of Interface Metrics for CBS and presents them in categorical form. From the study it was observed that only a limited amount of work has been done in the field [9]. The other main findings along with the possible future directions are.
Most of the interface metrics studied lack of validations which limits their usefulness.
The relationship of these metrics with external quality attributes was also studied and it was found that most of the interface metrics proposed share a relationship with maintainability, testability and reusability.
Interface Metrics need to be evaluated for a wide range of large scale real world applications for both metric validation and effective utilization for software quality assessment.

The overall study revealed that the interface complexity metrics domain is still has a scope in the field of software engineering and faces a number of research challenges in term of empirical validation and relationship with external software quality attributes. For the future research work, researchers can use these metrics for indirect coupling measure as well as indicator for predicting the various qualities attributes like maintainability, testability and reusability of component based software application.

References

**Sonal Gahlot** is an assistant professor in DPG College of Engineering since April 2016. She has completed her B.E in computer science and engineering from Gurgaon Institute of Technology and Management, Gurgaon in 2009. Tech in computer science and technology from ITM University with First Division in 2011, completed pre-Ph.D course work from MDU, Rohtak with first division in August 2013 and pursuing Ph.D from MDU, Rohtak. Her areas of interest are software engineering, component-based software engineering, data structures, computer networks.

**Pooja Rana** is an associate professor in Department of Computer Science and Applications, DAV Institute of Management, Faridabad, Haryana, India; and pursuing Ph.D in computer science from Faculty of Computer Science and Applications, M.D. University, Rohtak, Haryana, India. She holds a master of computer application (MCA) from M.D. University,
Rohtak, India. She obtained her master of technology degree in Information Technology from AAI-DU, Allahabad, India.

Her research interests include software process reengineering, software engineering, software reuse, software process customization and automation, and software process metrics. She has presented and published various papers in international and national journals/conferences.

Rajender Singh Chhillar is working as a professor and head of Department of Computer Science and Applications, Maharshi Dayanand University (MDU), Rohtak, Haryana, India. He was the member of monitoring committee of campus wide Networking, M. D. University, Rohtak. He obtained his Ph.D in computer science from Maharishi Dayanand University, Rohtak and master degree from Kurukshetra University, Kurukshetra, Haryana.

Dr. Chhillar's research areas include software engineering, software testing, computer network security, software metrics, component and aspect based metrics, data warehousing and data mining, information and network security and IT management. He has published more than 150 publications in international and national journals/conferences. Professor Chhillar has also authored two books: *Software Engineering: Metrics, Testing and Faults*, Excel Books House, New Delhi, *Application of Information Technology to Business*, Ramesh Books House, Jaipur.