# Estimation of Software Complexity in Component based System

Tarun Ahuja
Research Scholar
Chaudhary Devilal University
Sirsa

Dilbag Singh, PhD.
Reader
Computer Sc. and Engg. Deptt.
Chaudhary Devilal University
Sirsa

## ABSTRACT

Software complexity and software testing are interrelated and important aspects of software development. For testing issue we can relate software testing to measurable quantities like coupling, cohesion etc. in order to measure and understand complexity. Most of the applications are developed using existing source code. This code is named as component in component based software development. And the concept of using these components to develop software is called reusability. These components increase complexity of the software and can also decrease software quality. In this work a set of software metrics are taken that will check the interconnection between the software components and the application. How strong this relation defines the software complexity after using this software component. For this work three cases such as four components, six components and eight components having interconnection among them are taken. After applying software metrics on them it will be suggested that complexity increases when new components are added.

**Keywords –** Modularity, Complexity, Quality, Reusability, Component-based software system.

## 1. INTRODUCTION

Software engineering is the analysis, design, coding, implementation, verification, reviewing and maintenance of technical entities. Software engineering is not just producing software but means of producing them in most efficient, effective and controlled way. Software engineering is the use of sound engineering principles in order to obtain cost effective and reliable software that works efficiently on real machines. Software engineering is very vast term but software reusability has completely changed the view. Reusability not only makes the software development easier but also makes development process transparent. For making reusability happen we need software components. These components may be code, whole module etc. With the addition of these components whole life cycle of software development has changed.

The core of component-based software system is to reuse software components. The pressure for reducing software development life cycles and cost has led to an increasing interest in CBSS that not only facilitates the process of software development but also changes the ways to develop software applications. Most of the CBSS research has been inclined towards methods and approaches in the development and in comparison of software systems [11]. Some work tries to evaluate the complexity of tools used to create the software artifacts [1]. A very little work has been made for the development of measures/metrics that can be used to evaluate the complexity of components being developed, and the software quality using component integration [8]. The paper is organized as follows. Section 2 describes the Building system from components; Section 3 presents description of software reuse; Section 4 presents the proposed work; Section 5 presents the experimental result; Section 6 presents the conclusion.

## 2. COMPONENTS BASED SYSTEM

Component based software engineering is a paradigm that aims at constructing and designing system using a predefined set of software component. This assumption has several consequences for the system lifecycle. First, the development processes of component-based systems are separated from development processes of the components; the components should already been developed and possibly used in other products when the system development process starts. Second, a new separate process will appear: Finding and evaluating the components. Third, the activities in the processes will be different from the activities in non-component-based approach; for the system development the emphasis will be on finding the proper components and verifying them, and for the component development, reusability will be the main concern.

## 3. SOFTWARE REUSE

Software component reuse does not just indicate the reuse of application code but specification and designs[10]. The potential gains from reusing abstract product of development process such as specifications may be greater than those from reusing code components. The Objective of this paper is to estimate the software complexity on the basis of Software reuse effectively. Software reuse enables developers to leverage past accomplishments and facilitates significant improvements in software productivity and quality [5]. A critical problem in today's practice of software reuse is the lack of a standard process model which describes the necessary details to support reuse-based software development and evolution. Software reuse is counted as one of the biggest benefits of object-oriented programming, and indeed we have been reusing software libraries for decades [7]. However, reuse practices have mostly been ad hoc, and the potential benefits of software reuse are still rarely realized. One of the primary obstacles to the reuse of independently-developed binary components on the industrial level lies in that the existing component technologies do not clearly separate component assembly from component development [15].Software development with reuse is an approach which tries to maximize the reuse of existing software components. Benefit of this approach is that overall development costs of the software are decreased [10].

There is a difference in requirements and business ideas in these two cases and different approaches are necessary.

Components are built to be used and reused in many applications, some possibly not yet existing, in some possibly unforeseen way system development with components is focused on the identification of reusable entities and relations between them, beginning from the system requirements and from the availability of components already existing [2][3]. Much implementation effort in system development will no longer be necessary but the effort required in dealing with components; locating them, selecting those most appropriate, testing them, etc. will increase.

## 4. PROPOSED WORK

The software components are the main object that performs the code reusability. The proposed study aims to estimate degree of reusability on the basis of black-box components and component based systems. The work proposes and validates metrics for Interface complexity, of the system. The study uses methods and the properties of the component with its interfacing.

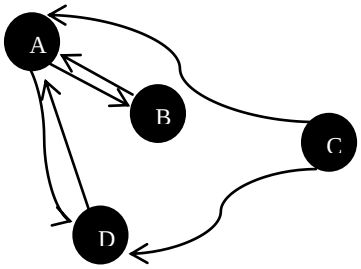


**Fig.1: Interaction Graph (i)**

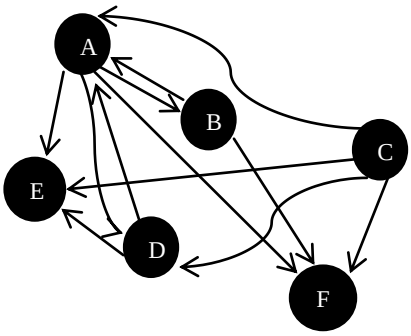


**Fig. 2: Interaction Graph(ii)**

Fig. 1 and fig. 2 shows the interaction between 4 and 6 components of any system is shown respectively. The arrow lines are showing interaction among different components. It is clear here that when two new components E,F are added in a CBSS then interaction among of past four components get also increased that clearly leads to complexity of those components.

In our approach we are estimating the dependency of one module on the other module in terms of no. of methods called by other module or no. of variable used. We can estimate the dependency with the help of coupling, cohesion and interface metrics. Moreover, Estimate the software quality in terms of software component reusability.

### 4.1 Interface metric

Interface complexity metric for software components is based on complexity involved in the interface methods and properties used in the interface. These interface methods may have parameters and return values, which are the only source of these parameters and return values and properties, weight values may be assigned to them, which can be used to measure the complexity of the target interface method.

### 4.2 Coupling metric

Coupling between components is the number of other components coupled to this component. In CBSS, coupling will be defined as: two components are coupled if and only if at least one of them acts upon the other. Since coupling is the extent to which the components are interdependent, a quantitative measure is to count the way in which one component may dependent on the other. There are usually two kinds of coupling: afferent coupling and efferent coupling.

### 4.3 Cohesion metric

Cohesion specifies the similarity of methods in a component. It is a measure of the extent to which the various functions performed by a component are related to one another.

## 5. EXPERIMENTAL RESULTS

After experimental evaluation of these metrics the value of coupling metric for the system is 0.95, 1.40, 2.00 in case of 4,6,8 components respectively, the value of cohesion metric for the system is 0.47, 0.51, 0.51 in case of 4,6,8 components respectively, the value of interface metric for the system is 1.18, 1.29, 1.34 in case of 4,6,8 components respectively.

And using the formula:

$$SCCMj = a*MVj + b*COMj + c*AIMj$$

Where a,b,c are the weights for system coupling, cohesion and interface metrics i.e 0.5, 0.2 and 0.3. and MVj represents coupling, COMj represents cohesion and AIMj represents interface metric. The values of SCCM that comes out are 0.92 in case of 4 components, 1.18 in case of 6 components, 1.56 in case of 8 components. When we execute all these values on the MATLAB we get a bar graph showing the relationship among all the three cases as shown in fig 3.

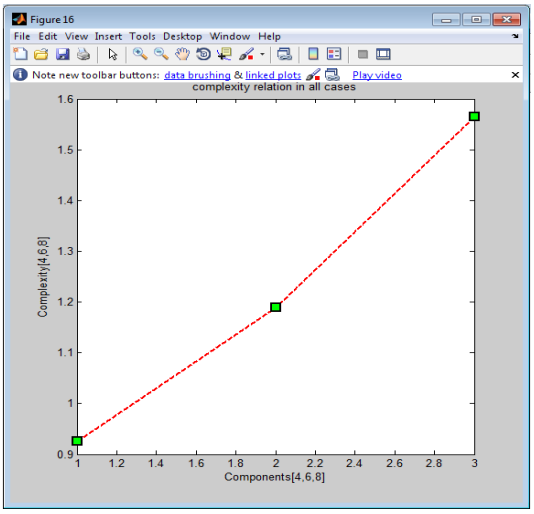


**Fig. 3: Comparison Graph**

The combined values of these metrics are calculated by using the system architecture shown in fig. 4.

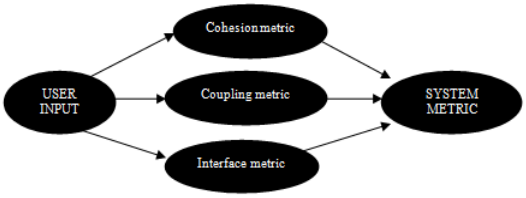


**Fig. 4: System Metric Generation**

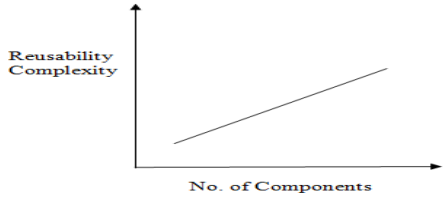


**Fig. 5: Relation between reusability, complexity and No. of components**

If combining result of all three metric is in efficient range then we can say that the quality of software is good in terms of software reusability. Moreover, larger the number of components, reusability is high but at the same time software is more complex (fig. 5). So we can remove those component on which other component are not depended. By doing this we can make the software less complex without affecting its reusability. With this approach ultimately we are improving reuse process, minimizing the amount of development work required for future projects, and ultimately reducing the risk of new projects that are based on repository knowledge.

## 6. CONCLUSION

Software complexity is very important to consider for software development. Software components are one of the major factors that provide the software reusability. The system will check that the use of the component based approach in the system increases or decreases the complexity of the software by using 4 components, 6 components and 8 components. The end result will show the bar graph that shows the complexity increases by adding components in the system and it decreases the reliability of software system.

## 7. REFERENCES

[1]. A.A. Toptsis, J. Emilrazan "Cognitive and Usability Perspectives in Software Platform Libraries", JNIT, Vol. 1, No. 2, pp. 25-34, 2010.

[2] Bass L., Clements P., and Kazman R., Software Architecture in Practice, Addison-Wesley, 1998.

[3]. Garlan D., Allen R., and Ockerbloom J.,Architectural Mismatch: Why Reuse is so hard, IEEE Software, Vo.12, issue 6, 1995.

[4]. G. Gui, P. D. Scott, "Measuring Software Component Reusability by Coupling and Cohesion Metrics", Journal of Computers, vol. 4, no. 9, pp.797-805, 2009.

[5] Jasmine K.S, Dr. R. Vasantha "A New Process Model for Reuse Based Software development Approach" Proceedings of the World Congress on Engineering 2008, Vol IWCE 2008, July 2 - 4, 2008, London,U.K.

[6]. Jianguo Chen "Complexity Metrics for Component-based Software Systems", vol5, issue3.24,*2011.*

[7] Jo Woodison ,Managing Software Reuse with Perforce", Mandarin Consulting.

[8] L.Kharb, R. Singh, "Complexity metrics for component-oriented software systems", ACM SIGSOFT Software Engineering Notes, vol. 33, no. 2, pp.1-3, 2008.

[9] Maurizio Pighin "A New Methodology for Component Reuse and Maintenance" University degli Studi di Udine, Italy.

[10] Prakriti Trivedi ,Rajeev kumar "Software Metrics to Estimate Software Quality using Software Component Reusability",IJCSI, Vol. 9 ,Issue 2, No 2, March 2012.

[11] S. Jing, C. Jiang, "An Approach to Predict Performance of Component-based Software with the Palladio Component Model and Stochastic Well-formed Nets", AISS, Vol. 2, No. 1, pp. 31-42, 2010.

[12] S Sedigh-Ali, A Ghafoor, R Paul,"Software Engineering Metrics for COTS-Based Systems",IEEE Computer, vol. 34, no. 5, pp.44- 50, 2001

[13] V. Narasimhan, B. Hendradjaya, "A New Suite of Metrics for the Integration of Software Components", the First International Workshop on Object Systems and Software Architectures (WOSSA'2004), South Australia, Australia, 2004.

[14] V. P. Venkatesan, M. Krishnamoorthy, "A Metrics Suite for Measuring Software Components", JCIT, Vol. 4, No. 2, pp. 138-153, 2009.

[15] Yoonsun Lim, Myung Kim, Seungnam Jeong and Anm o Jeong "A Reuse-Based Software Development Method" Dept. of Computer Science & Engineering, Ehwa Womans university,120-750 Seoul, Korea.