Predicting Quantitative Functional Dependency Metric based upon the Interface Complexity Metric in Component based Software Systems: A New Approach

Sonu Mittal
Research Scholar
SGVU, Jaipur
India

Pradeep Kumar Bhatia
Professor
GJUS&T, Hisar
India

ABSTRACT
One of the major issues in component based software systems structuring and quality prediction is the interdependencies of system components. This paper proposes a novel technique for determining the strength of functional coupling in component based software systems. Authors propose Strength of Functional Dependency (SFD) metric, which is based upon two new metrics Operational Coupling complexity Index (OCI), and Instance Coupling complexity Index (ICI). It allows us to quantify the functional dependencies, formed by different kinds of operations and instances between these components. Compared to other existing dependency metrics, which are often based on number of operations or instance variables between the components only, authors consider operational complexity and instance variables complexity as a measure to how strong this dependency is and therefore promote a more systematic approach to the reasoning about modularity in component based software systems.

This paper can be divided broadly into two parts. The first part quantifies interface operations and instance variables. The quantification is performed by considering the number of input, output parameters and their types. Based upon these factors of operations and instance variables, authors used analytical hierarchy approach (AHP) to assign weights to these factors and outcomes OCI, ICI and SFD. The second part shows the experimentation and validation of the proposed metrics. The advantages of the proposed method are discussed as well through a case study in this paper.

Keywords
Component based software system, Metrics, Component coupling, Functional Dependency, Operational complexity.

1. INTRODUCTION
The idea to construct software in the same way as hardware is constructed i.e. by integrating reusable components is becoming very popular in the last decades. As these components are usually black boxes in nature, their internal structure and coding is not provided with them. Efficient system functioning is provided by components through interacting, cooperating and coordinating with other components, which results in the form of dependency among components. The only means of their communication with the other components is through their interfaces. In other words, a component has required and provided interfaces. Two components that interact to each other are called "coupled". Further the coupling between the components happens when a component provides an interface and other components use it. That is it is directional and well known as dependency in the Component based software system literature [1-3, 5-7, 10, 14]. In this paper, Functional dependency is considered to be the main dependency affecting CBSSs. The functionality is provided through operations and instance variables passing between these components. Diagrammatically, it can be depicted as shown in figure 1.

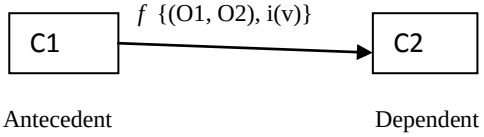

Figure 1: A functional dependency relationship

Here, component C1 is providing some functionality *f* to the component C2 via operations O1, O2 and a set of instance variables denoted by i(v) . So component C2 is dependent upon C1 and C1 is called antecedent to C2.

The coupling between components can be loose or tight, or somewhere in between [14]. The tightness of a coupling is not binary. It is not either "loose" or "tight". The degrees of tightness are continuous, not discrete. Dependencies can also characterize as "strong" or "weak". A tight coupling leads to strong dependencies, and a loose coupling leads to weak dependencies.

Higher dependency leads to a complex system, which results in poor understanding and a higher maintenance cost [1]. Analysis of CBSS dependencies is an important part of software research for understandability [18], testability [19], maintainability [20] and reusability [21] of a component based system. Thus, dependency metrics could have a real impact on the quality of the system delivered to the user.

Many researchers [3, 4, 8, 9, 16, 17] focuses on measuring the interaction complexity of integrated components. In the past, only a few papers based on graph theory addressed the evaluation of CBSS dependency [1-3, 5, 10, 14]. However, there has been no focus on how strong this dependency is?

In this paper, Authors have tried to consider this issue based upon the complexities of different operations and instance variables as a measure of functional dependency hence providing information for quantitative assessment of modularity in the system design. For this Authors have proposed SFD metric based upon two metrics Operational Coupling complexity Index (OCI) and Instance Coupling complexity Index (ICI).

This paper includes 9 major sections. In the next section, authors discuss the aim of this work and the solution strategy. Section 3 identifies the various factors and subfactors related

to functional dependency. Section 4 describes the Analytical Hierarchy Process (AHP) through which the weights are assigned to these factors and subfactors. Section 5 presents the metrics system that supports the measurement by considering individual factors separately. Section 6 provides the experimentation and validation of proposed metrics. Section 7 shows the small case study to apply these metrics and compare it with other related works. Section 8 is about the analysis of result and finally, section 9 concludes with its limitations and discusses further work.

2. AIM AND APPROACH

The ultimate goal of the proposed work is to provide an effective means of quantifying functional dependency. Authors consider coupling complexity index (coupling index in short) as a measure of dependency among system components interfaces, established by the different types and numbers of operations and instance variables (See Figure 1). Dependency shows the domain of subjective and qualitative estimation, while coupling index shows the domain of objective and quantitative measurement. It can be used for measuring various attributes of quality models, like reusability, testability, modifiability etc.

For measuring functional dependency strength, it is necessary to quantify the constituents of functions i.e. Operations and instance variables. For measuring the degree of coupling through operations and instances Authors propose OCI and ICI by normalizing it to 0-1. They are empirically validated and applied through a small case study. The overall approach can have following steps:-

1. Identify and classify the various factors and subfactors for measuring operational coupling index and instance variable index.
2. Assign weight to each factor and subfactors using Analytical Hierarchy Process (AHP) approach.
3. Propose SFD, OCI and ICI metrics.
4. Experimentation and evaluation of the proposed metrics.
5. Applying and comparing through a small case study.

3. IDENTIFY AND CLASSIFY THE FACTORS FOR MEASURING OPERATIONAL COUPLING BETWEEN TWO COMPONENTS.

To identify the various factors and subfactors first, we have to understand the basic structure of Component based software system. Based upon various literature reviews [22, 23, 24], It can be depicted by figure 2.

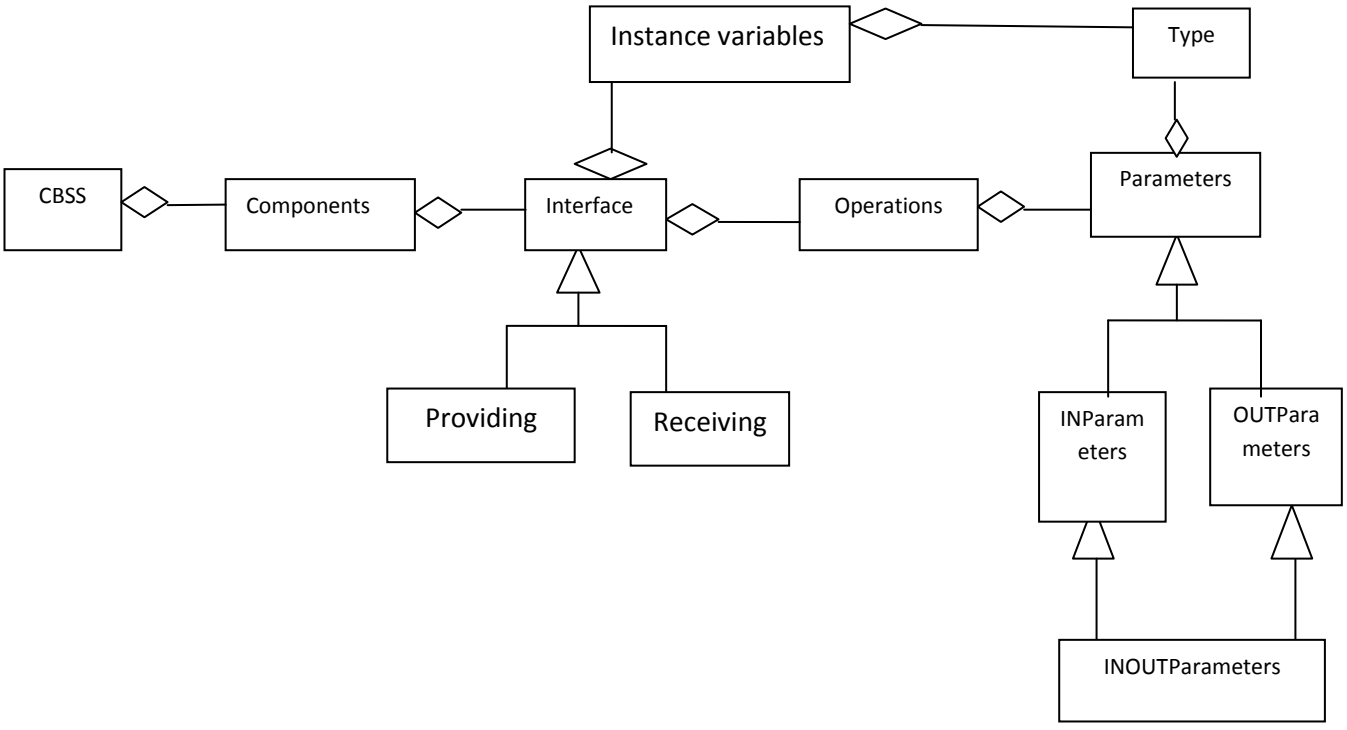

Figure 2: The basic structure of CBSS

A CBSS can be made up of many components. Each component consists of many Interfaces. The interface can be categorized as providing interface or receiving interface. Further each interface provides or receives functionality through operations or methods and instance variables. As components are black box in nature (especially COTS components), the interface operations provides information about only the number and type of Input and Output parameters. Similarly the instance variables will have types.

Most of the researchers [1-7, 10, 17] considered only the number of instance variables and the number of operations/methods invoked in each interface as a measure of direct dependency and complexity between/of the components. One of the important point of consideration here, Authors argue that not only the number of operations and instance variable should be the sole measure of the strength of functional dependency. The complexity of operations and instance variables should be considered while assigning and strengthening the functional dependency. Only a few researchers [8, 9, 16] considered this level of complexity as a measure for interface dependency between the components.

According to chiller et. al. [9], Interface coupling defined in terms of the number of methods and instance variables invoked by component from other components. Weighted values were assigned to constituents of interface coupling (methods and instance variables). Interface methods were classified according to data type of arguments and return values. Basic assumption in computing data type of instance variables, arguments and return type (interface methods) were: Primitive data types such as integer taken simple, Structured data type such as string, array, list, and vector taken medium, Complex data types includes class type, user defined components, pointers, references and others .If arguments were different for same interface method then higher data type taken. Similarly they compute data type of return types. Their result shows that complexity and dependency increases with increase in number of invoked methods and instance variables during an interface in a component based system.

Similar approach of strengthening the various types of operations/methods in an interface using the weighted assignment technique was conducted by kaur et.al. [8], and showed that the coupling increases with numbers and complexity of input and output parameters. Although, these methods considered operational/method level complexity by considering the number and type of arguments and instance variables in the interface as a major factor for calculating the complexity of the interfaces, yet authors want to draw some important points which may be useful while considering these complexities as to quantify the functional dependency between the components through these interfaces.

These researchers [8, 9] considered return value i.e. IN parameters as a part of OUT parameter and not considered it separately. Authors argue here that the IN parameter should be given equal weightage to the total number of OUT parameters because the methods with return values (IN parameters) will have more coupling than the methods without the return values but total number of arguments/parameters is equal in both the operations. For example a method with 2 simple and 3 complex OUT parameters and without any IN parameter should not be considered equal to the method with 2 simple and 2 complex OUT parameters but with one complex return value.

Another point that should be considered that if an interface/function is providing 2 methods/operations with simple or no arguments then its complexity and hence coupling index may be lower than an interface method which is providing very complex functionality because the number of parameters is not only the sole criteria in deciding the complexity of the function. It is the combination of many factors and sub factors. Accordingly, authors are classifying interface operations into four categories. Each return value is considered as IN Parameter and arguments passed as OUT parameters.

1. Interface operations without IN Parameter and without OUT Parameters

2. Interface operations with IN Parameter but without OUT parameters

3. Interface operations without IN Parameter but with OUT parameters

4. Interface operations with IN Parameter and with OUT parameters

Further both IN and OUT parameters complexity can vary according to data types of these parameters (TOP). So authors categorized them into three types as simple, medium and complex in the same way as in [9]. Primitive data types such as integer, Boolean, double etc. will be considered as simple (S), Structured data type such as string, array, list, and vector will be considered as medium (M), and class type, user defined components, pointers and references will be considered as complex(C).

Authors will take another factor into consideration that the overall weight for each data type will be taken different according to number of these data types present in these operations. Authors have assigned equal weightage to IN parameters and OUT parameters i.e. 0.5 to each. As return value either maybe or not. So IN parameters can have four sub factors:- No return value (NR), Return Simple (RS), Return Medium (RM), Return Complex (RC). OUT parameters numbers (NOP) may further be divided into no out parameters (NO), out parameters from 1 to 4 (1-4), out parameters from 5 to 8 (5-8), and out parameters greater than 8 (>8).

According to various combinations, the OUT parameters subfactors considered are No OUT parameters (NO), 1-4 simple type parameters (1-4S), 1-4 medium type parameters (1-4M), 1-4 complex type parameters (1-4C), 5-8 simple type parameters (5-8S), 5-8 medium type parameters (5-8M), 5-8 complex type parameters (5-8C),>8 simple type parameters (>8S), >8 medium type parameters (>8M), >8 complex type parameters (>8C). In the similar way to OUT parameters instance variables can have same factors.

4. ASSIGN WEIGHT TO EACH FACTOR AND SUBFACTORS USING ANALYTICAL HIERARCHY PROCESS (AHP) APPROACH.

Researchers in [8, 9] used weighted assignment technique to assign weights to these factors. Here authors will use Analytical Hierarchy Process (AHP) approach to assign weights to these factors. AHP is a technique that supports decision makers in structuring complex decisions, quantifying intangible factors, and evaluating choices in multi-objective decision situations. It is a comprehensive and rational decision-making framework that provides a powerful methodology for determining relative worth among a set of elements [11].Weighted sum method is easy to use and understand but weights to the attribute are assigned arbitrary and it is difficult task when a number of criteria are high. Another problem with weighted scoring method is that common numerical scaling is required to obtain final score. In AHP approach decision makers can compare each alternative that improves decision making procedure by accommodating the ambiguity in human decision making. AHP is especially suitable for complex decisions that involve the comparison of decision elements which are difficult to quantify. Its application has been reported in numerous fields, such as transportation planning, portfolio selection, corporate planning and marketing [13]. It involves:

a) Development of relative importance among the factors and sub factors using expert's opinion or through exhaustive paired comparison analysis,

b) Assigning a weightage for each of the factor and sub factor using priority vector,

c) Performing similar analysis for the alternative solution strategies for each of the attributes, and

d) Developing a single overall score for each of the alternate solution strategies.

The total score for all alternates will be 1.

Various researchers [12, 15] have successfully applied this method to assign weight to various factors in the field of computer engineering.

The priority for each alternative is assigned as follows:-

IN parameters (IN) and OUT parameters (OUT) have been assigned equal priority in deciding the coupling complexity between the components, that is, IN=OUT

For IN parameters NR < RS < RM < RC.

For OUT parameters Number of out parameters (NOP) has equal weightage to type of parameter (TOP), that is, NOP=TOP

Further for NOP the priority assigned will be as follows:-

0 < 1-4 < 5-8 < (>8)

TOP priority is assigned as: - Simple (S) < Medium (M) < Complex (C)

Similar method is adopted for instance variables.

Authors have used an open source tool Open Decision Maker (ODM) v 1.0.1, to make sensitivity analysis between the alternative factors and to calculate the overall weights using AHP. The critical consistency ratio is under 0.1 in all the cases. The results are shown in the Table 1 and Table 2

Table 1: Weighted values of IN parameters and OUT parameters of operations

Factors	Sub-Factors		Weight Value	Sum	Grand Total
IN	No Return Value (NR)		0.02	0.5	1
	Return	Simple (RS)	0.06		
		Medium (RM)	0.13		
		Complex (RC)	0.29		
OUT	No OUT Parameter (NO)		0.008	0.5	
	1-4	Simple (1-4 S)	0.012		
		Medium (1-4 M)	0.02		
		Complex (1-4 C)	0.04		
	5-8	Simple (5-8 S)	0.02		
		Medium (5-8 M)	0.04		
		Complex (5-8 C)	0.08		
	>8	Simple (>8 S)	0.04		
		Medium (>8 M)	0.08		
		Complex (>8 C)	0.16		

As instance variables will always be present when an interface provides some functionality to required interface, so it can't be zero. The weight value for instance variables is also decided in similar way as OUT parameter weights decided (See table 2).

Table 2: Weighted values of Instance variables

Factors		Weight Value	Grand Total
1-4	Simple (1-4 S)	0.02	1
	Medium (1-4 M)	0.03	
	Complex (1-4 C)	0.08	
5-8	Simple (5-8 S)	0.03	
	Medium (5-8 M)	0.08	
	Complex (5-8 C)	0.17	
>8	Simple (>8 S)	0.08	
	Medium (>8 M)	0.17	
	Complex (>8 C)	0.34	

5. THE PROPOSED METRICS

5.1 The OCI metric

Operational Coupling complexity Index (OCI) can be calculated as follows:-

OCI = Weighted value of IN parameter factor + Sum of weighted value of OUT parameter factors

Where IN Parameter value set can be IN= {0.02, 0.06, 0.13, 0.29} depending upon the above sub factors and OUT Parameter value is the combination of the weights of all the sub factors present in the operation according to type of parameter (TOP) and number of parameters (NOP).

5.2 The ICI metric

Instance Variable Coupling complexity Index (ICI) can be calculated as

ICI = Sum of weighted value of instance variable factors

5.3 The SFD metric

The strength/degree of functional dependency (SFD) can be measured as follows:

$$SFD = \sum_{i=1}^{n} OCI_i + ICI$$

Where OCI_i is the OCI value of i_{th} operation and n is the total number of operations in functional dependency.

6. EXPERIMENTATION AND VALIDATION OF PROPOSED METRICS

As discussed in the third section, the various types of operations can be present in a function. Each type of function will posses' different value for OCI, and hence will influence the SFD. The various possible minimum and maximum value cases for these different types of operations can be shown as follows:-

CASE I Operations with No IN Parameter (NR) and No OUT Parameters (NO)

OCI = 0.02 + 0.008 = 0.028

CASE II Operations with IN parameter but No OUT Parameters

Subcase 2.1 (Minimum value) **Return Simple (RS) value but No OUT (NO) parameters**

OCI = 0.06 + 0.008 = 0.068

Subcase 2.2 (Maximum Value) **Return Complex (RC) value but No OUT (NO) parameters**

OCI = 0.29 + 0.008 = 0.298

CASE III Operations with No IN Parameter (NR) but with OUT parameters

Subcase 3.1 (Minimum Value) **No return Value (NR) and 1 simple (1-4 S) OUT parameter**

OCI = 0.02 + 0.012 = 0.032

Subcase 3.2 (Maximum Value) **No return (NR) value and 10 simple (>8S), 10 complex (>8C) and 10 medium (>8M) type OUT Parameters**

OCI = 0.02 + 0.04 + 0.08 + 0.16 = 0.30

CASE IV Operations with IN parameters and OUT Parameters

Subcase 4.1 (Minimum Value) **Return Simple (RS) value and 1 simple (1-4 S) OUT parameter**

OCI = 0.06 + 0.012 = 0.072

Subcase 4.2 (Maximum Value) **Return Complex (RC) value and 10 simple (>8S), 10 complex (>8C) and 10 medium (>8M) type OUT Parameters**

OCI = 0.29 + 0.04 + 0.08 + 0.16 = 0.57

Graphically, the result values can be shown by figures 3 and 4.

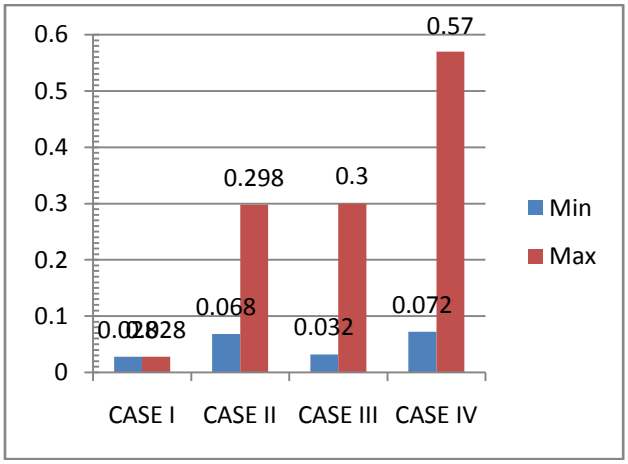

Figure 3: Minimum and maximum OCI values for all 4 types of operations

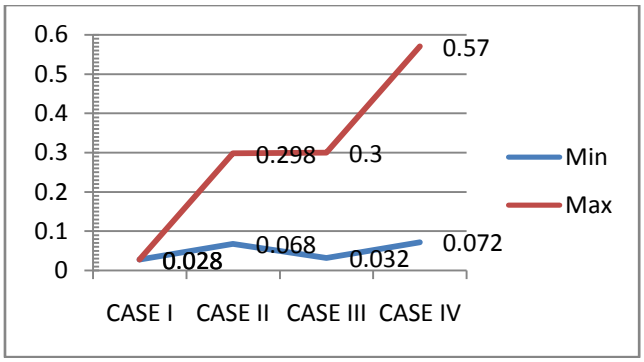

Figure 4: OCI value range for all the cases

Similarly, the ICI values for the minimum and maximum cases can be depicted as follows.

CASE 5.1 (Minimum ICI value)**: 1 Simple type instance variable**.

ICI = 0.02

CASE 5.2 (Maximum ICI value)**: 10 Simple type, 10 medium type and 10 complex type instance variables.**

ICI = 0.08 + 0.17 + 0.34 = 0.59

Graphically, It can be depicted by figure 5:-

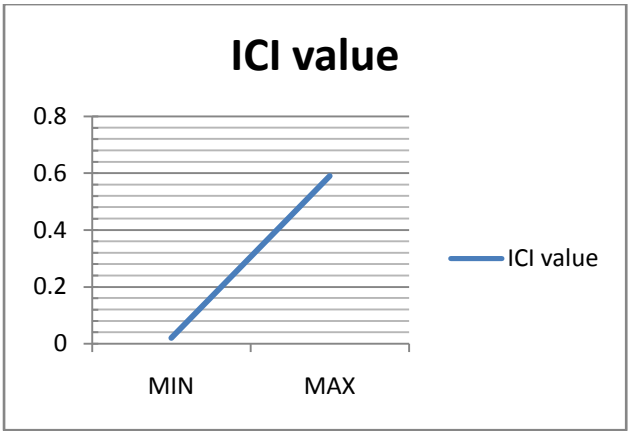

Figure 5: ICI value for all the possible cases

Thus the minimum value of OCI among all the cases will be in case I, that is 0.028 and the maximum value among all the cases will be in case IV, that is 0.57 (see figure 3 and 4). The minimum value for ICI will be for 1-4 S i.e. 0.02 and maximum value for ICI will be. 0.59 (See figure 5). Thus the graph shows that the ICI and OCI is not the sole measure of one factor. It is not a simple linear scale because it is the combination of many factors and subfactors and hence the SFD will also not increase only with the number of operations but it will increase with the combination of all the above factors and subfactors.

7. COMPARISON AND EVALUATION OF METRICS

The strength/degree of functional dependency (SFD) can be measured by using OCI and ICI as follows:

$$SFD = \sum_{i=1}^{n} OCI_i + ICI$$

Authors can show the advantage of our method by considering a case study

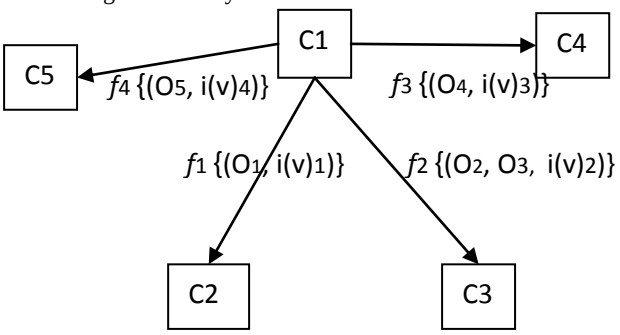

Figure 6: A small case study

As shown in the figure 6, Component C1 is providing interfaces to Components C2 (via function *f*1) and C3 (via function *f*2), Component C4 (via function *f*3) and component C5 (via function *f*4).

The ICI and OCI values are calculated on assumed values associated with operations and instances are described as follows: (see Table 3 and Table 4).

Table 3: Assumed parameters values and calculated OCI values

Operations	IN Parameters	OUT Parameters	OCI
O1	RM	5 M, 6 C	0.13 + 0.04 + 0.08 = 0.25
O2	NR	NO	0.02 + 0.008 = 0.028
O3	RS	6 S, 2M	0.06 + 0.02 + 0.02 = 0.10
O4	RC	9S,9C	0.29 + 0.04 + 0.16 = 0.49
O5	NR	6M,6C	0.02 + 0.04 + 0.8 = 0.14

Table 4: Assumed parameter values and calculated ICI values

Instance variable	Instance variables parameters	ICI
i(v)1	6 S	0.03
i(v)2	6 S	0.03
i(v)3	10 M	0.17
I(v)4	6S	0.03

Thus the SFD values for various functions between the two components can be calculated and it can be compared with the Interface coupling complexity (ICC) metric by chiller et. al. [9] values (See table 5). The ICC value can be calculated as ICC (External) = [w10 (Σ (m/v invoked (simple) +w11 Σ (m/v invoked (medium) + w12 Σ (m/v invoked (complex)]) where w10 to w12 are weight values of methods/operations and instance variables invoked by a component in from other components in a component based system i.e. external to component. These weight values are assigned according to the weighted assignment technique.

Table 5: Calculated SFD and ICC values

Function	∑ OCI	ICI	SFD	ICC
*f*1	0.25	0.03	**0.28**	(1* 0.14) + (6 * 0.02) = **0.26**
*f*2	0.028 + 0.10 = 0.128	0.08	**0.208**	(2 * 0.10) + (6 * 0.02) = **0.32**
*f*3	0.49	0.17	**0.66**	(1* 0.14) + (10 * 0.04) = **0.54**
*f*4	0.14	0.03	**0.17**	(1* 0.14) + (6 * 0.02) = **0.26**

8. RESULTS AND DISCUSSION

In the above section, authors have showed that how their metrics can help in measuring the functional dependency strength between the two components. Authors compared SFD with the ICC metric and results shows (See table 5) that the SFD values for $f1 > f4$ while ICC values for $f1 = f4$. Secondly, SFD values for $f2 < f1$ while ICC values for $f2 > f1$. These discrepancies in the result are due to IN parameter factors and show how the proper weightage to IN parameters can affect the overall coupling results, which lacked in the previous works [8, 9]. It also proves our point that if a function contains more number of simple operations then it may have lesser value of SFD than the function with lesser number of operations but complex in nature, because authors considers that the complexity of operation not depends upon only one factor but is a combination of factors and subfactors.
Higher value of SFD leads to lowers maintainability, testability and reusability of the Component based software systems. Thus the SFD metric can be proved very helpful in measuring the various quality attributes of the CBSS.

9. LIMITATIONS, CONCLUSION AND DIRECTIONS FOR FUTURE RESEARCH

In this paper, Authors proposed a metric system for quantifying the strength of functional dependency between the components, which is based upon the number and type of their functional constituents i.e. operations and instance variables.
Although the proposed metric system provides a systematic and strong AHP based approach to measure the strength of functional dependency between the components and shows how the complexities of operations can affect the functional dependency, yet it considers only the direct dependency between the components. Here authors are not considering the issue of indirect dependency. Secondly, authors are considering here that the same technique can be applied for measuring the coupling strength between the internal and external components and does not differentiate it as some researchers does [9]. Authors have tried to evaluate their metric system through a small case study but the empirical validations can further improve in deciding and assigning proper weightage to these metrics.
In 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 systems. This paper represents only the beginning of the research that should be undertaken to explore the AHP approach for the quantification and using SFD metric for measuring various quality attributes in CBSS area. So authors invite researchers to comment on the new approach, they proposed, whether captures the real essence in this area.

10. REFERENCES

[1] Sharma, A., Grover, P.S., Kumar, R., 2009. Dependency Analysis for Component-Based Software Systems Volume 34 Number 4. ACM Sigsoft.

[2] Li. B., 2003. Managing dependencies in component-based systems based on matrix model. Proc. Proceedings of Net. Object. Days, Citeseer, 2003, pp.22-25

[3] Gill N.S. and Balkishan, 2008. Dependency and interaction oriented complexity metrics of component-based systems. SIGSOFT Softw. Eng. Notes, vol.33, pp. 1-5., http://doi.acm.org/10.1145/1350802.1350810.

[4] N. Salman, 2006. Complexity Metrics AS Predictors of Maintainability and Integrability of Software components. Journal of Arts and Sciences.

[5] Lisa, C., Delugach, H. S., 2001. Dependency Analysis Using Conceptual Graphs, In Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001, pp: 117-130.

[6] Qu, B., Liu, Q., Lu, Y., 2010. A Framework for Dynamic Analysis Dependency in Component-Based System..

[7] Vieira, M. and Richardson, D. 2002. Analyzing Dependencies in Large Component-Based Systems. Proceedings of the 17th IEEE International Conference on Automated Software Engineering (ASE'02), 2002, pp 241 – 244.

[8] Kaur Ramanpreet, July 2010. Evaluation of Software Complexity using Weighted Assignment Technique for Component based System. M.Tech. Thesis.

[9] Chillar R. S., Ahlawat P. and Kumari U. 2012. Measuring Complexity of Component Based System Using Weighted Assignment Technique. Proc. 2nd International Conference on Information Communication and Management (ICICM 2012) IPCSIT vol. 55 (2012) © (2012) IACSIT Press, Singapore. DOI: 10.7763/IPCSIT.2012.V55.4

[10] Majdi Abdellatiefab, Abu Bakar Md Sultana, Abdul Azim Abd Ghania, Marzanah A.Jabara. 2011. Component-based Software System Dependency Metrics based on Component Information Flow Measurements. ICSEA 2011 : The Sixth International Conference on Software Engineering Advances, IARIA, 2011. ISBN: 978-1-61208-165-6

[11] Jayaswal, B. K., Patton, Peter C., Forman, Ernest H., 2007. The Analytic Hierarchy Process (AHP) in Software Development, Prentice Hall.

[12] DeJiu Chen, Martin Törngren, 2004. A Metrics System for Quantifying Operational Coupling in Embedded Computer Control Systems. EMSOFT'04, September 27–29, 2004, Pisa, Italy. Copyright 2004 ACM 1-58113-860-1/04/0009

[13] Saaty TL., 1980. Multicriteria decision making: The analytic hierarchy process. McGraw-Hill.

[14] Jyoti Rani , kirti seth, 2012. Dependency Analysis for Component based Systems using Minimum Spanning Tree International Conference on Advances in Computer Applications (ICACA 2012) Proceedings published by International Journal of Computer Applications® (IJCA).

[15] Sharma A, Grover R S, Sharma R. 2008. Estimation of Quality for Software Components – an Empirical Approach. SIGSOFT Software Engineering Notes, November 2008 Volume 33 Number 6 DOI: 10.1145/1449603.1449613.

[16] Khimta S, Sandhu P and Brar A, 2008. A Complexity measure for java bean based software components, World Academy of Science, Engineering and Technology, Volume 42.

[17] Kharb L, Singh R, 2008. Complexity metrics for Component oriented Software Systems. ACM SIGSOFT Software Engineering Notes, Vol 33, Issue 2, pp 1-3.

[18] Lucia A, Fasolino A.R. and Munro M, 1996. Understanding function behaviors through program slicing. wpc, 1996, pp. 9.

[19] Bates S and Horwitz S, 1993. Incremental program testing using program dependence graphs. Proc. Proceedings of the 20th ACM SIGPLANSIGACT symposium on Principles of programming languages, ACM, pp.384-396

[20] Gallagher K.B. and Lyle J.R., 1991. Using program slicing in software maintenance. IEEE Transactions on Software Engineering,, vol.17, pp. 751-61.

[21] Gui G. and Scott P.D., 2009. Measuring Software Component Reusability by Coupling and Cohesion Metrics Journal of Computers, vol.4, pp. 797-805.

[22] Monge.R, Alves C., Vallecillo A., 2000. A graphical representation of COTS based software architecture, In proc. IDEAS, Citeseer, 2000.

[23] Szyperski C, 1998. Component Software. Beyond Object-Oriented Programming. Addison-Wesley, 1998.

[24] http://www.umlcomponents.com last visited on 16/may/2013.