Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol.7 No.2, 2018, pp.11-17

DOI: https://doi.org/10.51983/ajcst-2018.7.2.1877

Optimizing the Software Metrics for UML Structural and Behavioral Diagrams Using Metrics Tool

D. Singh[1] and H. J. S. Sidhu[2]
[1]Department of Computer Science and Engineering, Guru Nanak Dev Engineering College, Ludhiana, Punjab, India
[2]Department of Computer Applications, Desh Bhagat University, Mandi Gobindgarh, Punjab, India
E-Mail: diljitsingh007@gmail.com, jeetsinder@gmail.com
(Received 15 May 2018; Revised 10 June 2018; Accepted 28 June 2018; Available online 5 July 2018)

***Abstract*- In this paper, we have proposed an efficient way to calculate the software metrics of structural and behavioural diagrams of unified modelling language (UML) with the help SD Metrics as a tool. This method is applied to measure the internal quality of attributes and functions of structural and behavioural diagrams of unified modelling language (UML). The SD metrics tool collect the information after parsing the XMI format generated by UML compiler of structural and behavioural UML diagrams. The object-oriented design made by structural and behavioural diagrams holds the important part of designing in the development process of the software. Early the measurement of metrics will lead to good quality of the software from coding, but now using the design metrics we calculate the cohesion, coupling and other metrics with easy and effective with the fewer efforts which improve the quality of software to be developed at the design phase.**
Keywords: UML diagrams, Object-oriented design, Metrics, Model-driven metrics

I. INTRODUCTION

The Unified Modeling Language (UML) is a representation as a graphical view of a model for a system which partial signifies the design and implementation of software to be developed by the software team. The UML diagrams contain the elements: The UML nodes that were connected with edges and also known as paths to the different objects or modules. The UML model might also contain the different documentation like use-cases. The diagram is basically defined by the graphical symbols which were represented on the various UML diagram. A diagram where the contents area are classes with all attributes well as functions of class diagram[4] A diagram which represents the use of classes is "class diagram". A diagram which represents the sequence of message exchanges between the objects is called "sequence diagram". In figure 1 list of structural and behavioural diagrams are represented. The structural view is represented by the UML diagrams show the static structure of the full system and its parts of different abstraction and implementation of the system and shows us how they are related with one another. The elements in a structure diagrams represent the valuable concepts of a system, and which may include implementation concepts as well, abstract and real world also. The structure diagrams are not consuming time-related concepts it does not show the details of the dynamic behaviour of the scenarios.

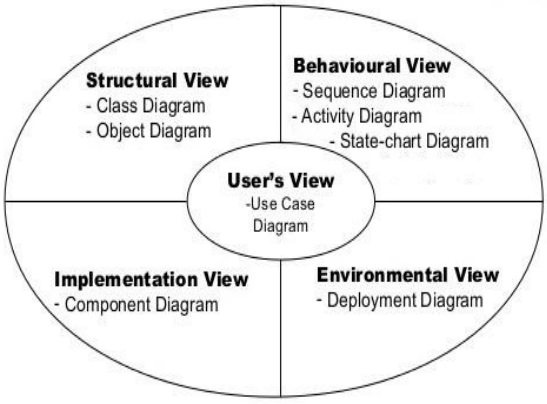

Fig. 1 UML Diagram

However, they may represent the relationships to the behaviours of the classifiers permitted in the various diagrams.

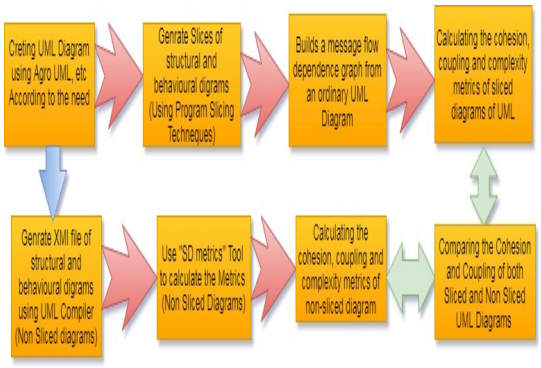

Fig. 2 Flow Chart

In the figure2 the flowchart is represented, in which the flow of our work is displayed. First of all the UML diagrams are created according to the requirement of the user with help of UML compiler like Agro UML, Plant UML, etc. Secondly XMI code is generated with the help of UML compiler and then the XMI code is taken as input for the SD Metrics Tool at the last various attributes and operations of the diagrams were evaluated and calculate the cohesion and coupling and other metric.

II. RELATED WORK

In this section, we briefly review the related work on the importance of UML metrics. Most of our work reported in metrics importance, technique and need for the UML structural and behavioural diagrams of UML. GargSushil *et al.*, [15] purposed the aspect-oriented programming which is a new paradigm for improving the system's features such as modularity, readability and maintainability. Aspect-oriented software development is a new technique to support the concerns in software development. Coupling is an internal software attribute that can be used to indicate the degree of interdependence.

R. Mall *et al.*, [16] propose a technique for static slicing of UML models. First, transform software architecture specified using UML into an intermediate representation named Model Dependency Graph (MDG). Model Dependency Graph (MDG) combines information of sequence diagrams along with the relevant information of the model. For a given slicing criterion, slicing algorithm traverses the constructed MDG to identify the relevant model elements. Algorithm's novelty lies in its computing a slice-based UML model.

SikkaPreeti *et al.*, [17] program slicing is proved in breaking down the large program into the small relevant parts that are needed as per the specified criteria. The objective of this paper is the betterment of existing program slicing techniques. Method / statistical analysis the methodology is proposed to start slicing the software from designing the various levels of the model and continue it with source code level. VermaPreety *et al.*, [12] for measuring the software, appropriate metrics are needed. To attain the various qualitative and quantitative aspects of software. To measure the software in terms of quality, size, efforts, efficiency, and reliability, performance etc. there are different metrics available in software engineering and it has been an area of interest for the various researchers. Measures of specific attributes of the process, project and product are used to compute Software metrics. This work proposes a similar approach of measuring software using various UML diagrams and applied Software size metric to evaluate the size of the Software. All of the above authors and number of other authors had proposed and implemented the techniques for calculating the metrics for one of UML diagram but we had purposed as well as implemented the best optimal technique other than the above authors for structural as well as behavioral diagrams to reduce the testability as well as maintainability of the software.

III. INTRODUCTION TO CLASS CLASS DIAGRAM AND SLICED CLASS DIAGRAM

In figure 1, class diagram and object diagram describes the structure of the class and object. The class serves as a data type for objects like in object-oriented unified modelling language (UML). The class represents set of objects having same responsibilities. An attribute is a named property of a class. An attribute has a named type and defines the types of its objects. Example name, roll and section are the attributes in student class. An operation is a function or services that the objects have to serve. Example, display (), add (), edit () and delete () are the operations in student class.

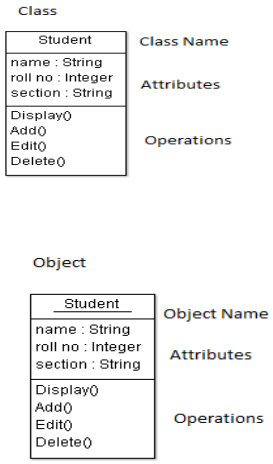

Fig. 3 Class and Object diagram

A. Creation of Class Diagram (UML)

The purpose of a class diagram is to depict the classes within a model. In an object-oriented application, classes have attributes (member variables), operations (member functions) and relationships with other classes. The UML class diagram can depict all these things quite easily. The fundamental element of the class diagram is an icon the represents a class. In figure 2, Class Diagram (Banking system) represents the five classes bank class, customer class, teller class, account class, loan class having their independent attributes and operations.

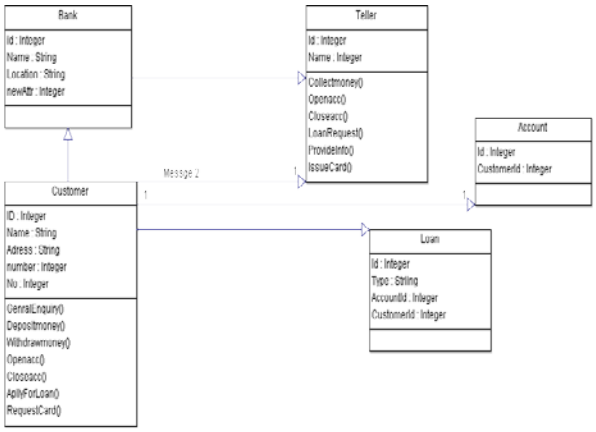

Fig. 4 Class Diagram (Banking system)

B. Class Dependency Graph

In figure 3, Class Dependency Graph (CDG), classes and their attributes, methods and their call parameters, together with method return values are represented as different types

of nodes. These would be represented by using appropriate dependence edges in the Class Dependency Graph (CDG). The member dependence edges represent the class memberships, while method dependence edges represent the dependence of the call parameters and return values (if any) on a method.

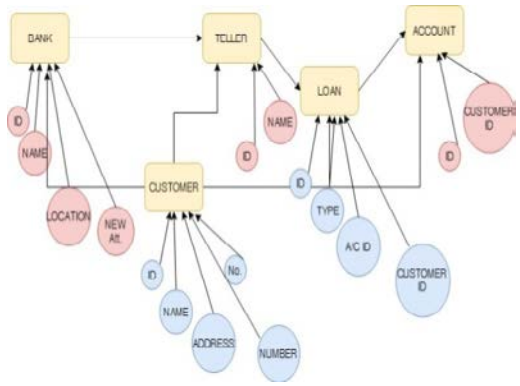

Fig. 5 Class Dependency Graph (CDG)

The data dependencies arise when the class methods, its parameters and return values directly or indirectly make use of the class attributes. In addition, data dependence edges also represent the effect of the calling parameters on the return value of a method. Relationship dependence edges represent how a class is related to another class, or how an instance of a class is related to the other class instances.

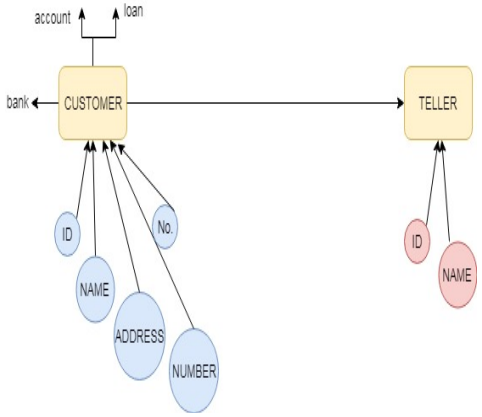

Fig.6 Slicing Graph of Class Dependency Graph (CDG)

In figure 4, Slicing Graph of Class Dependency Graph (CDG) is generated according to the requirement, in which customer and teller class are taken as a sliced criterion. At the same time, Class Dependency Graph (CDG) represent class relations in any different way compared to its corresponding class diagram. For example, the arrow represents generalization, aggregation, and composition relations using the relationship dependence edges. An association relation represents classes which are likely to communicate with other classes due to a method invocation.

C. Calculation of Slice-Based Degree of Cohesion from Class Dependency Graph

Slice-based cohesion metrics and details the formulae used in their calculation. Worked examples, illustrating the calculation of slice-based cohesion metrics for a class dependency graph (CDG), Degree of Cohesion (DCH), Number of Attributes Used (NAU), Total Number of Attributes (TNA)

The degree of Cohesion (DCH) = NAU/TNA
For computing the Degree of Cohesion (DCH), the connectivity between the classes and the number of attributes used by the method of a class. In figure 4, "Slicing Graph of Class Dependency Graph (CDG)" where two classes customer and teller were taken as sliced criteria. The degree of cohesion is shown with respect to the customer is 0.4 as well as with respect to the teller class the result is 1.0. The cohesion is interaction within a class and coupling is the interaction with the other classes. Always this high cohesion and low coupling are recommended in the software to develop successful software.

TABLE I DEGREE OF COHESION

Class Name	NAU	TNA	DCH
Customer	2	5	0.4
Teller	2	2	1.0

D. Calculation of Slice-Based Degree of Coupling from Class Dependency Graph

The degree of coupling is the ratio of a number of messages received to the number of the message passed. For finding the degree of coupling, message received coupling (MRC) and the message passed coupling (MPC) is used, it is the number of messages received and passed by a class. (MRC) Message received coupling is the complexity of message received by the classes, as MRC is the number of messages received by a class from the other classes. (MPC) Message Passed Coupling is denied as the number of the message passed among objects of the classes. The degree of coupling is given
Degree of coupling (DC) = MRC/ MPC

The degree of coupling is shown with respect to the customer is 0.0 as well as with respect to the teller class the result is 1.0. The cohesion is interaction within a class and coupling is the interaction with the other classes. Always this high cohesion and low coupling are recommended in the software to develop successful software.

TABLE II DEGREE OF COUPLING

Class name	MRC	MPC	DC
Class Customer	0	4	0
Class Teller	2	2	1

IV. CREATE SEQUENCE DIAGRAM OF UML

The UML Sequence Diagrams having time-dependent sequences of interactions between objects. They show the sequence of the messages. A Sequence Diagram has two dimensions: the vertical dimension represents time, and the horizontal dimension represents various instances. Normally time proceeds from top to bottom. Sequence Diagrams describe interactions among software components. In UML, a message is a request for a service from one UML actor to another.

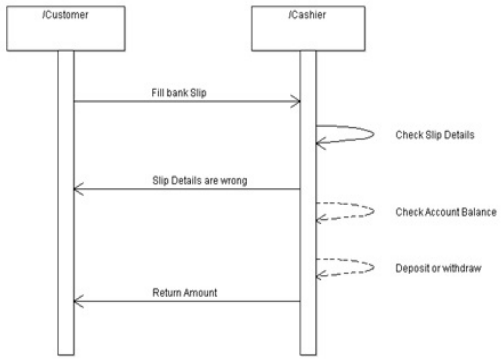

Fig. 7 Sequence Diagram

In fig. 1 the Sequence Diagram of UML is showing the various transactions of messages from one object to another object there are two objects in the Sequence Diagram object 1 customer, object 2 cashiers and having a flow of messages in the various directions. Each message is given a serial number for ease of understanding. These message numbers were taken in the explanation of scenarios and states in the next step of the design and implementation.

A. Recognize States and Scenarios of Sequence Diagram

Considering Sequence Diagram (SD), convert the SD into a Sequence Dependency Graph (SDG). The various States and Scenarios are tabled as shown in Table II.

TABLE III SCENARIO TABLE

Scenario 1	Scenario 2	Scenario 3
STATE X	STATE X	S4 : (m4, b, b)
S1 : (m1, a, b)	S2 :(m3, b, a)	S5 : (m5, b, b)
S2 : (m2, b, b)	END STATE X	END STATE X
S6 : (m6, b, a)	_ _	_ _
END STATE X	_ _	_ _

B. Create Sequence Dependency Graph (SDG)

In this section, the transformation of the Sequence Diagram into the Sequence Dependency Graph by using the scenarios and states as discussed in the last section. The Sequence Dependency Graph (SDG) is made for the representation of the components of the Sequence Diagram and this representation is helpful in making the slices of Sequence Diagrams.

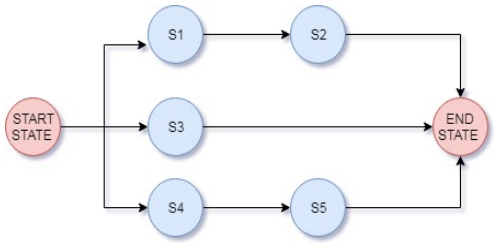

Fig. 8 Sequence Dependency Graph (SDG)

To create the Sequence Dependency Graph, this approach is taken from Monalisa Sharma et al (2007) which describe the nodes representing the various states; each node basically represents an event. States S1, S2, S3, S4 and S5 represent the transition from one object to another.

START STATE represents the object from where operation begins.
END STATE represents the object at where operation ends.

In fig 2 the Sequence Dependency Graph of Sequence Diagram represents the nodes and edges, the directed edges show the direction from one node to another node.

C. Perform Dynamic Slicing on Sequence Dependency Graph (SDG).

The Program slicing is the technique, which is used for the computation of the program statements, the program slice that may affect the value at some point of interest, refer to as slicing criteria. We are using this technique of program slicing, to make it applicable to the Sequence Dependence Graph (SDG) of the Sequence Diagram (SD). The Diagram basically indicates only the dynamic part, so the Dynamic Slicing of the Sequence Dependency Graph is made. Program slicing can be used in debugging to locate the source of errors more easily. Other applications of slicing include optimization, program analysis, and maintenance. A dynamic slice contains all statements that actually affect the value of a variable at a program point for a particular execution of the program rather than all statements that may have affected the value of a variable at a program point for any arbitrary execution of the program. The dynamic slicing of the Sequence Dependency Graph is done in figure 3.

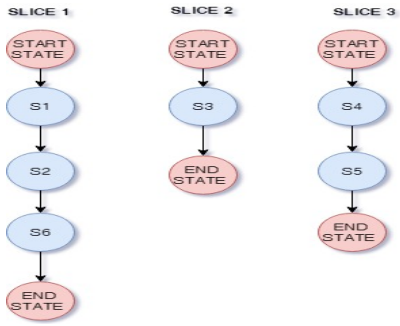

Fig. 9 Dynamic slicing of the Sequence Dependency Graph

To clarify the difference between static and dynamic slicing, In the case of static slicing, since we look at the whole program unit irrespective of a particular execution of the program. But, in the case of dynamic slicing, we consider a particular execution of the program. Sequence diagrams show the dynamic function and if we make the slices of the Sequence Dependency Graph by taking the criteria as the starting and end state of each transaction then in fig 4 a slice 1, slice 2, slice 3 is formulated from the fig 3 as the dynamic program slicing. The next step of the thesis will calculate the software metrics as cohesion and coupling.

D. Calculation of Slice-Based Cohesion of Sequence Dependency Graph (SDG)

In this section after performing all the steps, the slice-based cohesion metrics are calculated by using the formulas. These formulas are basically used for calculation of the statement based cohesion metrics in the earlier papers. The slice based cohesion metrics can also be calculated by using same formulas. Worked examples are shown, illustrating the calculation of slice-based cohesion metrics for a Sequence Dependency Graph(SDG). There are three slices 1,2,3 in fig 11 and their slice size is a number of nodes in each slice that is 4,2,2 respectively.

A total number of nodes by combining all the slices is module size that is 6 according to fig. In this example, the various formulas were calculated as follows:
(Tightness is the number of nodes included in every slice compared to the number)

Tightness (M) = $\frac{|SL\,\text{int}|}{length(M)} = \frac{sizeofslice}{\text{mod}\,ulesize} =$

TABLE IV TIGHTNESS FOR SLICES

Slice Name	Tightness Results
S1	3/6=0.50
S2	1/6=0.16
S3	2/6=0.33

Coverage is a comparison of the length of the slices to the length of the module

Coverage (M) = $\frac{1}{V\circ}\sum_{i=1}^{v\circ}\frac{SL}{length(M)} =$

$\frac{sumofslices}{(no.ofslices)\times(\text{mod}\,ulesize)}$ = 6/3*6=0.33

Min coverage is the ratio of the size of the smallest slice to the module size

Min coverage (M)= $\frac{1}{length(M)}\min i|SLi| = \frac{\min slicesize}{\text{mod}\,ulesize}$ = 1/6 = 0.2

Max coverage is the ratio of the size of the largest slice to the module size

Max coverage (M) = $\frac{1}{length(M)}\max i|SLi| = \frac{\max slicesize}{\text{mod}\,ulesize} =$

3/6 = 0.5

Overlap(M)= = $\frac{1}{|V\circ|}\sum_{i=1}^{|v\circ|}\frac{|SL\,\text{int}|}{|SLi|} = \frac{\sum \text{int}\,er\,\text{sec}\,tionsize}{slicesize} \Big/ no.ofslices =$

3/6+1/6+2/6 = 0.99

E. Calculation of Slice-Based Coupling Of Sequence Dependency Graph (SDG)

This section explains slice-based coupling metrics of the sequence dependency graph. In slice-based coupling metric as an alternative to information flow. They suggest that basing the metric on slices "refine the idea of information flow which has traditionally been associated with coupling measurement". Harman *et al.*, define module f and g the formula is given.

The coupling between two functions f and g is then defined as

Coupling (f, g) =

$$\frac{FF(f,g)\times length(f) + FF(g,f)\times length(g)}{length(f)\times length(g)}$$

The above formula is basically used for the set of program statements to find the coupling and now the same formulas are also used for the calculation of the slice-based coupling of Sequence Dependency Graph.

If Sequence Dependency Graph is considered then there are two types of coupling which will have to be calculated

1. InterSlice Coupling
2. Intra Slice Coupling

InterSlice Coupling: It is the coupling between the slices having a different end state.
Intra Slice Coupling: It is the coupling between the slices having same end states.

These can be clearer when we take an example In fig 2 the slice 1 and slice 2 having the same end state X so intra slice coupling between the slice1 and slice 2.

TABLE V THE COUPLING BETWEEN SLICE1 AND SLICE2

Module (g)	Module size	Module (f)	Common nodes	Total
1	3	1	0	0
2	1	2	0	0

0+0/4 = 0

TABLE VI THE COUPLING BETWEEN SLICE2 AND SLICE 3

Module (g)	Module size	Module (f)	Common nodes	Total
2	1	3	0	0
3	22	2	0	0

0+0/3 = 0

TABLE VII THE COUPLING BETWEEN SLICE1 AND SLICE 3

Module (g)	Module size	Module (f)	Common nodes	Total
1	3	3	0	0
3	2	1	0	0

0+0/5 = 0

Now the slice 1 and slice 2 having the same end state X but slice 3 is having the different end state Z so interslice coupling between (the slice1 and slice 3)and(the slice 2and slice 3)

F. Observing Results of Cohesion and Coupling

The statistical tests are used to correlate the data. Pearson's linear correlation is used to quantify the relationship between metrics. Such correlations measure linear associations between variables. Same statistical tests are also used for summarization of cohesion and coupling metrics. The statistical significance can be summarized by analyzing the values as
0.8 - 1.0 strong association
0.5 - 0.8 moderate association
0.0 - 0.5 weak or no association

A negative value indicates an inverse correlation. In our example of this section, we get results having 0.0 coupling so we are having weak or no association and moderate cohesion, which is true and can be visualized by analyzing the example interactions between the different states.

V. CREATE A STATE CHART DIAGRAM

In figure 1, Statechart Diagram of the banking system which describes the dynamic working of the banking system, in which there are seven states like states (S1, S2, S3, S4, S5, S6 and S7). At state S1 customer fill the details in bank slip, state S2 is for checking the account number filled by the customer, state S3 is for deposit the amount in the bank, state S4 is for the withdrawing the amount from the bank, state S5 is for the check balance, state S6 is for the withdrawing the amount as filled by the customer, and state S7 is for the updating of account balance after withdrawing or depositing the amount.

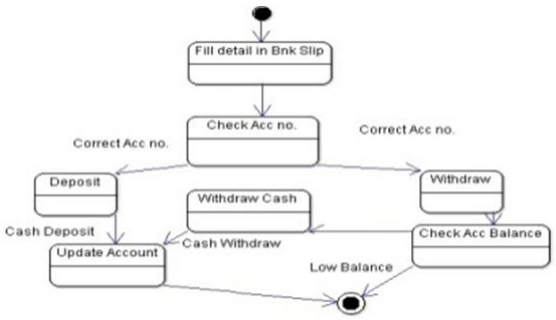

Fig. 10 Statechart Diagram of the Banking System

A. Create a State Dependency Diagram (STDG) from Statechart Diagram.

Now, create a State Dependency Graph of the banking system is created from the Statechart Diagram shown in Figure 1, for that, first, Criteria Table of the banking system is drawn from the Statechart Diagram of the banking system. Using Criteria Table, State Dependency Graph of the banking system is created in Figure 2.

B. Perform Dynamic Slicing on the Slices of the STDG

A Slicing technique is basically used to facilitate the process of testing and debugging. For our research work, by applying a dynamic slicing, the state of the object in these types of the diagram is continuously changing. Dynamic slices of the State Dependency Graph are shown in figure 3. These slices result from the different possibilities which have been drawn from Start State to End State. These different possibilities are called the slices of the State Dependency Graph of the Banking System. In our work, there are three different slices which are as follows

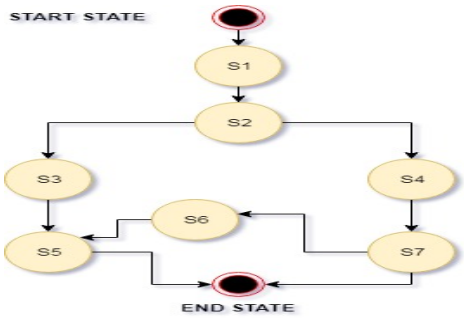

Fig. 11 State Dependency Graph (STDG) of Banking System

1. *Slice 1:* Slice 1 is a first possibility to reach at End State from Start State. This Slice is having S1, S2, S3 and S5 states.
2. *Slice 2:* Slice 2 is a second possibility to reach at End State from Start State. This Slice is having S1, S2, S4 and S7 states.
3. *Slice 3:* Slice 3 is a third possibility to reach at End State from Start State. This Slice is having S1, S2, S4, S7 and S5 states.

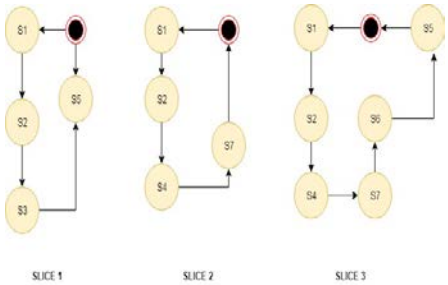

Fig. 12 Dynamic Slicing on the Slices of the STDG

C. Calculating the Complexity Metrics of State Chart Diagram

The various connecting lines of a specific state diagram represent the interaction from one state to another state. The more interactions there are, the higher the complexity of the state transition graph. Actually, we are measuring the relation of edges to nodes in a graph. The only here the nodes are not statements, but the states and the edges are not branches but interaction transitions.

TABLE VIII METRICS VALUES FOR XML CODE OF STATE CHART DIAGRAM FROM SD METRICS TOOL

Name	Trans	States	Conditions
Slice 1	5	4	0
Slice 2	5	4	0
Slice3	5	6	0

The design goal is to have as few transitions as possible since every state transition has to be tested at least once and that drives the test. The complexity of state chart diagram is computed by the equation 1

$$I = \frac{Nr\ states}{Nr\ states\ Transitions + nr\ Transitions\ Conditions_0}$$

TABLE IX COMPLEXITY FOR SLICES

Name	Complexity
Slice 1	0.80
Slice 2	0.80
Slice 3	0.83

VI. RESULTS AND DISCUSSION

According to the concept of Software Engineering, a system will be understandable, if the Cohesion of the system is high, means the high self-contained functionality of the system is high and Coupling of the system is low, means less interaction between the subparts of the system is low. If a system has high Cohesion and Low Coupling, then that system is an ideal system. Here the Complexity for Slice1&2 is 0.80, and complexity for Slice 3 is 0.83. So, using program Slicing gives accurate results if observed in contrast to previous research on the metrics of the Statechart diagram.

VII. CONCLUSION AND FUTURE SCOPE

All the classes are interrelated with each other. So the concept of cohesion as well as the coupling is implemented. The cohesion is interaction within a class and coupling is the interaction with the other classes. Always this high cohesion and low coupling are recommended in the software to develop successful software. The cohesion and coupling were the metrics and other metrics according to the requirements will be calculated in future.

REFERENCES

[1] S.V. Kumar, Santosh, "Impact of coupling and cohesion in object-oriented technology," *Journal of software engineering and applications*, Vol. 5, pp. 671-676, 2012.

[2] L. Kambow, D. Singh, "Visualizing the software metrics of state chart diagram using program slicing," *International Journal of Applied Information System (IJAIS)*, ISSN: 2249-0868, the foundation of computer science, New York, USA, Vol. 2, pp. 9, 2013.

[3] T. Rani, M. Sanyal and S. Garg, "Measuring Software Design Class Metrics: A Tool Approach," *International Journal of Engineering Research &Technology (IJERT)*, ISSN: 2278-0181, Vol. 1, No. 7, September 2012.

[4] Virtual Machinery, "Object-Oriented Software Metrics - Introduction and overview", *Virtual Machinery*, [Online]. Available: http://www.virtualmachinery.com/jhawkmetrics.htm.

[5] Kumar and S.K. Khalsa, "Determining cohesion and coupling for class diagram through slicing techniques", *IJACE*, Vol. 4, No.1, pp. 19-24, Jan-June 2012.

[6] D. Singh and A. Kamra, "Measuring Software design metrics of UML Structural and behavioural diagrams," *International Journal of computer &Mathematicl Sciences (IJCMS)*, ISSN: 2347-8527, Vol. 6, No. 5, May 2017.

[7] M. Genero, "Defining and validating metrics for conceptual models," [PhD thesis]. University of Castilla-La Mancha, 2002.

[8] Weyuker, "Evaluating software complexity measures," *IEEE Transactions on Software Engineering*, 14(9), pp.1357-1365, 1998.

[9] S. Chidamber, "A metrics suite for object-oriented design", *IEEE Transactions on Software Engineering*, June 1994.

[10] M. Seyyed, "Object-Oriented Metrics", Sharif University of Technology, *International Journal of Science And Research, Department of Computer Engineering*, January 2006.

[11] T. Mythili, "Quality Metrics Tool for Object-Oriented Programming", *International Journal of Computer Theory and Engineering*, Vol. 2, No. 5, October 2010.

[12] P. Verma, "Effect of different UML diagrams to evaluate the size metrics for different software projects", *Global Journal of Computer Science and Technology Software and Engineering*, Vol. 15, No. 8, version 1.0, February. 2015.

[13] N. Ana, "Evolution of Object-Oriented Coupling Metrics: A Sampling of 25 Years of Research," RWTH Aachen Univ., Aachen, Germany Horst Lichter , Yi Xu, pp. 16-18, May 2015.

[14] Alshammari "A Hierarchical Security Assessment Model for Object-Oriented Programs," Fac. of Science & Technol., Queensland Univ. of Technol., Brisbane, QLD, Australia, Colin Fidge, Diane Corney, pp. 13-14, July 2011.

[15] S. Garg, K.S. Kahlon and P.K. Bansal, "How to Measure Coupling in AOP from UML Diagram" *International Journal of Computer Science and Telecommunications*, Vol. 2, No. 8, November 2011.

[16] Jaiprakash, T. Lallchandani and R. Mall,"Static Slicing of UML Architectural Models,"*Journal of Object Technology*, vol. 8, No. 1, pages 159–188, January– February 2009.

[17] P. Sikka and K. Kaur, "Mingling of Program Slicing to Designing Phase", *Indian Journal of Science and Technology*, Vol. 9, No. 44, DOI: 10.17485/ijst/2016/v9i44/105091, November 2016.